On the Power of Pair Encodings: Frameworks for Predicate Cryptographic Primitives

Mridul Nandi and Tapas Pandit
Indian Statistical Institute, Kolkata
mridul@isical.ac.in,tapasgmmath@gmail.com

Abstract

Recently Attrapadung (Eurocrypt 2014) proposed a generic framework for fully (adaptively) secure predicate encryption (PE) based on a new primitive, called *pair encodings*. The author shows that if the underlying pair encoding scheme is either perfectly secure or computationally (doubly-selectively) secure, then the PE scheme will be fully secure. Although the pair encodings were solely introduced for PE, we show that these can also be used to construct predicate signatures, a signature analogue of PE. More precisely, we propose a generic construction for predicate signature (PS) from the pair encoding schemes. Our construction provides the signer privacy, and unforgeability in the adaptive-predicate model. Thereafter, we instantiate many PS schemes with new results, e.g., the first practical PS schemes for regular languages, the first attribute-based signature (ABS) scheme with constant-size signatures in adaptive-predicate model, the unbounded ABS with large universes in key-policy flavor, etc.

Following the CCA conversions of Yamada et al. (PKC 2011, 2012) and Nandi et al. (ePrint Archive: 2015/457), one can have CCA-secure PE from CPA-secure PE if the primitive PE has either verifiability or delegation. We show that the fully secure CPA-construction of Attrapadung possesses the verifiability. The aforesaid approach degrades the performance of the resultant CCA-secure PE scheme. As an alternative, we provide a direct fully secure CCA-construction for PE from the pair encoding scheme. This costs an extra computation of group element in encryption and three extra pairing computations in decryption as compared to the CPA-construction of Attrapadung.

The predicate signcryption (PSC) is a super class of the existing class, the attribute-based signcryption (ABSC), where the confidentiality, unforgeability and signer privacy are well preserved. By combining the proposed frameworks for PS and PE, we provide a generic construction for PSC from the pair encodings. It achieves the perfect privacy, and the strong unforgeability and CCA security in the adaptive-predicates model. The construction has the support of *combined-setup*, where the distributions of public parameters and keys in the underlying signature and encryption schemes are identical. The proposed PSC provides many new results, e.g., the first PSC schemes for regular languages, the first ABSC with constant-size signcryptions and constant-size keys respectively, the unbounded ABSC with large universes in adaptive-predicates model, etc.

Keywords: Pair encodings, predicate encryption, predicate signature, predicate signcryption.

1 Introduction

The dual system methodology of Waters [50] is a well known tool for constructing the 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 [50] was unreachable. Recently, Attrapadung [2] introduced a new primitive, called pair encoding schemes which are implicitly contained in many predicate encryption schemes. Using the pair encodings [2], the author proposed a generic framework for adaptively secure predicate encryption, which captures the core technique of the dual system methodology [50]. They showed that by applying the generic approach on the pair encoding, the adaptively-secure PE is possible. Their 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. Concurrently and independently, Wee [52] proposed the notion of predicate encodings which is exactly identical to the perfectly secure pair encodings of [2]. Some of the instantiations in [52] are similar to [2], viz., the ABE for small universe with improved efficiency and doubly-spatial encryption. Later, Attrapadung and Yamada [6] showed a conversion for obtaining the dual of a computationally secure pairing encoding scheme. The conversion is required to construct dual of a predicate encryption scheme which is based on computational secure pair encoding scheme.

Predicate signature (PS) [4] is a signature analogue of predicate encryption (PE) [13], where Alice signs a document under an associated data index (policy), provided Alice's key index $x \in \mathcal{X}$ is related to the associated data index $y \in \mathcal{Y}$. The term "related" is defined by a binary relation $\sim$, called predicate defined over $\mathcal{X} \times \mathcal{Y}$, where $\mathcal{X}$ and $\mathcal{Y}$ are respectively called key space and associated data space. Some times, we call the tuple $(\sim, \mathcal{X}, \mathcal{Y})$ as predicate tuple. The attribute-based signature (ABS) [36] is a larger subclass of PS. Like ABS, the predicate signature schemes are available in two forms, key-policy predicate signature (KP-PS) and signature-policy predicate signature (SP-PS). If the contents of $\mathcal{X}$ have complex representations than the contents of $\mathcal{Y}$, then the predicate signature is called KP-PS, otherwise it is SP-PS. Similar to ABS, we have two types of security, the unforgeability and signer privacy. The former ensures that the signatures are generated by a valid user and later protects from revealing the signer key index.

The concept of signcryption was introduced by Zheng [56]. Since then, many signcryption schemes [1, 38, 35] have been proposed. Among the three well known paradigms of [1], the paradigm "Commit then Encrypt and Sign ($\mathcal{C}t\mathcal{E}\&\mathcal{S}$)" runs faster as the implicit subroutines execute in parallel in signcrypt and unsigncrypt algorithms.

Attribute-based signcryption (ABSC) [24] is a natural extension of attribute-based encryption (ABE) and attribute-based signature (ABS) such that the confidentiality, unforgeability and signer privacy are well maintained. Like ABS, if the key is labeled with the set of attributes and the policies (signer policy and receiver policy) are associated with the signcryption, then the ABSC is known to be the key-policy attribute-based signcryption (KP-ABSC) and its dual form is called the signcryption-policy attribute-based signcryption (SCP-ABSC). The *combined-setup* in ABSC [17, 43] and combining public-key scheme [27, 48] allow to keep the distributions of public parameters and keys in the underlying signature and encryption schemes identical. In this setup [17, 43], the underlying ABS and ABE schemes have identical setup and key-gen algorithms. On the contrary, the setup and key-gen algorithms for ABS and ABE in *independent-setup* are not necessarily identical and they are run independently to generate the keys and public parameters for the individual primitives, ABS and ABE. We note that the same keys and public parameters are used for both the primitives, ABS and ABE in combined-setup as opposed to independent-setup. Therefore, the schemes in combined-setup have a benefit of sizes of key and public parameters than the schemes in independent-setup.

Predicate signcryption (PSC) is a generalization of the attribute-based signcryption. In this paper, we have a special interest the PSC with combined-setup. Let $(\sim, \mathcal{X}, \mathcal{Y})$ be a predicate tuple. For this setup, a user will have a key $\mathcal{SK}_x$ for some key-index $x \in \mathcal{X}$. This key will be used for both signcrypt and unsigncrypt. To signcrypt a message $m \in \mathcal{M}$, the sender has to choose two data-indices, $y_s \in \mathcal{Y}$ and $y_e \in \mathcal{Y}$.

The data-index y_s is called the sender data-index. A sender with a key $\mathcal{SK}_x$ can signcrypt the message m on behalf of community represented by y_s if $x \sim y_s$. The data-index y_e is called the receiver data-index. A receiver with a key $\mathcal{SK}_x$ can legitimately unsigncrypt a signcryption with the underlying receiver policy, y_e if $x \sim y_e$. Similar to ABSC, if the contents of $\mathcal{X}$ have complex representations than the contents of $\mathcal{Y}$, then the predicate signcryption is called key-policy predicate signcryption (KP-PSC). Its dual form is called signcryption-policy predicate signcryption (SCP-PSC). For example, consider a predicate signcryption for regular languages over an alphabet Σ. If the key-indices are represented by the regular languages over Σ and both the data-indices are represented by the strings over Σ, then the form of PSC is called KP-PSC for regular languages. Its dual form is called SCP-PSC for regular languages. Other forms of PSC are possible, but in combined-setup, these are the only forms of PSC in our consideration.

A predicate signcryption scheme is said to have public verifiability if a receiver always can convince the third party that the received message (m, y_s) was actually sent by a user whose key index satisfies y_s (policy of the sender). As pointed out in [43] that any predicate signcryption which extends the $\mathcal{C}t\mathcal{E}\&\mathcal{S}$-approach of [1] can be shown to have public verifiability. But, for that, we have to assume the relaxed-binding property of the primitive commitment scheme.

Motivation. All the predicate encryption schemes of [2, 6, 52] were shown to be CPA-secure in the adaptive-predicate model. Using the technique [54, 53, 39], the above CPA-secure constructions 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 of the traditional approach [54, 53, 39] 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 degrade the performance of decryption. Therefore, a direct CCA-secure construction of PE based on pair encoding scheme whose cost is very close to that of the existing CPA-secure construction [2] is always welcome.

The available pair encoding schemes [2, 6, 52] have been reached out for most of the practical predicate families. Therefore, it is interesting to see a construction of predicate signatures from the pair encoding schemes which were solely introduced for predicate encryptions.

Recently, Pandit, Pandey and Barua [43] proposed an ABSC scheme which supports perfect privacy, public verifiability, combined-setup and follows the $\mathcal{C}t\mathcal{E}\&\mathcal{S}$ paradigm. The strong unforgeability and IND-CCA security of the ABSC scheme were proven in the adaptive-predicates models. In current state-of-art, no generic construction of predicate signcryption with the above properties was known. Therefore, it is interesting to see any generic construction of predicate signcryptions from the pair encoding schemes along with all the above properties. The above discussion is summarized as the questions given below.

(1). Is it possible to construct a generic predicate signature scheme from the pair encoding scheme and at the same time, it enjoys all the features analogous to those of [2]?

(2). 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 [2]?

(3). Is it possible to construct a generic predicate signcryption scheme from the pair encoding scheme which achieves all the features and security of [43]?

Our Contribution. All the above questions are answered in a novel way. We provide generic constructions of predicate signature, (CCA-secure) predicate encryption and predicate signcryption schemes from the pair encoding schemes. If the underlying pair encoding scheme with a least security[1], fulfills some (natural) conditions, then the PS and PSC schemes will achieve the perfect signer privacy, and the unforgeability in adaptive-predicate(s) model. But to ensure the adaptive-predicate(s) IND-CCA security of the PE and PSC schemes, we assume either both the computational security (CMH and SMH) or the PMH security of the underlying pair encoding scheme. All the constructions are given in the setting of composite order bilinear groups. The unforgeability (of PS and PSC) and IND-CCA security (of PE and PSC) are proven under the three subgroup assumptions, DSG1, DSG2, DSG3 and the extra hardness assumption(s) required for the CMH (and SMH)-security of the underlying pair encoding scheme. If the primitive pair encoding scheme has PMH-security, then we do not need any extra hardness assumption. In this case, we say that the corresponding predicate scheme is *cost free*. Through these generic constructions what we achieved are summarized below:

- **Predicate Signature.** All the pair encoding schemes of [2, 6, 52] maintain the least security and satisfy the the natural conditions (see **Conditions 3.1** of Section 3.1). Therefore, the resultant predicate signature schemes are adaptive-predicate unforgeable and perfectly private. Our generic predicate signature can be used to derive the following new results.
  - (PS for Regular Languages.) Predicate signature schemes for regular languages in both the forms, key-policy and signature-policy are provided in this paper. Both the schemes support the large universe alphabet. To the best of our knowledge, these are the first practical non-trivial predicate signature schemes beyond ABS.
  - (Unbounded KP-ABS.) We present an unbounded KP-ABS scheme with large universes, where the size of the universe is super-polynomial and no restriction has been imposed on the access polices and sets of attributes. To the best of our knowledge, this is the first large universes KP-ABS construction with the feature *unbounded*.
  - (Constant-size Signatures and Constant-size Keys.) Till date, the only available ABS scheme [4] with constant-size signatures for general access structures is known to be unforgeable in the selective-predicate model. We propose the first KP-ABS with constant-size signatures, where the unforgeability is proven in adaptive-predicate model. A dual version, SP-ABS with constant-size keys is also provided in this paper.
  - (Cost Free Signatures.) The following predicate signatures are cost free as the underlying pair encoding schemes are PMH secure. The schemes are ABS for both small and large universes and predicate signatures for policy over doubly-spatial predicate. We can also obtain predicate signature schemes with constant-size keys and constant-size signatures respectively for both zero inner product and non-zero inner product predicates. Predicate signatures for doubly-spatial predicate and negated spatial predicate can be derived. Spatial signature schemes with constant-size keys and constant-size signatures respectively can be instantiated.
- **CCA Secure Predicate Encryption.** We obtain adaptive-predicate IND-CCA predicate encryption schemes from the pair encoding schemes via the following two approaches:
  - (Traditional Approach.) We first show that if the underlying pair encoding scheme fulfills the condition (1) of **Conditions 3.1**, then the fully secure construction in Section 4.3 of [2] satisfies

[1] We consider two notions of security for the pair encoding scheme, perfect and computational. The perfect security is called the perfectly master-key hiding (PMH). The computational security are of two types, the selectively master-key hiding (SMH) and co-selectively master-key hiding (CMH). The least security means either PMH or CMH.

verifiability (Definition 2.9). Then, by applying the CCA conversion technique [53, 54, 39], one can obtain adaptive-predicate IND-CCA predicate encryption schemes.

- (New Approach (**Direct CCA-secure PE**).) We pointed out earlier in this paper that the traditional approach has a major drawback, the degradation of performance of the decryption. This motivates us to study the direct adaptive-predicate CCA-secure construction of predicate encryptions from the pair encodings. For this construction, we assume the conditions (1) and (3) of **Conditions 3.1** on the pair encodings. It has one extra group element in ciphertext and three extra pairing computations in decryption as compared to the CPA construction of [2].

All the underlying pair encodings [2, 6, 52] satisfy the conditions (1) and (3) of **Conditions 3.1**. Therefore, using this new approach, we are able to achieve CCA security of all the predicate encryptions found in [2, 6, 52].

Recently, Blömer and Liske [8] proposed a direct CCA-secure construction of predicate encryptions from the pair encodings. Although they did not use the traditional approach, a sort of verifiability checking is involved before the actual CPA-decryption. The number of pairing computations for this checking is nearly equal to the number of paring computations in CPA-decryption. Therefore, our direct CCA-secure construction of PE has a better performance than [8].

– **Predicate Signcryption.** All the pair encoding schemes of [2, 6, 52] either have both the computational security (CMH and SMH) or the PMH security. Also, the aforementioned pair encodings fulfill the natural conditions, **Conditions 3.1**. Therefore, the resultant predicate signcryption schemes are strongly unforgeable and IND-CCA secure in adaptive-predicates models, and perfectly private. All the predicate signcryption schemes have the combined-setup, public verifiability and follow the $\mathcal{C}t\mathcal{E}\&\mathcal{S}t\mathcal{S}$ paradigm of [43]. To the best of our knowledge, all the results described below are new except the SCP-ABSC with small universes construction of [43].

- (PSC for Regular Languages.) We present predicate signcryptions for regular languages in both policies, key-policy (KP) and signcryption-policy (SCP) which support the large universe alphabet.
- (Unbounded ABSC.) Unbounded ABSC schemes with large universes in both forms, KP and SCP are provided in this paper.
- (Constant-size Signcryptions and Constant-size keys.) A KP-ABSC with constant-size signcryptions and an SCP-ABSC with constant-size keys are proposed in this paper.
- (Cost Free Signcryptions.) We can instantiate many cost free predicate signcryptions which are analogous to the cost free predicate signatures.

Outline of Our Constructions. In brief, a pair encoding scheme [2] consists of four deterministic algorithms, $\mathsf{Param}, \mathsf{Enc1}, \mathsf{Enc2}$ and Pair. Let $N \in \mathbb{N}$. $\mathsf{Param}(\boldsymbol{j}) \to n$, where $\boldsymbol{j}$ is the index for a system parameter and n describes the length of the common parameters $\boldsymbol{h} \in \mathbb{Z}_N^n$. $\mathsf{Enc1}(x) \to (\boldsymbol{k}_x, m_2)$, where $\boldsymbol{k}_x$ is a sequence of polynomials over $\mathbb{Z}_N$ with $|\boldsymbol{k}_x| = m_1$ and m_2 is length of the random coin $\boldsymbol{r} \in \mathbb{Z}_N^{m_2}$. $\mathsf{Enc2}(y) \to (\boldsymbol{c}_y, \omega_2)$, where $\boldsymbol{c}_y$ is a sequence of polynomials over $\mathbb{Z}_N$ with $|\boldsymbol{c}_y| = \omega_1$ and $\omega_2 + 1$ is length of the random coin $\boldsymbol{s} = (s_0, \ldots, s_{\omega_2}) \in \mathbb{Z}_N^{\omega_2+1}$. $\mathsf{Pair}(x, y) \to \boldsymbol{E} \in \mathbb{Z}_N^{m_1 \times \omega_1}$. The correctness says for $x \sim y$, $(\boldsymbol{k}_x, m_2) \leftarrow \mathsf{Enc1}(x)$, $(\boldsymbol{c}_y, \omega_2) \leftarrow \mathsf{Enc2}(y)$ and $\boldsymbol{E} \leftarrow \mathsf{Pair}(x, y)$, we have $\boldsymbol{k}_x(\alpha, \boldsymbol{r}, \boldsymbol{h})\boldsymbol{E}\boldsymbol{c}_y^\top(\boldsymbol{s}, \boldsymbol{h}) = \alpha s_0$.

1. Outline for PS. Before describing the outline of the predicate signature, we state the following two facts:

–**Fact 1.** A signature in nothing but a diluted key for a policy y computed from an actual (strong) key $\mathcal{SK}_x$ with $x \sim y$, where the message m and the policy y are to be binded.

–**Fact 2.** To maintain the signer privacy, the signature is labeled with policy y, at least not labeled with the key-index x.

The above facts are implicitly used in many predicate signatures and also provide an insight to the predicate signature. Let $(N := p_1p_2p_3, \mathbb{G}, \mathbb{G}_T, e)$ be a composite order bilinear groups and $g_T := e(g, g)$, where $g \in \mathbb{G}_{p_1}$ and $\mathbb{G} = \mathbb{G}_{p_1} \times \mathbb{G}_{p_2} \times \mathbb{G}_{p_3}$.

In the following, we first give an outline of the initial structure of our predicate signature using structure of predicate encryption of [2] based on pair encodings. Recall that $\mathcal{SK}_x = g^{\boldsymbol{k}_x(\alpha,\boldsymbol{r},\boldsymbol{h})} \cdot \boldsymbol{R}_3 \in \mathbb{G}^{m_1}$ with $\boldsymbol{R}_3 \in \mathbb{G}_{p_3}^{m_1}$ is the key structure of [2] for the key-index x.

- Signature Generation. Suppose Alice is playing the role of a sender. Let $\mathcal{SK}_x = g^{\boldsymbol{k}_x(\alpha,\boldsymbol{r},\boldsymbol{h})} \cdot \boldsymbol{R}_3 \in \mathbb{G}^{m_1}$ be the key of Alice. To sign a message m under a policy y with $x \sim y$, Alice first runs $\boldsymbol{E} \leftarrow \mathsf{Pair}(x, y)$. Then, it generates the signature as $\boldsymbol{\delta}_y := \mathcal{SK}_x^{E} \cdot \boldsymbol{R}_3' \in \mathbb{G}^{\omega_1}$ where $\boldsymbol{R}_3' \xleftarrow{\mathrm{U}} \mathbb{G}_{p_3}^{\omega_1}$. On simplification, we have $\boldsymbol{\delta}_y = g^{\boldsymbol{k}_x(\alpha,\boldsymbol{r},\boldsymbol{h})\boldsymbol{E}} \cdot \widetilde{\boldsymbol{R}}_3$, where $\widetilde{\boldsymbol{R}}_3 := \boldsymbol{R}_3^{E} \cdot \boldsymbol{R}_3'$. This signature $\boldsymbol{\delta}_y$ plays the role of a diluted key, derived from the actual key $\mathcal{SK}_x$.

- Signature Verification. The verification process considered here is a probabilistic one as it is performed by running the routines which are similar to the encryption and decryption of the predicate encryption of [2]. Since, a signature is a poor or diluted key, so verifying a signature is nothing but checking its capability to extract out some information from the part of a ciphertext. Therefore, to verify a signature $\boldsymbol{\delta}_y$, we first prepare a verification text (it is same as the ciphertext, but without the message m) $\mathcal{V} := (\mathcal{V}_{\mathsf{INT}} := g_T^{\alpha s_0}, \boldsymbol{\mathcal{V}}_y := g^{\boldsymbol{c}_y(\boldsymbol{s},\boldsymbol{h})})$. The signature is accepted if $e(\boldsymbol{\delta}_y, \boldsymbol{\mathcal{V}}_y) = \mathcal{V}_{\mathsf{INT}}$ else rejected. We note that the $\mathbb{G}_{p_3}$ part of $\boldsymbol{\delta}_y$ gets canceled in the verification due to the orthogonal property of the composite order bilinear groups.

- Correctness. For $x \sim y$, $e(\boldsymbol{\delta}_y, \boldsymbol{\mathcal{V}}_y) = e(g, g)^{\boldsymbol{k}_x(\alpha,\boldsymbol{r},\boldsymbol{h})\boldsymbol{E}\boldsymbol{c}_y^{\top}(\boldsymbol{s},\boldsymbol{h})} = g_T^{\alpha s_0}$, where the last equality is obtained from the correctness of pair encoding scheme.

Limitations of the initial structure of the signature. The above initial structure of the signature only shows that Alice is capable to generate such a signature. We note that neither the message nor the policy is binded in the above signature which is very crucial to guarantee unforgeability. Although the above signature is not labeled with the key-index x, it misses a very important property of the predicate signature, the perfect-privacy of the signer.

To overcome the limitations of the above signature, we have to modify the initial structure. The modifications are explained briefly in the following two steps.

- Step 1. The initial structure of the signature is $\boldsymbol{\delta}_y = g^{\boldsymbol{k}_x(\alpha,\boldsymbol{r},\boldsymbol{h})\boldsymbol{E}} \cdot \widetilde{\boldsymbol{R}}_3 \in \mathbb{G}^{\omega_1}$. To ensure unforgeability, the message m and the policy y are to be binded to $\boldsymbol{\delta}_y$. The binding is to be done in such a way that the binding part of the signature cannot be updated, once the signature has been generated. The binding is made in the following way. A collision resistance hash function $H : \{0, 1\}^* \to \mathbb{Z}_N$ and two others parameters, $g^{\theta_1}, g^{\theta_2}$ are added to the public parameters $\mathcal{PP}$. A group element, $g^{\tau(\theta_1\hbar+\theta_2)}$ is composed with the first component of $g^{\boldsymbol{k}_x(\alpha,\boldsymbol{r},\boldsymbol{h})\boldsymbol{E}}$, where $\tau \xleftarrow{\mathrm{U}} \mathbb{Z}_N$ and $\hbar = H(m, y)$. Additionally, $g^{-\tau}$ is given as a part of the signature. In other words, the modified signature becomes, $\boldsymbol{\delta}_y = g^{\boldsymbol{v}} \cdot \widetilde{\boldsymbol{R}}_3 \in \mathbb{G}^{\omega_1+1}$, where $\boldsymbol{v}$ is implicitly set to $\boldsymbol{v} := (-\tau, \boldsymbol{\psi} + \boldsymbol{k}_x(\alpha, \boldsymbol{r}, \boldsymbol{h})\boldsymbol{E}) \in \mathbb{Z}_N^{\omega_1+1}$ and $\boldsymbol{\psi} := (\tau(\theta_1\hbar+\theta_2), 0, \ldots, 0) \in \mathbb{Z}_N^{\omega_1}$.

  To verify this signature, the verification text is to be changed to $\mathcal{V} := (\mathcal{V}_{\mathsf{INT}} := g_T^{\alpha s_0}, \boldsymbol{\mathcal{V}}_y := g^{\boldsymbol{c}_y^{\mathsf{M}}(\boldsymbol{s},\theta_1,\theta_2,\boldsymbol{h})})$, where $\boldsymbol{c}_y^{\mathsf{M}}(\boldsymbol{s}, \theta_1, \theta_2, \boldsymbol{h}) := (c_0(s_0, \theta_1, \theta_2, \hbar), \boldsymbol{c}_y(\boldsymbol{s}, \boldsymbol{h})) \in \mathbb{Z}_N^{\omega_1+1}$, $c_0(s_0, \theta_1, \theta_2, \hbar) := s_0(\theta_1\hbar + \theta_2)$ and $\hbar := H(m, y)$. The verification is same as before, i.e., the signature is accepted if $e(\boldsymbol{\delta}_y,$

$\boldsymbol{\mathcal{V}}_y) = \mathcal{V}_{\mathsf{INT}}$ else rejected. For correctness of the verification, we assume that $c_{y,\iota}(\boldsymbol{s}, \boldsymbol{h}) = s_0$ for some $\iota \in [\omega_1]$.

- Step 2. For perfect-privacy, the authors of [37] assume the perfectly hiding property of the underlying non-interacting witness-indistinguishable (NIWI) scheme. For the ABS schemes of [41, 42, 43], an additional secret sharing (0-sharing) was used to assure the perfect-privacy. For perfect-privacy of the proposed signature, we explore a novel approach (for details, refer to Section 3.4) which works irrespective of the predicate families. This is done by uniformly sampling from the orthogonal space $(\mathbf{V}_{\mathsf{M}})^{\perp}$ of $\mathbf{V}_{\mathsf{M}} := \{\boldsymbol{c}_y^{\mathsf{M}}(\boldsymbol{s}, \theta_1, \theta_2, \boldsymbol{h}) \in \mathbb{Z}_N^{\omega_1+1} \mid \boldsymbol{s} := (s_0, \ldots, s_{\omega_2}) \in \mathbb{Z}_N^{\omega_2+1}\}$. The final signature of the proposed construction (for complete description, refer to Section 3.3) has the form, $\boldsymbol{\delta}_y = g^{\boldsymbol{v}+\boldsymbol{v}_{\mathsf{sp}}} \cdot \widetilde{\boldsymbol{R}_3} \in \mathbb{G}^{\omega_1+1}$, where $\boldsymbol{v}_{\mathsf{sp}} \xleftarrow{\mathrm{U}} (\mathbf{V}_{\mathsf{M}})^{\perp}$. The verification is same as before, where $\boldsymbol{v}_{\mathsf{sp}}$ gets canceled due to the orthogonality of $\boldsymbol{v}_{\mathsf{sp}}$ and $\boldsymbol{c}_y^{\mathsf{M}}(\boldsymbol{s}, \theta_1, \theta_2, \boldsymbol{h})$.

  We show that uniformly sampling from $(\mathbf{V}_{\mathsf{M}})^{\perp}$ is done by solving the homogeneous system, $\boldsymbol{A}^{\top}\boldsymbol{X} = \mathbf{0}$, where $\boldsymbol{A} \in \mathbb{Z}_N^{\omega_1 \times (\omega_2+1)}$. For $1 \leq \iota \leq \omega_1$ and $0 \leq j \leq \omega_2$, the $(\iota, j)^{th}$ entry of the matrix $\boldsymbol{A}$ is of the form, $a_{\iota,j} + \sum_{i\in[n]} a_{\iota,j,i}h_i$, where $a_{\iota,j}$ and $a_{\iota,j,i}$ are co-efficients of ι^{th} polynomial of $\boldsymbol{c}_y$. The only available information to solve the system are $a_{\iota,j}$, $a_{\iota,j,i}$ and g^{h_i} for $1 \leq \iota \leq \omega_1$, $0 \leq j \leq \omega_2$ and $i \in [n]$. Since, h_i's are not given explicitly, applying Gaussian elimination on $\boldsymbol{A}$ is troublesome. Although h_i's are not given explicitly, we manage to solve the system $\boldsymbol{A}^{\top}\boldsymbol{X} = \mathbf{0}$ smoothly. For that we assume a restriction on the underlying pair encoding scheme which is very natural. This restriction is given as condition (2) in Section 3.1. To the best of our knowledge, most of the pair encodings (in fact, all the pair encoding of [2, 6, 52]) satisfy this condition.

One of the motivations of this paper is to achieve the adaptive security. We utilize the dual system proofs of [50] in a novel way to guarantee the adaptive unforgeability of the proposed construction. For the proof of adaptive-predicate unforgeability of the proposed signature, we abstract the dual system methodology of [50] as a signature analogue of [2]. The hybrid arguments over the games in this signature analogue of the dual system proof technique follow the style of [41, 43]. However, the hybrid arguments in [41, 43] were handled for a particular ABS through the linear secret sharing scheme (LSSS). But, here we manage the dual system proof technique generically for arbitrary predicates. In this style, we consider the semi-functional (mimic) forms of the original objects, the verification text, signatures and keys. Through the hybrid arguments, we finally reach to a game, where the $\mathcal{V}_{\mathsf{INT}}$ is chosen independently and uniformly at random from $\mathbb{G}_T$. This implies that the forgery will be invalid with respect to the verification text $\mathcal{V}$.

2. Outline for Direct CCA-secure PE. We have already observed that the initial structure of the verification text for the predicate signature is similar to the structure of the ciphertext of [2]. We also notice that the verification text for the final signature has a structure which is similar to a variant of the ciphertext of [2]. In fact, if a message m is included in the verification text, the structure of the ciphertext becomes, $\mathsf{CT} := (y, C_{\mathsf{INT}} := m \cdot g_T^{\alpha s_0}, C_y^{\mathsf{M}} := g^{\boldsymbol{c}_y^{\mathsf{M}}(\boldsymbol{s},\theta_1,\theta_2,\boldsymbol{h})})$ with $\hbar = H(m, y)$. After a rearrangement, we have $\mathsf{CT} := (\mathsf{C}_{\mathsf{cpa}}, C_0 := g^{s_0(\theta_1\hbar+\theta_2)})$, where $\mathsf{C}_{\mathsf{cpa}} := (y, C_{\mathsf{INT}}, \boldsymbol{C}_y := g^{\boldsymbol{c}_y(\boldsymbol{s},\boldsymbol{h})})$. Now, look at the major concern in the construction of the CCA-secure ciphertext. It simply says that a new ciphertext (mostly well-formed or ill-formed but up to certain extent) cannot be created from a given ciphertext. In the traditional approach [53, 54], the aforementioned concern is handled by using a strong one-time signature (OTS) scheme. We also note that for the traditional approach, the $\mathsf{C}_{\mathsf{cpa}}$ part of the CT is not required. However, in this direct CCA-secure construction, the hash value, $\hbar$ appearing in C_0 is changed to $H(\mathsf{C}_{\mathsf{cpa}})$ to form CCA-secure ciphertext CT. We note that the additional component C_0 in CT is a natural replacement for OTS in the traditional approach [53, 54], where s_0 plays the role of a signing key of the OTS.

For the decryption of the CCA-ciphertext, we use the chemistry between the verification text $\boldsymbol{\mathcal{V}}_y$ and predicate signature $\boldsymbol{\delta}_y$. In fact, given a ciphertext $\mathsf{CT} = (y, C_{\mathsf{INT}}, \boldsymbol{C}_y^{\mathsf{M}} = (C_0, \boldsymbol{C}_y))$ and a key $\mathcal{SK}_x$, the decryption is performed in the following way. It first creates an alternative key $\mathcal{SK}_x^{\mathsf{M}} := g^{\boldsymbol{v}} \cdot \widetilde{\boldsymbol{R}}_3$ exactly in the same manner as $\boldsymbol{\delta}_y$ found in Step 1. Then, it returns $C_{\mathsf{INT}}/e(\mathcal{SK}_x^{\mathsf{M}}, \boldsymbol{C}_y)$. For complete description of the direct CCA-secure PE, refer to Section 6.1.

The adaptive security is obtained by modifying the dual system proof style of [2] (which incorporates the dual system methodology of [50]). This modification is required to answer the decrypt queries appropriately.

3. Outline for PSC. Our proposed predicate signature and predicate encryption have the identical key distribution and identical $\mathcal{PP}$ distribution. Using the advantage of these distributions, we take a step towards the first generic construction of predicate signcryptions from the pair encodings. Recall the outline of the construction of ciphertext in direct CCA-secure predicate encryption. It first runs the encryption of [2] (we call it $\mathsf{Encrypt}^*$) to create the CPA-secure ciphertext $\mathsf{C}_{\mathsf{cpa}}$. Then, an additional (delayed) component $C_0 := g^{s_0(\theta_1 \hbar + \theta_2)}$ is computed to complete the CCA-ciphertext $\mathsf{CT} := (\mathsf{C}_{\mathsf{cpa}}, C_0)$.

Both the proposed modules, the predicate signature and predicate encryption use the individual hash functions. In the proposed predicate signcryption, we handle the hash values in signature and encryption using a single hash function H. Let $\hbar_s$ and $\hbar_e$ be the hash values respectively used in predicate signature and predicate encryption. To distinguish the hash values $\hbar_s$ and $\hbar_e$, we respectively use the prefix bit of the hash arguments to be '1' and '0'.

Suppose Alice has a key $\mathcal{SK}_x$ for the key-index x. To signcrypt a message m under a sender's policy y_s and a receiver's policy y_e, Alice does the following. She runs in parallel $(\mathsf{com}, \mathsf{decom}) \leftarrow \mathsf{Commit}(m)$ and $(\mathsf{vk}, \mathsf{signk}) \leftarrow \mathsf{OTS.Gen}(1^\kappa)$. Computes $\hbar_s := H(1, \mathsf{vk}, y_s)$. Then, runs in parallel $\boldsymbol{\delta}_{y_s} \leftarrow \mathsf{PS.Sign}(\mathsf{vk}, \mathcal{SK}_x, y_s)$ and $\mathsf{C}_{\mathsf{cpa}} \leftarrow \mathsf{PE.Encrypt}^*(\mathsf{decom}, y_e)$. To bind all the components, Alice computes the hash value, $\hbar_e := H(0, \mathsf{com}, \boldsymbol{\delta}_{y_s}, \mathsf{vk}, \mathsf{C}_{\mathsf{cpa}})$. Similar to the direct CCA-ciphertext, she computes the delayed component $C_0 := g^{s_0(\theta_1 \hbar_e + \theta_2)}$. Then, the one-time signature is generated as $\delta_o \leftarrow \mathsf{OTS.Sign}(C_0||y_s, \mathsf{signk})$. The final signcryption is of the form, $\mathsf{U} := (\mathsf{com}, \delta := (\boldsymbol{\delta}_{y_s}, \delta_o, \mathsf{vk}), \mathsf{CT})$.

Given a signcryption U, key $\mathcal{SK}_x$ and sender's policy y_s, the unsigncrypt algorithm does the following steps. If the output of $\mathsf{OTS.Ver}(C_0||y_s, \delta_0, \mathsf{vk})$ is 0, it returns $\perp$. Otherwise, it runs in parallel $\mathsf{flag} \leftarrow \mathsf{PS.Ver}(\mathsf{vk}, \boldsymbol{\delta}_{y_s}, y_s)$ and $\mathsf{decom} \leftarrow \mathsf{PE.Decrypt}(\mathsf{CT}, \mathcal{SK}_x)$. If $\mathsf{flag} = 0$, it returns $\perp$ else $m \leftarrow \mathsf{Open}(\mathsf{com}, \mathsf{decom})$.

We note that the OTS is used mainly to provide the strong unforgeability of the signcryption. In the original $\mathcal{C}t\mathcal{E}\&\mathcal{S}$ paradigm of [1], com is signed and decom is encrypted. We deviate a bit from the approach of $\mathcal{C}t\mathcal{E}\&\mathcal{S}$ [1], where we sign (under PS) vk, instead of com. Later, we derive a simple predicate signcryption by a minor modification (viz., ignoring the primitive scheme, commitment) of the current signcryption. For details of modification, we refer to Section 7.

Again for the security of PSC, we utilize the dual system methodology of [50]. The proof for unforgeability of the proposed PSC implements the dual system proof of [50] in almost similar way as it is done for PS. But, in the unforgeability model for PSC, we have to answer the extra queries, the unsigncrypt queries. Similarly, we have to respond the extra queries, the signcrypt queries in the confidentiality model for PSC. The distributions of the objects involved in the dual system methodology for PSC are the joint distributions of the objects (of some forms) involved in the dual system methodology for PS and PE. Therefore, we re-use the part of the proofs for PS and PE in the security proof for PSC. This style is similar to [43], where we handled it for a particular ABSC through LSSS. But, in this novel approach, we are able to manage it generically for arbitrary predicates.

Related Works. In addition to the fully CPA-secure construction of PE, Attrapadung [2] showed a dual conversion for the pair encodings. If the source pair encoding P is perfectly secure, then the dual of P, denoted by $\mathbb{D}(\mathsf{P})$ is also perfectly secure encoding. Using this conversion the full security of the dual of a PE, denoted by $\mathbb{D}(\mathsf{PE})$, 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 [6] showed that the same dual conversion of [2] actually works for the computationally secure encodings. By applying this conversion on the underlying pair encoding of previously proposed KP-ABE [2], the authors achieved the first fully secure unbounded CP-ABE and a CP-ABE with short keys for Boolean formulas. Recently, Chen, Gay and Wee [19] and Attrapadung [3] proposed new generic frameworks for achieving adaptively secure ABE in the prime order bilinear groups which are nothing but the prime order version of [52] and [2] respectively. The main difference between the frameworks of [19] and [3] is that the former deals with only the perfectly secure encodings, whereas the later can deal with the computationally secure encodings.

Attribute-based signature. In the literature many ABS schemes [36, 37, 41, 23, 42, 34, 46, 47, 55, 47, 55] have been studied. Among them only the schemes [37, 41, 23] were known to achieve the signer privacy, adaptive-predicate unforgeability in the standard model and support the general access structures. In [37], the authors proposed a general framework for ABS using the credential bundle and NIWI scheme as primitives. This general framework provides the attribute-based signatures for monotone span programs in signature-policy form. The authors showed two practical instantiations of ABS in the standard model using Groth-Sahai proof system [26] for satisfiability of pairing product equations. In the first instantiation, they used Boneh-Boyen signature [10] as the candidate for credential bundle, whereas in the second instantiation, other Boneh-Boyen signature [9] was used. The ABS construction of [41] is based on the concept of dual pairing vector space of [40] and relies on DLIN assumption. The authors first utilized the dual system methodology [50] in ABS for adaptive unforgeability. The ABS of [41] is more efficient as compared to [37], since the later uses the Groth-Sahai non-interactive zero-knowledge (NIZK) proof systems [26] as building blocks. Although the performance of the ABS construction [41] defeats the same of [37], the scheme [41] has the following drawbacks. The size of the public parameters is linear to the size of sub-universe and a bound is imposed on the number of times an attribute could appear in a policy. The ABS schemes [37, 41, 23] have signature-policy form, among them the schemes [37, 41] support the large universe.

Functional signature. Bellare and Fuchsbauer [7] proposed a notion of policy-based signature which unifies the existing signatures, e.g., group signatures [16], mess signatures [14], attribute-based signatures [37], etc. For a policy-based signature (PBS) scheme, the authors defined the policy language $\mathcal{L}$ to be any member of the complexity class **NP**. In this scheme, a key $\mathcal{SK}_p$ which is associated with policy p can sign a message m (without revealing p) if $(p, m) \in \mathcal{L}$. Since $\mathcal{L} \in \mathbf{NP}$, the message m together with the witness w is to be supplied while generating the signature. If we restrict the policy language to come from the complexity class **P** ($\subseteq$ **NP**), then what we have is nothing but the predicate signatures, where the witness is computed in polynomial time. At the same time, Boyle, Goldwasser and Ivan [15] introduced a concept of functional signatures. In this signature, a key $\mathcal{SK}_f$ is associated with a function f and the key $\mathcal{SK}_f$ has the power to sign a message m if m belongs to its range. This can be considered as a special case of PBS, in which the policy language $\mathcal{L}$ is the set of all pairs, (f, m) such that m is in the range of f and the witness for (f, m) is a pre-image m under f.

The authors [7] showed a generic construction of attribute-based signature from PBS, but they did not explicitly mention the practical instantiation of ABS. If we instantiate the ABS of [7] using Groth-Ostrovsky-Sahai proof system [25] for **NP**-complete languages such as circuit satisfiability, then there is a huge blowup in the size of the signature due to Karp reduction. On the other hand, if we use Groth-Sahai proof system [26] for satisfiability of pairing product equations, then the ABS supports only the restricted predicate family, viz., the conjunction and disjunction of pairing product equations. Recently, Sakai, Attrapadung and Hanaoka [45] proposed an efficient ABS for arbitrary circuits from the symmetric external Diffie-Hellman assumption. Their ABS construction is based on the efficiency of Groth-Sahai proof system [26] and the expressiveness of Groth-Ostrovsky-Sahai proof system [25].

Attribute-based signcryption. In recent years, many ABSC schemes [49, 44, 43, 21, 17] have been proposed to deal with various aspects, e.g., efficiency, expressibility, security feature, model, etc. Among them only the scheme of [43] has the support of combined setup, signer privacy, and confidentiality and unforgeability in the adaptive-predicates model. However, we show that this can be instantiated from our framework.

Functional signcryption. Recently, Datta, Dutta and Mukhopadhyay [20] proposed a concept of functional signcryption (FSC) which provides the functionalities of functional signature [15] and functional encryption [12] together. Using indistinguishability obfuscation, CPA-secure PKE, statistical simulation-sound NIZK proof system with proof of knowledge and a weakly unforgeable signature scheme as building blocks, they proposed a generic construction of FSC. They showed an ABSC scheme for general circuits from the proposed functional signcryption. The authors did not provided any practical instantiation of ABSC scheme. As discussed above, if Groth-Ostrovsky-Sahai proof system [25] (modified form to assure statistical simulation-sound) for **NP**-complete language is applied, then size of signcryption will blow up a huge due to Karp reduction. However, this construction neither supports combined-setup nor enjoys public verifiability. The security of the ABSC scheme was shown to be secure in the selective-predicates models (Definitions 2.20 and 2.24).

Organization. This paper is organized as follows. The basic notation, composite order bilinear groups, hardness assumptions, the syntaxes and security definitions of commitment, one-time signature, predicate encryption, predicate signature, predicate signcryption and pair encoding schemes, and other related things are given in Section 2. Framework, security and instantiations of predicate signature are respectively provided in Section 3, Section 4 and Section 5. A direct-CCA secure PE and its security proof are given in Section 6. Framework, security and instantiations of predicate signcryption are illustrated respectively in Section 7, Section 8 and Section 9.

2 Preliminaries

2.1 Notations

For a set X, $x \xleftarrow{\text{R}} X$ denotes that x is randomly picked from X according to the distribution R. Likewise, $x \xleftarrow{\text{U}} X$ indicates x is uniformly selected from X. For an algorithm A and variables x, y, the notation $x \longleftarrow A(y)$ (or $A(y) \longrightarrow x$) carries the meaning that when A is run on the input y, it outputs x. The symbols, *poly* and PPT stand for polynomial and probabilistic polynomial-time respectively. For $a, b \in \mathbb{N}$, define $[a, b] := \{i \in \mathbb{N} : a \leq i \leq b\}$ and $[b] := [1, b]$. Let $str_1||\ldots||str_n$ denote the concatenation of the strings, $str_1, \ldots, str_n \in \{0,1\}^*$. For algorithms $A_1, \ldots, A_n$ and variables $x_1, \ldots, x_n, y_1, \ldots, y_n$, the notation $x_1 \longleftarrow A_1(y_1); \| \ldots ; \| \ x_n \longleftarrow A_n(y_n);$ stands for the parallel execution of $x_1 \longleftarrow A_1(y_1), \ldots, x_n \longleftarrow A_n(y_n)$.

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

For a matrix $\boldsymbol{M}$, the notations $\boldsymbol{M}^\top$ and M_{ij} denotes the transpose of $\boldsymbol{M}$ and entry of $\boldsymbol{M}$ at $(i,j)^{th}$ position respectively. The notation $\boldsymbol{M}_i$ denotes the i^{th} row of the matrix $\boldsymbol{M}$. $\mathsf{Null}(\boldsymbol{M})$ represents the nullity of the matrix $\boldsymbol{M}$. The notation $\mathbf{0}_{m\times n}$ stands for an $m \times n$ matrix with all the entries as 0. For a group $\mathbb{G}$ and $n \in \mathbb{N}$, the entries from $\mathbb{G}^n$ are assumed to be the row vectors.

Let $\mathbb{G}$ be a cyclic group of order N with respect to the group operation '$\cdot$'. For $g \in \mathbb{G}$ and $\boldsymbol{h} \in \mathbb{Z}_N^n$, we define $g^{\boldsymbol{h}} := (g^{h_1}, \ldots, g^{h_n})$. For $\boldsymbol{X}, \boldsymbol{Y} \in \mathbb{G}^n$, the notation $\boldsymbol{X}\cdot\boldsymbol{Y}$ stands for component wise group operations, i.e., $\boldsymbol{X} \cdot \boldsymbol{Y} := (X_1 \cdot Y_1, \ldots, X_n \cdot Y_n) \in \mathbb{G}^n$. For $\boldsymbol{W} \in \mathbb{G}^n$ and $\boldsymbol{E} \in \mathbb{Z}_N^{n\times m}$, we define $\boldsymbol{W}^{\boldsymbol{E}} := \boldsymbol{z} \in \mathbb{G}^m$, where $z_i := W_1^{E_{1i}} \cdots W_n^{E_{ni}}$. If $\boldsymbol{W} = g^{\boldsymbol{w}}$, for $g \in \mathbb{G}$ and $\boldsymbol{w}^\top \in \mathbb{Z}_N^n$, then we can write $\boldsymbol{W}^{\boldsymbol{E}} = g^{\boldsymbol{wE}}$.

For a matrix $\boldsymbol{A} \in \mathbb{Z}_q^{\ell\times\vartheta}$, we define the linear space $\mathsf{Ker}(\boldsymbol{A}) := \{\boldsymbol{u} \in \mathbb{Z}_q^\ell \mid \boldsymbol{u}^\top \boldsymbol{A} = \mathbf{0}\}$. For $(\boldsymbol{X}, \boldsymbol{x}) \in \mathbb{Z}_q^{\ell\times\vartheta} \times \mathbb{Z}_q^\ell$, an affine space generated by $(\boldsymbol{X}, \boldsymbol{x})$ is defined by $\mathsf{Aff}(\boldsymbol{X}, \boldsymbol{x}) := \{\boldsymbol{X}\boldsymbol{u} + \boldsymbol{x} \mid \boldsymbol{u} \in \mathbb{Z}_q^\vartheta\} \subset \mathbb{Z}_q^\ell$. The nullity of a matrix $\boldsymbol{A}$ is defined by $\mathsf{Null}(\boldsymbol{A}) :=$ the dimension of $\mathsf{Ker}(\boldsymbol{A}^\top)$.

2.2 Composite Order Bilinear Groups

Composite order bilinear groups [11, 32] are defined to be a tuple $\mathcal{J} := (N := p_1p_2p_3, \mathbb{G}, \mathbb{G}_T, e)$, where p_1, p_2, p_3 are three distinct primes and $\mathbb{G}$ and $\mathbb{G}_T$ are cyclic groups of order N and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is a map with the following properties:

1. (Bilinear). For all $g_1 \in \mathbb{G}_1$, $g_2 \in \mathbb{G}_2$ and $\forall s, t \in \mathbb{Z}_p, e(g_1^s, g_2^t) = e(g_1, g_2)^{st}$.
2. (Non-degenerate). $e(g_1, g_2)$ has order p in $\mathbb{G}_T$.
3. (Computable). There is an efficient algorithm for computing $e(g_1, g_2)$ for all $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$.

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

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

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

2.3 Hardness Assumptions in Composite Order Bilinear Groups

We describe here three Decisional SubGroup (DSG) assumptions [33] for 3 primes, DSG1, DSG2 and DSS3 in composite order bilinear groups. Let $\mathcal{J} := (N = p_1p_2p_3, \mathbb{G}, \mathbb{G}_T, e) \xleftarrow{\text{U}} \mathcal{G}_{\mathsf{cbg}}(1^\kappa)$ be the common parameters for each assumptions. In the following, we define instance for each assumption.

- DSG1. Let $g \xleftarrow{\text{U}} \mathbb{G}_{p_1}; Z_3 \xleftarrow{\text{U}} \mathbb{G}_{p_3}; T_0 \xleftarrow{\text{U}} \mathbb{G}_{p_1};\ T_1 \xleftarrow{\text{U}} \mathbb{G}_{p_1p_2}$. Define $\mathcal{D} := (\mathcal{J}, g, Z_3)$.

- DSG2. Let $g, Z_1 \xleftarrow{\mathrm{U}} \mathbb{G}_{p_1}$; $Z_2, W_2 \xleftarrow{\mathrm{U}} \mathbb{G}_{p_2}$; $W_3, Z_3 \xleftarrow{\mathrm{U}} \mathbb{G}_{p_3}$; $T_0 \xleftarrow{\mathrm{U}} \mathbb{G}_{p_1 p_3}$; $T_1 \xleftarrow{\mathrm{U}} \mathbb{G}$. Then define $\mathcal{D} := (\mathcal{J}, g, Z_1 Z_2, W_2 W_3, Z_3)$.
- DSG3. Let $\alpha, s \xleftarrow{\mathrm{U}} \mathbb{Z}_N$; $g \xleftarrow{\mathrm{U}} \mathbb{G}_{p_1}$; $W_2, Y_2, g_2 \xleftarrow{\mathrm{U}} \mathbb{G}_{p_2}$; $Z_3 \xleftarrow{\mathrm{U}} \mathbb{G}_{p_3}$; $T_0 := e(g,g)^{\alpha s}$; $T_1 \xleftarrow{\mathrm{U}} \mathbb{G}_T$. Define $\mathcal{D} := (\mathcal{J}, g, g^{\alpha} Y_2, g^s W_2, g_2, Z_3)$.

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

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

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

2.4 Some Results of Linear Algebra

We recall the three types of elementary row operations (for details, refer to [29]) on a matrix.

- **type**-1: Interchange rows i and j (in short, we write $R_i \leftrightarrow R_j$).
- **type**-2: Multiply row i by k, with $k \neq 0$ (in short, $R_i \leftarrow kR_i$).
- **type**-3: Add k times row j to row i (in short, $R_i \leftarrow R_i + kR_j$).

Similarly, we can define three types of elementary column operations. Let $\boldsymbol{\mathcal{E}}$ be a matrix obtained by applying a single elementary row operation on the identity matrix, called elementary matrix. Note that the affect of a single elementary row (resp. column) operation on a matrix $\boldsymbol{B}$ can also be obtained by pre (resp. post)-multiplying the matrix $\boldsymbol{B}$ by the corresponding elementary matrix $\boldsymbol{\mathcal{E}}$ (resp. $\boldsymbol{\mathcal{E}}^{\top}$).

Definition 2.2. A matrix $\boldsymbol{M}$ is said to be row (resp. column) equivalent to a matrix $\boldsymbol{B}$ if $\boldsymbol{M}$ is obtained from $\boldsymbol{B}$ by applying a finite sequence of elementary row (resp. column) operations.

Definition 2.3. A non-zero row of a matrix R is said to be row reduced if (1) the first non-zero entry of the row is equal to 1 (called leading 1) and (2) the column containing the leading 1 has all its other entries 0.

Definition 2.4. A matrix R is said to be row reduced if each of its non-zero rows is row reduced.

A well known result that will be used very often is given below.

Theorem 2.1. *If two matrices $\boldsymbol{B}$ and $\boldsymbol{M}$ are row equivalent, then the systems $\boldsymbol{BX} = \boldsymbol{0}$ and $\boldsymbol{MX} = \boldsymbol{0}$ have the same solutions.*

But, the scenario is slightly changed in case of column equivalence.

Theorem 2.2. *Suppose the matrix $\boldsymbol{M}$ is obtained from $\boldsymbol{B}$ by applying n elementary column operations, i.e., $\boldsymbol{B}\boldsymbol{\mathcal{E}}_1^{\top}\boldsymbol{\mathcal{E}}_2^{\top}\cdots\boldsymbol{\mathcal{E}}_n^{\top} = \boldsymbol{M}$, where $\boldsymbol{\mathcal{E}}_i$'s are elementary matrices. Then, $\boldsymbol{v}$ is a solution of the system $\boldsymbol{MX} = \boldsymbol{0}$ if and only if $\boldsymbol{\mathcal{E}}_1^{\top}\boldsymbol{\mathcal{E}}_2^{\top}\cdots\boldsymbol{\mathcal{E}}_n^{\top}\boldsymbol{v}$ is a solution of $\boldsymbol{BX} = \boldsymbol{0}$.*

Theorem 2.3. *Let R be a ring with 1. Let $\boldsymbol{B} \in R^{m \times n}$ be a matrix such that for $i \in [m]$, $B_{i1} = 1$ if $i = 1$ else 0. For $t \in R$, define $\tilde{\boldsymbol{t}} := (t, 0, \ldots, 0)^{\top} \in R^{m \times 1}$. Let $\boldsymbol{B}_{\mathsf{M}} := [\tilde{\boldsymbol{t}} : \boldsymbol{B}] \in R^{m \times (n+1)}$ be the augmented matrix. Then, $(v_1, \ldots, v_n)^{\top}$ is a solution of $\boldsymbol{BX} = \boldsymbol{0}$ if and only if for each $v_0 \in R$, $(v_0, -tv_0 + v_1, v_2, \ldots, v_n)^{\top}$ is a solution of the system $\boldsymbol{B}_{\mathsf{M}}\boldsymbol{X} = \boldsymbol{0}$.*

Proof. The proof is straightforward. □

Remark 2.1. From the above theorem, we have $\mathsf{Null}(\boldsymbol{B}_{\mathsf{M}}) = \mathsf{Null}(\boldsymbol{B}) + 1$.

Assumption: The factorization problem is intractable. For our purpose, we mainly apply the elementary row operations of type-2 and type-3. However, for simple representation of the solutions, one may use elementary row and column operations of type-1. Theorems 2.1 and 2.2 assume the fact that $k \neq 0$ (involved in type-2 operation) which implies that k in invertible. When matrices are considered over a field, then $k \neq 0$ implies k is invertible. But if the matrices are not defined over field, then we may be in trouble. Here we consider the matrix $\boldsymbol{A}$ over $\mathbb{Z}_N$, with $N = p_1p_2p_3$ which is not a field. Since we assume that the factorization problem is intractable, perhaps it can help out from the said trouble. Let $0 \neq k \in \mathbb{Z}_N$. It is sufficient to show that k is co-prime to N. If k is not a co-prime to N, then we can establish an algorithm for breaking the factorization problem in polynomial time of the parameter κ. In fact, $\gcd(k, N)$ is a non-trivial factor of N, which is a contradiction.

2.5 Commitment scheme

A non-interactive commitment scheme consists of three PPT algorithms - Setup, Commit and Open.

- Setup: It takes a security parameter κ and outputs a public commitment key $\mathcal{CK}$.
- Commit: It takes as input a message m, the public commitment key $\mathcal{CK}$ and returns a pair (com, decom), where com is a commitment of the message m and decom is the decommitment.
- Open: takes a pair (com, decom), the public commitment key $\mathcal{CK}$ as input and outputs m or $\perp$.

For correctness, it is required that[2] $\mathsf{Open}(\mathsf{Commit}(m)) = m$ for all message $m \in \mathcal{M}$, where $\mathcal{M}$ is the message space.

2.6 Security of Commitment

A commitment scheme is said to have hiding, binding and relaxed-binding properties if it satisfies the following respectively:

Hiding: For all PPT $\mathscr{A}$ the following is negligible:

$$\left| \Pr\left[\begin{array}{l} \mathcal{CK} \longleftarrow \mathsf{CSetup}(1^\kappa),\ (m_0, m_1, st) \longleftarrow \mathscr{A}(\mathcal{CK}), \\ b \xleftarrow{\text{U}} \{0,1\}, (\mathsf{com}_b, \mathsf{decom}_b) \longleftarrow \mathsf{Commit}(\mathcal{CK}, m_b), \end{array} : \mathscr{A}(\mathcal{CK}, st, \mathsf{com}_b) = b \right] - \frac{1}{2} \right|.$$

Binding: For all PPT $\mathscr{A}$ the following is negligible:

$$\Pr\left[\begin{array}{l} \mathcal{CK} \longleftarrow \mathsf{CSetup}(1^\kappa),\ (\mathsf{com}, \mathsf{decom}, \mathsf{decom}') \longleftarrow \mathscr{A}(\mathcal{CK}), \\ m \longleftarrow \mathsf{Open}(\mathsf{com}, \mathsf{decom}),\ m' \longleftarrow \mathsf{Open}(\mathsf{com}, \mathsf{decom}'), \end{array} : (m \neq m') \wedge (m, m' \neq \perp) \right].$$

Relaxed-Binding: For all PPT $\mathscr{A}$ the following is negligible:

$$\Pr\left[\begin{array}{c} \mathcal{CK} \longleftarrow \mathsf{CSetup}(1^\kappa),\ (m, st) \longleftarrow \mathscr{A}(\mathcal{CK}), (\mathsf{com}, \mathsf{decom}) \longleftarrow \mathsf{Commit}(m), \\ \mathsf{decom}' \longleftarrow \mathscr{A}(\mathcal{CK}, st, \mathsf{com}, \mathsf{decom}),\ m' \longleftarrow \mathsf{Open}(\mathsf{com}, \mathsf{decom}'), \end{array} : (m \neq m') \wedge (m' \neq \perp) \right].$$

Remark 2.2. It is immediate that the relaxed-binding property is weaker than the binding property.

[2] For brevity, we just omit $\mathcal{CK}$ in Open and Commit algorithm throughout this paper

2.7 Signature Scheme

A signature scheme consists of three PPT algorithms - Gen, Sign and Ver.

- Gen: It takes a security parameter κ as input and outputs a verification key vk and a signing key signk.
- Sign: It takes a message m and a signing key signk as input and returns a signature σ.
- Ver: It receives a message m, a signature σ and a verification key vk as input and returns a boolean value 1 for acceptance or 0 for rejection.

2.8 Strongly Unforgeable One-Time Signature

Strongly unforgeable one-time signature (OTS) model is defined as a game between a challenger $\mathscr{B}$ and an adversary $\mathscr{A}$, where $\mathscr{A}$ has to forge a signature for a message. It consists of the following phases:

Gen: The challenger $\mathscr{B}$ runs $\mathsf{Gen}(1^\kappa) \longrightarrow (\mathsf{vk}, \mathsf{signk})$. Then vk is given to the adversary $\mathscr{A}$.

Query: The adversary $\mathscr{A}$ is given access to the oracle $\mathsf{Sign}(., \mathsf{signk})$ at most once. Let (m, σ) be the corresponding query message and relied signature.

Forgery: The adversary outputs a signature (m^*, σ^*).

We say the adversary succeeds in this game if $\mathsf{Ver}(m^*, \sigma^*, \mathsf{vk}) = 1$ and $(m, \sigma) \neq (m^*, \sigma^*)$.

Let $\mathsf{Adv}^{\mathrm{sUF-CMA}}_{\mathscr{A},\mathrm{OTS}}(\kappa)$ denote the success probability for any adversary $\mathscr{A}$ in the above experiment. A signature scheme is said to be strongly unforgeable one-time signature or simply strong OTS if $\mathsf{Adv}^{\mathrm{sUF-CMA}}_{\mathscr{A},\mathrm{OTS}}(\kappa)$ is at most negligible function in κ.

2.9 Predicate Family

To define a predicate-based cryptosystem, we have to define predicate family. The predicate family is defined for an index set Λ. For most of the predicate families, the index sets are considered to be subsets of $\{\boldsymbol{j} : \boldsymbol{j} \in \mathbb{N}^i \text{ and } i \in \mathbb{N}\}$. The following definition of predicate family is adopted from [8, 2].

Definition 2.5 (Predicate Family)**.** For an arbitrary index set Λ, we define predicate family to be $\sim := \{\sim_{\boldsymbol{j}}\}_{\boldsymbol{j} \in \Lambda}$, where $\sim_{\boldsymbol{j}} : \mathcal{X}_{\boldsymbol{j}} \times \mathcal{Y}_{\boldsymbol{j}} \to \{0, 1\}$ is an indicator function, and $\mathcal{X}_{\boldsymbol{j}}$ and $\mathcal{Y}_{\boldsymbol{j}}$ are respectively called key space and associative data space.

The function $\sim_{\boldsymbol{j}}$ is also called predicate or binary relation over $\mathcal{X}_{\boldsymbol{j}} \times \mathcal{Y}_{\boldsymbol{j}}$. For $(x, y) \in \mathcal{X}_{\boldsymbol{j}} \times \mathcal{Y}_{\boldsymbol{j}}$, we write $x \sim_{\boldsymbol{j}} y$ if $\sim_{\boldsymbol{j}} (x, y) = 1$ else $x \not\sim_{\boldsymbol{j}} y$. For a predicate family, the corresponding index set Λ is called system-index space. A member $\boldsymbol{j}$ of the index space Λ is called index for system parameter or simply system-index. To design a predicate-based scheme for some predicate family, first a system-index $\boldsymbol{j}$ is fixed for that family. Then, this index will define a predicate tuple $(\sim_{\boldsymbol{j}}, \mathcal{X}_{\boldsymbol{j}}, \mathcal{Y}_{\boldsymbol{j}})$ 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 current study, there are many predicate families which are used to provide access control over the data. In the following, we describe some of the predicates. 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 $x = (x_1, \ldots, x_\ell) \in \mathcal{X}$ and $y = (y_1, \ldots, y_\ell) \in \mathcal{Y}$, we define $x \sim y$ if and only if $< x, y >= 0$. 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 $< x, y >\neq 0$. The corresponding encryption schemes are known as inner-product encryption (IPE).

(Doubly)-spatial relation. $\mathcal{X} = \mathcal{Y} := \{\mathsf{Aff}(\boldsymbol{A}, \boldsymbol{a}) \mid (\boldsymbol{A}, \boldsymbol{a}) \in \mathbb{Z}_q^{\ell \times k} \times \mathbb{Z}_q^{\ell},\ 0 \leq k \leq \ell\}$. For $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, doubly-spatial relation is defined by $x \sim^{\mathsf{ds}} y$ if and only if $y \cap x \neq \emptyset$. For spatial relation, we restrict $\mathcal{Y}$ to be $\mathbb{Z}_q^{\ell}$. In [18], the doubly-spatial relation was defined over $\mathcal{X} \times \mathcal{Y}$, where $\mathcal{X} := \{\mathsf{Ker}(\boldsymbol{X}) \mid \boldsymbol{X} \in \mathbb{Z}_q^{\ell \times k},\ 0 \leq k \leq \ell\}$ and $\mathcal{Y} := \{\mathsf{Aff}(\boldsymbol{A}, \boldsymbol{a}) \mid (\boldsymbol{A}, \boldsymbol{a}) \in \mathbb{Z}_q^{\ell \times k} \times \mathbb{Z}_q^{\ell},\ 0 \leq k \leq \ell\}$. The predicate encryption using the (doubly)-spatial relation is called (doubly)-spatial encryption ((D)SE). The authors in [18] 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 $\mathcal{X} = 2^{\mathcal{U}}$ 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 [2], 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 ℓ be a system-index for this new access structure based relation. We define $\mathcal{U} := \{\mathsf{Aff}(\boldsymbol{A}, \boldsymbol{a}) \mid (\boldsymbol{A}, \boldsymbol{a}) \in \mathbb{Z}_q^{\ell \times k} \times \mathbb{Z}_q^{\ell},\ 0 \leq k \leq \ell\}$. Let $\mathcal{X} := 2^{\mathcal{U}}$ and $\mathcal{Y}$ be the set of all policies of the form $(\boldsymbol{M}, \rho)$, where $\boldsymbol{M} \in \mathbb{Z}_q^{d \times r}$ and $\rho : [d] \to \mathcal{U}$ is a row labeling function. For $S := \{Y_1, \ldots, Y_t\} \in \mathcal{X}$ and $\mathbb{A} := (\boldsymbol{M}, \rho) \in \mathcal{Y}$, we define $S \sim \mathbb{A}$ if and only if there exist coefficients $\{\mu_i\}_{i \in \mathcal{I}}$ with $\mathcal{I} = \{i \in [d] \mid \exists Y_j \in S \text{ with } \rho(i) \sim^{\mathsf{ds}} Y_j\}$ such that $\sum_{i \in \mathcal{I}} \mu_i \boldsymbol{M}_i = (1, \boldsymbol{0})$. The encryption scheme realizing this relation is called policy over doubly-spatial encryption [6, 2].

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, Σ 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 $\delta : Q \times \Sigma \to Q$ is called transition function. The language, also called *regular language*, recognized by a deterministic finite automaton (DFA) M is defined as

$$\mathcal{L}(M) = \{\sigma_1\sigma_2 \cdots \sigma_n \in \Sigma^* : \delta(\cdots \delta(\delta(q_0, \sigma_1), \sigma_2) \cdots \sigma_n) \in F\}.$$

Let 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 δ by Tr, then a DFA M can always be represented by $(Q, \Sigma, \mathsf{Tr}, q_0, F)$. Let Σ be an alphabet, $\mathcal{X} := \Sigma^*$ and $\mathcal{Y}$ be the set of all DFAs with the same alphabet Σ. 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) [51] for regular languages.

A relation defined over $\mathcal{X} \times \mathcal{Y}$ is called symmetric if $\mathcal{X} = \mathcal{Y}$ and $x \sim y \Leftrightarrow y \sim x$ 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 2.6 (Dual predicate)**.** For a predicate tuple $(\sim, \mathcal{X}, \mathcal{Y})$, its dual predicate tuple $(\bar{\sim}, \bar{\mathcal{X}}, \bar{\mathcal{Y}})$ is defined by $\bar{\mathcal{X}} := \mathcal{Y}$, $\bar{\mathcal{Y}} := \mathcal{X}$ and for $(x, y) \in \bar{\mathcal{X}} \times \bar{\mathcal{Y}}$, $x \bar{\sim} y$ holds if and only if $y \sim x$ holds. The predicate $\bar{\sim}$ is called dual predicate of $\sim$.

Remark 2.3. In this paper, we consider predicate-based constructions for all the relations described above and their dual (for asymmetric relations). By predicate-based constructions, we mean predicate encryption (PE), predicate signature (PS) and predicate signcryption (PSC). If the underlying predicate or relation of PE, PS and PSC is not clearly stated, we assume that the PE, PS and PSC stand for one of the aforementioned relations.

Here we are interested to design the predicate signature, predicate encryption and predicate signcryption 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 $\boldsymbol{j}$ to be N as described in [2]. For notational simplicity, we omit $\boldsymbol{j}$ and write $(\sim_N, \mathcal{X}_N, \mathcal{Y}_N)$ or simply $(\sim, \mathcal{X}, \mathcal{Y})$ depending upon requirements.

Definition 2.7. (Domain-transferable [2]). 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 2.4. Attrapadung [2] showed that the equality predicate (for IBE) is domain-transferable. Since, all other predicates are defined through the equality predicate, all the predicates of [2, 52] are domain-transferable.

2.10 Predicate Encryption

A predicate encryption (PE) scheme for a predicate family $\sim$ consists of four PPT algorithms - Setup, KeyGen, $\mathsf{Encrypt}$ and $\mathsf{Decrypt}$.

- Setup: It takes a security parameter κ and a system-index $\boldsymbol{j}$ as input, outputs public parameters $\mathcal{PP}$ and master secret key $\mathcal{MSK}$.
- KeyGen: It takes as input $\mathcal{PP}$, $\mathcal{MSK}$ and a key-index $x \in \mathcal{X}$ and outputs a secret key $\mathcal{SK}_x$ corresponding to x.
- $\mathsf{Encrypt}$: 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.
- $\mathsf{Decrypt}$: It takes as input $\mathcal{PP}$, a ciphertext C and a key $\mathcal{SK}_x$. It returns a value from $\mathcal{M} \cup \{\perp\}$.

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

$$\mathsf{Decrypt}(\mathcal{PP}, \mathsf{Encrypt}(\mathcal{PP}, m, y), \mathcal{SK}_x) = m \text{ (resp. } \perp) \text{ if } x \sim y \text{ (resp. } x \not\sim y).$$

Public Data-Index PE. In the syntax of predicate encryption, we keep data-index y as a part of the ciphertext. The predicate encryption of this form is called *public data-index* PE or PE *with public data-index*.

There is another notion of predicate encryption, called PE *with hidden data-index*. In this encryption, plaintext and data-index are both concealed. Predicate encryption considered throughout this paper is the predicate encryption with public data-index. Unless stated otherwise, by a predicate encryption we mean the predicate encryption with public data-index.

Form of PE. Unlike ABE, we cannot define ciphertext-policy and key-policy for predicate encryption in general. But, for further references in this paper, we specify CP-PE and KP-PE for some relations defined in Section 2.9. A functional encryption for the acceptance relation in regular languages is called CP-FE for regular languages and the functional encryption for its dual relation is called KP-FE for regular languages. A predicate encryption for the policy over doubly-spatial relation is called ciphertext-policy over doubly-spatial encryption (CP-DSE) and the predicate encryption for its dual relation is called key-policy over doubly-spatial encryption (KP-DSE).

Definition 2.8. A ciphertext C is said to be **ill-formatted** if for all $m \in \mathcal{M}$, $y \in \mathcal{Y}$, $\Pr[\mathsf{Encrypt}(\mathcal{PP}, m, y) = \mathsf{C}] = 0$. Otherwise it is called **correctly-formatted**.

Definition 2.9 (Verifiability [53])**.** A predicate encryption scheme PE is said to have verifiability property if there is a polynomial time algorithm Verify such that for all ciphertexts C (possibly ill-formatted) with the data-index $y \in \mathcal{Y}$, and all $x, \tilde{x} \in \mathcal{X}$ with $x \sim y, \tilde{x} \sim y$, it holds that (for simplicity, we skip $\mathcal{PP}$ from $\mathsf{Decrypt}$):

- (Soundness). $\mathsf{Verify}(\mathcal{PP}, \mathsf{C}, x, \tilde{x}) = 1 \Rightarrow \mathsf{Decrypt}(\mathsf{C}, \mathcal{SK}_x) = \mathsf{Decrypt}(\mathsf{C}, \mathcal{SK}_{\tilde{x}})$.
- (Completeness). $\mathsf{Verify}(\mathcal{PP}, \mathsf{C}, x, \tilde{x}) = 1$ if C is correctly-formatted ciphertext.

Roughly speaking, verifiability verifies that a ciphertext is correctly-formatted or if it is ill-formatted then it can be decrypted to the same message under two keys with two different indices both related to the associated data-index.

2.11 Security of Predicate Encryption

Definition 2.10 (Adaptive-Predicate IND-CCA Security)**.** A PE scheme is said to be IND-CCA secure in adaptive-predicate model (or AP-IND-CCA secure) if for all PPT algorithms $\mathscr{A} := (\mathscr{A}_1, \mathscr{A}_2)$, the advantage

$$\mathsf{Adv}_{\mathscr{A},\mathsf{PE}}^{\mathrm{AP-IND-CCA}}(\kappa) := \left| \Pr\left[b = b' \wedge \mathsf{NRn}\right] - \frac{1}{2} \right|$$

in $\mathsf{Exp}_{\mathscr{A},\mathsf{PE}}^{\mathrm{AP-IND-CCA}}(\kappa)$ defined in Figure 1 is negligible function in security parameter κ, where $\mathscr{A}$ 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 $x \nsim y^*$.

- $\mathsf{KeyGen\ oracle}$ $(\mathcal{O}_K)$: Given a key-index x, oracle returns $\mathcal{SK}_x \longleftarrow \mathsf{KeyGen}(\mathcal{PP}, \mathcal{MSK}, x)$.
- $\mathsf{Decrypt\ oracle}$ $(\mathcal{O}_D)$: Given (C, x), it runs $\mathcal{SK}_x \longleftarrow \mathsf{KeyGen}(\mathcal{MSK}, x)$ if $\mathcal{SK}_x$ has not been generated previously[3], and then returns $\mathsf{Decrypt}(\mathcal{PP}, \mathsf{C}, \mathcal{SK}_x)$.

We may refer the above security model as AP-IND-CCA security model in this paper.

[3] The challenger maintains a log for storing the pairs of the forms $(x, \mathcal{SK}_x)$. Before generating a key for an index x, it searches x in the log. If x is not found, then it runs $\mathcal{SK}_x \longleftarrow \mathsf{KeyGen}(\mathcal{PP}, \mathcal{MSK}, x)$ and inserts $(x, \mathcal{SK}_x)$ in the log, otherwise it answers the query using $\mathcal{SK}_x$ available in the log.

$\underline{\mathsf{Exp}_{\mathscr{A},\mathsf{PE}}^{\mathrm{AP-IND-CCA}}(\kappa)}$:

- $(\mathcal{PP}, \mathcal{MSK}) \longleftarrow \mathsf{Setup}(1^\kappa, \boldsymbol{j})$
- $(m_0, m_1, y^*, st) \longleftarrow \mathscr{A}_1^{\{\mathcal{O}_K,\ \mathcal{O}_D\}}(\mathcal{PP})$ with $|m_0| = |m_1|$
- $b \overset{\mathrm{U}}{\longleftarrow} \{0, 1\}$
- $\mathsf{C}^* \longleftarrow \mathsf{Encrypt}(\mathcal{PP}, m_b, y^*)$
- $b' \longleftarrow \mathscr{A}_2^{\{\mathcal{O}_K,\ \mathcal{O}_D\}}(\mathcal{PP}, \mathsf{C}^*, st)$

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

Definition 2.11 (Selective-Predicate IND-CCA Security)**.** Likewise in IND-CCA security in selective-predicate model (or SP-IND-CCA security), the adversary $\mathscr{A}$ submits the challenge data-index y^* before receiving $\mathcal{PP}$ of PE. In this case, the advantage of $\mathscr{A}$ is denoted by $\mathsf{Adv}_{\mathscr{A},\mathsf{PE}}^{\mathrm{SP-IND-CCA}}(\kappa)$.

Definition 2.12 (IND-CPA Security)**.** A weaker notion of the above security can be defined similarly as above except, $\mathscr{A}$ is not given access to $\mathcal{O}_D$ oracle. It is called IND-CPA security in both adaptive-predicate (AP-IND-CPA) and selective predicate (SP-IND-CPA) models. The advantages of $\mathscr{A}$ in the respective models are denoted by $\mathsf{Adv}_{\mathscr{A},\mathsf{PE}}^{\mathrm{AP-IND-CPA}}(\kappa)$ and $\mathsf{Adv}_{\mathscr{A},\mathsf{PE}}^{\mathrm{SP-IND-CPA}}(\kappa)$.

2.12 Predicate Signature

A predicate signature (PS) scheme for a predicate family $\sim$ consists of four PPT algorithms - Setup, KeyGen, Sign and Ver.

- Setup: It takes a security parameter κ and a system-index $\boldsymbol{j}$ as input, outputs public parameters $\mathcal{PP}$ and master secret key $\mathcal{MSK}$.
- KeyGen: It takes as input $\mathcal{PP}$, $\mathcal{MSK}$ and a key-index $x \in \mathcal{X}$ and outputs a secret key $\mathcal{SK}_x$ corresponding to x.
- Sign: It takes $\mathcal{PP}$, a message $m \in \mathcal{M}$, a secret key $\mathcal{SK}_x$ and an associated data-index $y \in \mathcal{Y}$ with $x \sim y$ and returns a signature δ.
- Ver: It receives $\mathcal{PP}$, a message $m \in \mathcal{M}$, a signature δ and a claimed associated data-index y as input. It returns a boolean value 1 for acceptance or 0 for rejection.

Correctness. For all $(\mathcal{PP}, \mathcal{MSK}) \longleftarrow \mathsf{Setup}(1^\kappa, \boldsymbol{j})$, all $m \in \mathcal{M}$, all $x \in \mathcal{X}$, $\mathcal{SK}_x \longleftarrow \mathsf{KeyGen}(\mathcal{PP}, \mathcal{MSK}, x)$ and all $y \in \mathcal{Y}$ with $x \sim y$, it is required that

$$\mathsf{Ver}(\mathcal{PP}, m, \mathsf{Sign}(\mathcal{PP}, m, \mathcal{SK}_x, y), y) = 1.$$

Remark 2.5. As in ABS of [37], we assume that signer sends both signature and data-index y to receiver.

Public Data-Index PS. The predicate signature defined above allows the data-index to be publicly available to the receiver. This form of predicate signature is called *public data-index* PS or PS *with public* *data-index*. From now onwards, by predicate signature we mean the predicate signature with public data-index.

Form of PS. Similar to the form of PE, we specify signature-policy predicate signature (SP-PS) and key-policy predicate signature (KP-PS) for some relations defined in Section 2.9. A predicate signature for the access structure based relation is called signature-policy attribute-based signature (SP-ABS) for access structures and its dual form is called key-policy attribute-based signature (KP-ABS) for access structures. A predicate signature for the acceptance relation in regular languages is called SP-PS for regular languages and its dual form is called KP-PS for regular languages. A predicate signature for the policy over doubly-spatial relation is called signature-policy over doubly-spatial signature (SP-DSS) and its dual form is called key-policy over doubly-spatial signature (KP-DSS).

2.13 Security of Predicate Signature

Definition 2.13 (Signer Privacy)**.** A PS scheme is said to be perfectly private if for all $(\mathcal{PP}, \mathcal{MSK}) \longleftarrow \mathsf{Setup}(1^\kappa, \boldsymbol{j})$, all $x_1, x_2 \in \mathcal{X}$, $\mathcal{SK}_{x_1} \longleftarrow \mathsf{KeyGen}(\mathcal{PP}, \mathcal{MSK}, x_1)$, $\mathcal{SK}_{x_2} \longleftarrow \mathsf{KeyGen}(\mathcal{PP}, \mathcal{MSK}, x_2)$, all $m \in \mathcal{M}$, and all $y \in \mathcal{Y}$ with $x_1 \sim y$ and $x_2 \sim y$, the distributions of $\mathsf{Sign}(\mathcal{PP}, m, \mathcal{SK}_{x_1}, y)$ and $\mathsf{Sign}(\mathcal{PP}, m, \mathcal{SK}_{x_2}, y)$ are identical, where the random coins of the distributions are only the random coins involved in the Sign algorithm.

Note that the signer-privacy defined above is also called perfect-privacy. A predicate signature scheme with signer-privacy is called perfectly private.

Definition 2.14 (Adaptive-Predicate Unforgeability)**.** A PS scheme is said to be existential unforgeable in adaptive-predicate model (or AP-UF-CMA) if for all PPT algorithms $\mathscr{A}$, the advantage

$$\mathsf{Adv}_{\mathscr{A},\mathsf{PS}}^{\mathrm{AP-UF-CMA}}(\kappa) := \Pr\left[\mathsf{Ver}(\mathcal{PP}, m^*, \delta^*, y^*) = 1 \wedge \mathsf{NRn}\right]$$

in $\mathsf{Exp}_{\mathscr{A},\mathsf{PS}}^{\mathrm{AP-UF-CMA}}(\kappa)$ defined in Figure 2 is negligible function in κ, where $\mathscr{A}$ is provided access to key-gen oracle $\mathcal{O}_K$ and sign oracle $\mathcal{O}_{Sg}$ (described below) and NRn is a natural restriction that (m^*, x, y^*) with $x \sim y^*$ was never queried to $\mathcal{O}_{Sg}$ oracle and for each key-index x queried to $\mathcal{O}_K$, it holds that $x \not\sim y^*$.

- $\mathsf{KeyGen\ oracle}$ $(\mathcal{O}_K)$: Given a key-index x, oracle returns $\mathcal{SK}_x \longleftarrow \mathsf{KeyGen}(\mathcal{PP}, \mathcal{MSK}, x)$.
- $\mathsf{Sign\ oracle}$ $(\mathcal{O}_{Sg})$: Given (m, x, y), it runs $\mathcal{SK}_x \longleftarrow \mathsf{KeyGen}\mathcal{PP}, \mathcal{MSK}, x)$ if $\mathcal{SK}_x$ has not been generated previously, and then returns $\delta \longleftarrow \mathsf{Sign}(\mathcal{PP}, m, \mathcal{SK}_x, y)$.

$\underline{\mathsf{Exp}_{\mathscr{A},\mathsf{PS}}^{\mathrm{AP-UF-CMA}}(\kappa):}$

- $(\mathcal{PP}, \mathcal{MSK}) \longleftarrow \mathsf{Setup}(1^\kappa, \boldsymbol{j})$
- $(\delta^*, m^*, y^*) \longleftarrow \mathscr{A}^{\{\mathcal{O}_K,\ \mathcal{O}_{Sg}\}}(\mathcal{PP})$

Figure 2: Experiments for unforgeability

We may refer the above security model as AP-UF-CMA security model in this paper.

Definition 2.15 (Selective-Predicate Unforgeability)**.** There is another variant of unforgeability, called selective-predicate existential unforgeability (SP-UF-CMA), where $\mathscr{A}$ submits a challenge data-index $y^* \in \mathcal{Y}$ (later on which $\mathscr{A}$ will forge) before obtaining the $\mathcal{PP}$ of ABS.

Definition 2.16 (Strong Unforgeability)**.** The unforgeability defined in Definition 2.14 and 2.15 are called weak unforgeability because $\mathscr{A}$ is not allowed to forge on queried messages. In strong unforgeability, $\mathscr{A}$ is allowed to forge δ^* even on the queried message (m^*, y^*) but then $\delta^* \neq \delta$ must hold, where δ is a signature obtained from sign oracle on the query message (m^*, y^*).

We use the notations, AP-sUF-CMA and SP-sUF-CMA respectively for strong unforgeability in adaptive-predicate and selective-predicate models.

2.14 Predicate Signcryption

A predicate signcryption (PSC) scheme for a predicate family $\sim$ consists of four PPT algorithms - Setup, KeyGen, $\mathsf{Signcrypt}$ and $\mathsf{Unsigncrypt}$.

- Setup: It takes a security parameter κ and a system-index $\boldsymbol{j}$ as input, outputs public parameters $\mathcal{PP}$ and master secret key $\mathcal{MSK}$.
- KeyGen: It takes $\mathcal{PP}$, master secret $\mathcal{MSK}$ and a key-index $x \in \mathcal{X}$ as input and outputs a secret key $\mathcal{SK}_x$ corresponding to x.
- $\mathsf{Signcrypt}$: It takes $\mathcal{PP}$, a message $m \in \mathcal{M}$, a signing key $\mathcal{SK}_x$, an associated data-index $y_s \in \mathcal{Y}$ for sender with $x \sim y_s$ and an associated data-index $y_e \in \mathcal{Y}$ for receiver as input and returns a signcryption U for (y_s, y_e) (we assume that U implicitly contains y_e).
- $\mathsf{Unsigncrypt}$: It takes as input $\mathcal{PP}$, a signcryption U, a secret key $\mathcal{SK}_x$ and an associated data-index $y_s \in \mathcal{Y}$ for sender. It returns a value from $\mathcal{M} \cup \{\perp\}$.

Correctness. For all $(\mathcal{PP}, \mathcal{MSK}) \longleftarrow \mathsf{Setup}(1^\kappa, \boldsymbol{j})$, all $m \in \mathcal{M}$, all key-indices $x \in \mathcal{X}$, $\mathcal{SK}_x \longleftarrow \mathsf{KeyGen}(\mathcal{PP}, \mathcal{MSK}, x)$, all sender's associated data-indices $y_s \in \mathcal{Y}$ with $x \sim y_s$, all receiver's associated indices $y_e \in \mathcal{Y}$, all signcryptions $\mathsf{U} \longleftarrow \mathsf{Signcrypt}(\mathcal{PP}, m, \mathcal{SK}_x, y_s, y_e)$ and all indices $\tilde{x} \in \mathcal{X}$, $\mathcal{SK}_{\tilde{x}} \longleftarrow \mathsf{KeyGen}(\mathcal{PP}, \mathcal{MSK}, \tilde{x})$, it is required that $\mathsf{Unsigncrypt}(\mathcal{PP}, \mathsf{U}, \mathcal{SK}_{\tilde{x}}, y_s) = m$ (resp. $\perp$) if $\tilde{x} \sim y_e$ (resp. $\tilde{x} \nsim y_e$).

Remark 2.6. Similar to predicate signature, we assume that the signer sends both signcryption and sender's data-index y_s to the receiver in predicate signcryption. Again, the syntax of predicate signcryption defined above says that U implicitly contains y_e. So, the predicate signcryption allows the data-index of sender and receiver to be publicly available to the receiver. In other words, the predicate signcryption defined above is called *public data-indices* PSC or PSC *with public data-indices*. From now onwards, by predicate signcryption we mean the predicate signcryption with public data-indices. Similar to the forms of PS and PE, we can define the forms of PSC for the relations defined in Section 2.9.

2.15 Security of Predicate Signcryption

Definition 2.17 (Signer Privacy)**.** A PSC scheme is said to be perfectly private if for all $(\mathcal{PP}, \mathcal{MSK}) \longleftarrow \mathsf{Setup}(1^\kappa, \boldsymbol{j})$, all key-indices $x_1, x_2 \in \mathcal{X}$, all keys $\mathcal{SK}_{x_1} \longleftarrow \mathsf{KeyGen}(\mathcal{PP}, \mathcal{MSK}, x_1)$, $\mathcal{SK}_{x_2} \longleftarrow \mathsf{KeyGen}(\mathcal{PP}, \mathcal{MSK}, x_2)$, all messages $m \in \mathcal{M}$, all sender's associated data-indices $y_s \in \mathcal{Y}$ such that $x_1 \sim y_s$

and $x_2 \sim y_s$, and all receiver's associated data-indices $y_e \in \mathcal{Y}$, the distributions of $\mathsf{Signcrypt}(\mathcal{PP},\, m,\, \mathcal{SK}_{x_1},\, y_s,\, y_e)$ and $\mathsf{Signcrypt}(\mathcal{PP}, m, \mathcal{SK}_{x_2}, y_s, y_e)$ are identical, where the random coins of the distributions are only the random coins involved in $\mathsf{Signcrypt}$ algorithm.

The signer-privacy defined above is also called perfect-privacy. A PSC scheme with signer-privacy is called perfectly private.

Definition 2.18 (Adaptive-Predicates IND-CCA Security)**.** A PSC scheme is said to be IND-CCA secure in adaptive-predicates model (or APs-IND-CCA secure) if for all PPT algorithms $\mathscr{A} := (\mathscr{A}_1, \mathscr{A}_2)$, the advantage

$$\mathsf{Adv}_{\mathscr{A},\mathrm{PSC}}^{\mathrm{APs-IND-CCA}}(\kappa) := \left|\Pr\left[b = b' \wedge \mathsf{NRn}\right] - \frac{1}{2}\right|$$

in $\mathsf{Exp}_{\mathscr{A},\mathrm{PSC}}^{\mathrm{APs-IND-CCA}}(\kappa)$ defined in Figure 3 is negligible function in security parameter κ, where $\mathscr{A}$ is provided access to key-gen oracle $\mathcal{O}_K$, signcrypt oracle $\mathcal{O}_S$ and unsigncrypt oracle $\mathcal{O}_U$ (described below), and NRn is a natural restriction that (U^*, x, y_s^*) with $x \sim y_e^*$ was never queried to $\mathcal{O}_U$ and for each key-index x queried to $\mathcal{O}_K$, it holds that $x \not\sim y_e^*$.

- $\mathsf{KeyGen\ oracle}$ ($\mathcal{O}_K$): Given a key-index x, oracle returns $\mathcal{SK}_x \longleftarrow \mathsf{KeyGen}(\mathcal{PP},\, \mathcal{MSK}, x)$.
- $\mathsf{Signcrypt\ oracle}$ ($\mathcal{O}_S$): Given (m, x, y_s, y_e), it runs $\mathcal{SK}_x \longleftarrow \mathsf{KeyGen}(\mathcal{PP}, \mathcal{MSK}, x)$ if $\mathcal{SK}_x$ has not been generated previously, and then returns $\mathsf{Signcrypt}(\mathcal{PP}, m, \mathcal{SK}_x,\, y_s, y_e)$.
- $\mathsf{Unsigncrypt\ oracle}$ ($\mathcal{O}_U$): Given (U, x, y_s), it runs $\mathcal{SK}_x \longleftarrow \mathsf{KeyGen}(\mathcal{PP}, \mathcal{MSK}, x)$ if $\mathcal{SK}_x$ has not been generated previously, and then returns $\mathsf{Unsigncrypt}(\mathcal{PP}, \mathsf{U},\, \mathcal{SK}_x, y_s)$.

$\mathsf{Exp}_{\mathscr{A},\mathrm{PSC}}^{\mathrm{APs-IND-CCA}}(\kappa)$:

- $(\mathcal{PP}, \mathcal{MSK}) \longleftarrow \mathsf{Setup}(1^\kappa, \boldsymbol{j})$
- $(m_0, m_1, x, y_s^*, y_e^*, st) \longleftarrow \mathscr{A}_1^{\{\mathcal{O}_K,\, \mathcal{O}_S,\, \mathcal{O}_U\}}(\mathcal{PP})$ with $|m_0| = |m_1|$
- $b \overset{\mathrm{U}}{\longleftarrow} \{0, 1\}$
- $\mathsf{U}^* \longleftarrow \mathsf{Signcrypt}(\mathcal{PP}, m_b, \mathcal{SK}_x, y_s^*, y_e^*)$
- $b' \longleftarrow \mathscr{A}_2^{\{\mathcal{O}_K,\, \mathcal{O}_S,\, \mathcal{O}_U\}}(\mathcal{PP}, \mathsf{U}^*, st)$

Figure 3: Experiment for confidentiality (adaptive-predicates IND-CCA security)

We may refer the above security model as APs-IND-CCA security model in this paper.

Definition 2.19 (Selective-Predicate IND-CCA Security)**.** Similarly to Definition 2.18, except in this model, $\mathscr{A}$ has to submit the challenge receiver's data-index y_e^* before receiving $\mathcal{PP}$ of PSC and the challenge sender's data-index y_s^* in the challenge phase.

Remark 2.7. The selective-predicate IND-CCA (SP-IND-CCA) security model (Definition 2.19) is weaker than APs-IND-CCA security model (Definition 2.18).

Definition 2.20 (Selective-Predicates IND-CCA Security)**.** Similarly to Definition 2.18, except in this model, $\mathscr{A}$ has to submit the challenge receiver's data-index y_e^* and challenge sender's data-index y_s^* before receiving $\mathcal{PP}$ of PSC.

Remark 2.8. Selective-predicates IND-CCA (SPs-IND-CCA) security model (Definition 2.20) is weaker than SP-IND-CCA security model (Definition 2.19).

Definition 2.21 (Adaptive-Predicates Unforgeability)**.** A PSC scheme is said to be existential unforgeable in adaptive-predicates model (or APs-UF-CMA secure) if for all PPT $\mathscr{A}$, the advantage

$$\mathsf{Adv}^{\mathrm{APs-UF-CMA}}_{\mathscr{A},\mathrm{PSC}}(\kappa) := \Pr\left[m^* \neq \perp \wedge \mathsf{NRn}\right]$$

in $\mathsf{Exp}^{\mathrm{APs-UF-CMA}}_{\mathscr{A},\mathrm{PSC}}(\kappa)$ defined in Figure 4 is negligible function in κ, where $\mathscr{A}$ is provided access to keygen oracle $\mathcal{O}_K$, signcrypt oracle $\mathcal{O}_S$ and unsigncrypt oracle $\mathcal{O}_U$ (described below), and NRn is a natural restriction that for each tuple (m, x, y_s, y_e) queried to $\mathcal{O}_S$ oracle, $(m, y_s, y_e) \neq (m^*, y^*_s, y^*_e)$ and for each key-index $x \in \mathcal{X}$ queried to $\mathcal{O}_K$ oracle, it holds that $x \not\sim y^*_s$.

- $\mathsf{KeyGen\ oracle}$ $(\mathcal{O}_K)$: Given a key-index x, oracle returns $\mathcal{SK}_x \longleftarrow \mathsf{KeyGen}(\mathcal{PP}, \mathcal{MSK}, x)$.
- $\mathsf{Signcrypt\ oracle}$ $(\mathcal{O}_S)$: Given (m, x, y_s, y_e), it runs $\mathcal{SK}_x \longleftarrow \mathsf{KeyGen}(\mathcal{PP}, \mathcal{MSK}, x)$ if $\mathcal{SK}_x$ has not been generated previously, and then returns $\mathsf{Signcrypt}(\mathcal{PP}, m, \mathcal{SK}_x, y_s, y_e)$.
- $\mathsf{Unsigncrypt\ oracle}$ $(\mathcal{O}_U)$: Given (U, x, y_s), it runs $\mathcal{SK}_x \longleftarrow \mathsf{KeyGen}(\mathcal{PP}, \mathcal{MSK}, x)$ if $\mathcal{SK}_x$ has not been generated previously, and then returns $\mathsf{Unsigncrypt}(\mathcal{PP}, \mathsf{U}, \mathcal{SK}_x, y_s)$.

$\mathsf{Exp}^{\mathrm{APs-UF-CMA}}_{\mathscr{A},\mathrm{PSC}}(\kappa)$:

- $(\mathcal{PP}, \mathcal{MSK}) \longleftarrow \mathsf{Setup}(1^\kappa, \boldsymbol{j})$
- $(\mathsf{U}^*, y^*_s, y^*_e) \longleftarrow \mathscr{A}^{\{\mathcal{O}_K,\ \mathcal{O}_S,\ \mathcal{O}_U\}}(\mathcal{PP})$
- $m^* \longleftarrow \mathsf{Unsigncrypt}(\mathcal{PP}, \mathsf{U}^*, \mathcal{SK}_x, y^*_s, y^*_e)$ where $x \sim y^*_e$

Figure 4: Experiment for unforgeability (adaptive-predicates UF-CMA security)

We may refer the above security model as APs-UF-CMA security model in this paper.

Definition 2.22 (Adaptive-Predicates Strong Unforgeability)**.** The above unforgeability (Definition 2.21) is also called weak unforgeability in the sense that in forgery $\mathscr{A}$ is not allowed to forge for the queried messages. In strong unforgeability (we use notation, APs-sUF-CMA), the restriction $(m, y_s, y_e) \neq (m^*, y^*_s, y^*_e)$ is replaced by $(\mathsf{U}, m, y_s, y_e) \neq (\mathsf{U}^*, m^*, y^*_s, y^*_e)$, where U is the reply for the query (m, x, y_s, y_e) to $\mathcal{O}_S$ oracle.

Definition 2.23 (Selective-Predicate Strong Unforgeability)**.** Similarly to Definition 2.22, except in this model, $\mathscr{A}$ has to submit the challenge sender's data-index y^*_s before receiving $\mathcal{PP}$ of PSC and the challenge receiver's data-index y^*_e at the time of forgery.

Remark 2.9. The selective-predicate sUF-CMA (SP-sUF-CMA) security model (Definition 2.23) is weaker than APs-sUF-CMA security model (Definition 2.22).

Definition 2.24 (Selective-Predicates Strong Unforgeability)**.** Similarly to Definition 2.22, except in this model, $\mathscr{A}$ has to submit the challenge sender's data-index y^*_s and challenge receiver's data-index y^*_e before receiving $\mathcal{PP}$ of PSC.

Remark 2.10. Selective-predicates sUF-CMA (SPs-sUF-CMA) security model (Definition 2.24) is weaker than SP-sUF-CMA security model (Definition 2.23).

2.16 Pair Encoding Scheme

A Pair Encoding Scheme [2], P for a predicate family, $\sim$ consists of four deterministic algorithms, Param, $\mathsf{Enc1}$, $\mathsf{Enc2}$ and Pair.

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

– $\mathsf{Enc1}(x \in \mathcal{X}, N) \longrightarrow (\boldsymbol{k}_x := (k_1, \ldots, k_{m_1}), m_2)$, where k_ι's for $\iota \in [m_1]$ are polynomial over $\mathbb{Z}_N$ and $m_2 \in \mathbb{N}$ specifies the number of its own variables. We require that each polynomial k_ι is a linear combination of monomials, $\alpha, r_j, h_i r_j$, where $\alpha, r_1, \ldots, r_{m_2}, h_1, \ldots, h_n$ are variables. In other words, it outputs a set of coefficients $\{b_\iota, b_{\iota,j}, b_{\iota,j,i}\}_{\iota \in [m_1], j \in [m_2], i \in [n]}$ which define the sequence of polynomials

$$\Big(k_\iota(\alpha, \boldsymbol{r}, \boldsymbol{h}) := b_\iota \alpha + \big(\sum_{j \in [m_2]} b_{\iota,j} r_j\big) + \big(\sum_{\substack{j \in [m_2] \\ i \in [n]}} b_{\iota,j,i} h_i r_j\big)\Big)_{\iota \in [m_1]}$$

where $\boldsymbol{r} := (r_1, \ldots, r_{m_2})$.

– $\mathsf{Enc2}(y \in \mathcal{Y}, N) \longrightarrow (\boldsymbol{c}_y := (c_1, \ldots, c_{\omega_1}), \omega_2)$, where c_ι's for $\iota \in [\omega_1]$ are polynomial over $\mathbb{Z}_N$ and $\omega_2 \in \mathbb{N}$ specifies the number of its own variables. We require that each polynomial c_ι is a linear combination of monomials $s_j, h_i s_j$, where $s_0, \ldots, s_{\omega_2}, h_1, \ldots, h_n$ are variables. In other words, it outputs a set of coefficients $\{a_{\iota,j}, a_{\iota,j,i}\}_{\iota \in [\omega_1], j \in [0, \omega_2], i \in [n]}$ which define the sequence of polynomials

$$\Big(c_\iota(\boldsymbol{s}, \boldsymbol{h}) := \sum_{j \in [0, \omega_2]} a_{\iota,j} s_j + \sum_{\substack{j \in [0, \omega_2] \\ i \in [n]}} a_{\iota,j,i} h_i s_j\Big)_{\iota \in [\omega_1]}, \text{ where } \boldsymbol{s} := (s_0, \ldots, s_{\omega_2}).$$

– $\mathsf{Pair}(x, y, N) \longrightarrow \boldsymbol{E} \in \mathbb{Z}_N^{m_1 \times \omega_1}$.

Correctness:

1. For all $N \in \mathbb{N}$, $(\boldsymbol{k}_x, m_2) \longleftarrow \mathsf{Enc1}(x, N)$, $(\boldsymbol{c}_y, \omega_2) \longleftarrow \mathsf{Enc2}(y, N)$, and $\boldsymbol{E} \longleftarrow \mathsf{Pair}(x, y, N)$, we have $\boldsymbol{k}_x(\alpha, \boldsymbol{r}, \boldsymbol{h}) \boldsymbol{E} \boldsymbol{c}_y^\top(\boldsymbol{s}, \boldsymbol{h}) = \alpha s_0$ if $x \sim y$.
2. For $p | N$, if $(\boldsymbol{k}_x, m_2) \longleftarrow \mathsf{Enc1}(x, N)$ and $(\boldsymbol{k}'_x, m_2) \longleftarrow \mathsf{Enc1}(x, p)$, then we require that $\boldsymbol{k}'_x = \boldsymbol{k}_x \bmod p$. Similar type of condition is required for $\mathsf{Enc2}$.

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

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

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

2.17 Security of Pair Encoding Scheme

We consider two forms of security, viz., perfect security and computational security as defined in [2].

- Perfect Security: A pair encoding scheme is said to be *perfectly master-key hiding* (PMH) if for $N \in \mathbb{N}$, $x \not\sim_N y$, $n \longleftarrow \mathsf{Param}(\boldsymbol{j})$, $(\boldsymbol{k}_x, m_2) \longleftarrow \mathsf{Enc1}(x, N)$ and $(\boldsymbol{c}_y, \omega_2) \longleftarrow \mathsf{Enc2}(y, N)$, the following two distributions are identical:
$$\{\boldsymbol{c}_y(\boldsymbol{s}, \boldsymbol{h}), \boldsymbol{k}_x(\alpha, \boldsymbol{r}, \boldsymbol{h})\} \text{ and } \{\boldsymbol{c}_y(\boldsymbol{s}, \boldsymbol{h}), \boldsymbol{k}_x(0, \boldsymbol{r}, \boldsymbol{h})\}$$
where the random coins of the distributions are $\alpha \xleftarrow{\text{U}} \mathbb{Z}_N$, $\boldsymbol{h} \xleftarrow{\text{U}} \mathbb{Z}_N^n$, $\boldsymbol{s} \xleftarrow{\text{U}} \mathbb{Z}_N^{\omega_2+1}$ and $\boldsymbol{r} \xleftarrow{\text{U}} \mathbb{Z}_N^{m_2}$.

- 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 $G \in \{\text{SMH,CMH}\}$ if for $b \xleftarrow{\text{U}} \{0, 1\}$, all PPT adversary $\mathscr{A} := (\mathscr{A}_1, \mathscr{A}_2)$, the advantage $\mathsf{Adv}^{\mathrm{G}}_{\mathscr{A},\mathrm{P}}(\kappa) := \left|\Pr[\mathsf{Exp}^G_{\mathscr{A},0}(\kappa) = 1] - \Pr[\mathsf{Exp}^G_{\mathscr{A},1}(\kappa) = 1]\right|$ in the experiment $\mathsf{Exp}^G_{\mathscr{A},b}(\kappa)$ defined below is negligible function in security parameter κ:
$$\mathsf{Exp}^G_{\mathscr{A},b}(\kappa) := \left(\begin{array}{c} (N := p_1p_2p_3, \mathbb{G}, \mathbb{G}_T, e) \longleftarrow \mathcal{G}_{\mathsf{cbg}}(1^\kappa); \\ (g, g_2, g_3) \xleftarrow{\text{u}} \mathbb{G}_{p_1} \times \mathbb{G}_{p_2} \times \mathbb{G}_{p_3}; \\ \alpha \xleftarrow{\text{U}} \mathbb{Z}_N;\ n \longleftarrow \mathsf{Param}(\boldsymbol{j});\ \boldsymbol{h} \xleftarrow{\text{U}} \mathbb{Z}_N^n; \\ st \longleftarrow \mathscr{A}_1^{\mathcal{O}^1_{G,b,\alpha,\boldsymbol{h}}(.)}(g, g_2, g_3); \\ b' \longleftarrow \mathscr{A}_2^{\mathcal{O}^2_{G,b,\alpha,\boldsymbol{h}}(.)}(st) \end{array}\right)$$
where $\mathscr{A}$ is provided the access to two oracles, $\mathcal{O}^1_{G,b,\alpha,\boldsymbol{h}}(.)$ and $\mathcal{O}^2_{G,b,\alpha,\boldsymbol{h}}(.)$ defined below:
  - For Selective Security: $\mathcal{O}^1$ is allowed only once, while $\mathcal{O}^2$ is allowed to query polynomially many times
    - $\mathcal{O}^1_{\mathsf{SMH},b,\alpha,\boldsymbol{h}}(y^*)$: Run $(\boldsymbol{c}_{y^*}, \omega_2) \longleftarrow \mathsf{Enc2}(y^*, p_2)$, pick $\boldsymbol{s} \xleftarrow{\text{U}} \mathbb{Z}_N^{\omega_2+1}$ and return $\boldsymbol{C}_{y^*} := g_2^{\boldsymbol{c}_{y^*}(\boldsymbol{s},\boldsymbol{h})}$.
    - $\mathcal{O}^2_{\mathsf{SMH},b,\alpha,\boldsymbol{h}}(x)$: If $x \sim_{p_2} y^*$, return $\perp$. Run $(\boldsymbol{k}_x, m_2) \longleftarrow \mathsf{Enc1}(x, p_2)$, pick $\boldsymbol{r} \xleftarrow{\text{U}} \mathbb{Z}_N^{m_2}$ and return
$$\boldsymbol{K}_x := \begin{cases} g_2^{\boldsymbol{k}_x(0,\boldsymbol{r},\boldsymbol{h})} & \text{if } b = 0 \\ g_2^{\boldsymbol{k}_x(\alpha,\boldsymbol{r},\boldsymbol{h})} & \text{if } b = 1. \end{cases}$$
  - For Co-selective Security: Both the oracles, $\mathcal{O}^1$ and $\mathcal{O}^2$ are allowed to query only once.
    - $\mathcal{O}^1_{\mathsf{CMH},b,\alpha,\boldsymbol{h}}(x^*)$: Run $(\boldsymbol{k}_{x^*}, m_2) \longleftarrow \mathsf{Enc1}(x^*, p_2)$, pick $\boldsymbol{r} \xleftarrow{\text{U}} \mathbb{Z}_N^{m_2}$ and then return
$$\boldsymbol{K}_{x^*} := \begin{cases} g_2^{\boldsymbol{k}_{x^*}(0,\boldsymbol{r},\boldsymbol{h})} & \text{if } b = 0 \\ g_2^{\boldsymbol{k}_{x^*}(\alpha,\boldsymbol{r},\boldsymbol{h})} & \text{if } b = 1. \end{cases}$$
    - $\mathcal{O}^2_{\mathsf{CMH},b,\alpha,\boldsymbol{h}}(y)$: If $x^* \sim_{p_2} y$, return $\perp$. Run $(\boldsymbol{c}_y, \omega_2) \longleftarrow \mathsf{Enc2}(y, p_2)$, pick $\boldsymbol{s} \xleftarrow{\text{U}} \mathbb{Z}_N^{\omega_2+1}$ and then return $\boldsymbol{C}_y := g_2^{\boldsymbol{c}_y(\boldsymbol{s},\boldsymbol{h})}$.

Remark 2.11. 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)-SMH (resp. (t_1, t_2)-CMH) security. What considered in [2], are $(1, poly)$-SMH and $(1, 1)$-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.

3 Framework for Predicate Signature

For better explanation of the uniform sampling process used in sign algorithm, we define **h-free** variable for the random variables appearing in $\mathsf{Enc2}$ as follows.

Definition 3.1. A variable (or coin) s_j for some $j \in [0, \omega_2]$ appearing in $\mathsf{Enc2}$ of a pair encoding scheme is called "h-free" variable (or coin) if there exists a unique $\iota \in [\omega_1]$ such that $c_\iota(\boldsymbol{s}, \boldsymbol{h}) = a_{\iota,j} s_j$, otherwise it is called "non-h-free" variable (or coin).

3.1 Natural Requirements on Pair Encodings

For the correctness of the proposed constructions, we keep a restriction on the underlying pair encoding scheme. The condition (1) defined in **Conditions 3.1** is such a restriction on the pair encodings. The condition (1) is also used in the security proof to ensure perfectness of the simulation.

One of the important features considered in the proposed predicate signature and signcryption is the signer-privacy. To ensure the perfect-privacy of the signer in the proposed constructions, we have to uniformly sample from $\mathbf{V}^\perp := \{\boldsymbol{v} \in \mathbb{Z}_N^{\omega_1} \mid <\boldsymbol{v}, \boldsymbol{u}>= 0 \ \forall \ \boldsymbol{u} \in \mathbf{V}\}$, where $\mathbf{V} := \{\boldsymbol{c}_y(\boldsymbol{s}, \boldsymbol{h}) \in \mathbb{Z}_N^{\omega_1} \mid \boldsymbol{s} := (s_0, \ldots, s_{\omega_2}) \in \mathbb{Z}_N^{\omega_2+1}\}$. Now, finding the elements of $\mathbf{V}^\perp$ is nothing but solving the system $\boldsymbol{A}^\top \boldsymbol{X} = \mathbf{0}$, where $\boldsymbol{A}$ is a matrix of dimension $\omega_1 \times (\omega_2 + 1)$. More precisely, the matrix $\boldsymbol{A}$ is completely given by:

$$\boldsymbol{A} := \left(a_{\iota,j} + \sum_{i \in [n]} a_{\iota,j,i} h_i \right)_{\substack{1 \le \iota \le \omega_1 \\ 0 \le j \le \omega_2}}$$

Note that h_i's are not given explicitly, but available in the form of g^{h_i}, where g is a generator for the underlying group. To solve the system $\boldsymbol{A}^\top \boldsymbol{X} = \mathbf{0}$, we will apply the Gaussian elimination method which is simply a sequence of elementary row (and/or column) operations. Since, h_i's are not known, it is difficult to find the inverses of some elements of $\boldsymbol{A}$ which are required for the elementary operations of type-2. So, to smooth the process of elementary operations, we impose a restriction on the pair encodings. The condition (2) given in **Conditions 3.1** is such a restriction of the pair encoding.

The security of the proposed constructions is proven using the dual system methodology of Waters [50]. In this methodology, by applying the hybrid arguments over the hybrid games we reach to a final game. The last game change (from previous to final game) relies on the DSG3 assumption. In the final game change, to maintain the correct distribution of the semi-functional signatures (resp. alt-keys[4]) for predicate signature (resp. CCA-secure predicate encryption), we impose the following restriction on the pair encoding scheme. For $(x, y) \in \mathcal{X} \times \mathcal{Y}$ with $x \sim y$, $(\boldsymbol{k}_x, m_2) \longleftarrow \mathsf{Enc1}(x, N)$ and $\boldsymbol{E} \longleftarrow \mathsf{Pair}(x, y, N)$, we require that $\boldsymbol{k}_x(\alpha, \mathbf{0}, \mathbf{0})\boldsymbol{E} := (*, 0, \ldots, 0) \in \mathbb{Z}_N^{\omega_1}$, where $*$ is any entry from $\mathbb{Z}_N$. This restriction is fulfilled by the condition (3) defined in **Conditions 3.1** on the pair encodings. We could replace the condition (3) by the former restriction, but the condition (3) is easily evaluated on the pair encoding scheme.

Conditions 3.1 (Sufficient)**.** We put the following conditions on the pair encodings. To best of our knowledge, most of the pair encoding schemes satisfy these conditions.

1. $c_\iota(\boldsymbol{s}, \boldsymbol{h}) = s_0$ for some $\iota \in [\omega_1]$. W.l.o.g, we assume $c_1(\boldsymbol{s}, \boldsymbol{h}) = s_0$.

2. For $j \in [0, \omega_2]$, either (a) [case - s_j **is h-free**]: there is a unique $\iota \in [\omega_1]$ such that $c_\iota(\boldsymbol{s}, \boldsymbol{h}) = a_{\iota,j} s_j$ or (b) [case - s_j **is non-h-free**]: first the case-(a) has not happened, then if $a_{\iota,j,i'} \neq 0$ (appearing at

[4] A special key is created to answer the decrypt query.

(ι, j) position of the matrix $\boldsymbol{A}$) for some $\iota \in [\omega_1]$, $i' \in [n]$, we require that i' must be unique and for all $\iota \in [\omega_1]$, $i \in [n]$ with $i \neq i'$, $a_{\iota,j,i} = 0$, $a_{\iota,j} = 0$ (appearing at (ι, j) position of the matrix $\boldsymbol{A}$) and $h_{i'}$ is co-prime to N.

3. For $(x, y) \in \mathcal{X} \times \mathcal{Y}$ with $x \sim y$, let $(\boldsymbol{k}_x, m_2) \longleftarrow \mathsf{Enc1}(x, N)$ and $\boldsymbol{E} \longleftarrow \mathsf{Pair}(x, y, N)$. Suppose there are $\iota_1, \ldots, \iota_\ell \in [m_1]$ such that $b_{\iota_i} \neq 0$ for $i \in [\ell]$. W.l.o.g, we assume that $\iota_i = i$, i.e., $b_i \neq 0$ for $i \in [\ell]$. Then, we require that $E_{ij} = 0$ for $i \in [\ell]$ and $j \in [2, \omega_1]$.

We note that the 1st and 2nd conditions are put on Enc2 and 3rd condition is imposed on Enc1 and Pair. A pair encoding which satisfies the 1st condition is referred as normal in [6]. Most of the pair encoding schemes considered in [2, 52, 6] satisfy the condition 2(a), i.e., for $j \in [0, \omega_2]$, the coin s_j is h-free. For better understanding, we work out the following pair encoding schemes of [2].

The pair encoding scheme given in Figure 5 was used to realize unbounded KP-ABE with large universe. We show that this pair encoding satisfies **Conditions 3.1**. The condition (1) is so obvious. To verify the condition (2), we see that for each the random variable s_i, there is a component c_ι such that $c_\iota(\boldsymbol{s}, \boldsymbol{h}) = s_i$. Therefore, it is an example, where all the coins are h-free. For verifying the condition (3), we first notice that only $b_1 \neq 0$. Hence, we have to show that $E_{1j} = 0$ for $j \in [2, \omega_1]$. From the correctness of the scheme, we find that the monomials containing k_1 appear in the correctness are exactly $k_1 c_1$, so the first row of the matrix $\boldsymbol{E}$ must be $(1, \boldsymbol{0})$. Hence, we are done.

- Param $\rightarrow 6$. Let $\boldsymbol{h} := (h_0, h_1, \phi_1, \phi_2, \phi_3, \eta)$.
- Enc1$(\Gamma := (\boldsymbol{M}, \rho)) \rightarrow \boldsymbol{k}(\alpha, \boldsymbol{r}, \boldsymbol{h}) = (k_1, k_2, k_3, \{k_{4,i}, k_{5,i}, k_{6,i}\}_{i \in [\ell]})$, where $k_1 := \alpha + r\phi_1 + u\eta$, $k_2 := u$, $k_3 := r$, $k_{4,i} := \boldsymbol{M}_i \boldsymbol{v}^\top + r_i \phi_3$, $k_{5,i} := r_i$, $k_{6,i} := r_i(h_0 + h_1 \rho(i))$, and $v_1 := r\phi_2$, $\boldsymbol{r} := (r, u, r_1, \ldots, r_\ell, v_2, \ldots, v_k)$, $\boldsymbol{v} := (v_1, \ldots, v_k)$.
- Enc2$(S \subseteq \mathbb{Z}_N) \rightarrow \boldsymbol{c}(\boldsymbol{s}, \boldsymbol{h}) = (c_1, c_2, c_3, c_4, \{c_{5,y}\}_{y \in S}, \{c_{6,y}\}_{y \in S})$, where $c_1 := s$, $c_2 := s\eta$, $c_3 := s\phi_1 + w\phi_2$, $c_4 := w$, $c_{5,y} := w\phi_3 + s_y(h_0 + h_1 y)$, $c_{6,y} := s_y$ and $\boldsymbol{s} := (s, w, \{s_y\}_{y \in S})$.
- Correctness: If $x \sim y$, i.e., $\Gamma(S) = \mathsf{True}$, there exist reconstruction coefficients $\{\mu_i\}_{i \in \mathcal{I}}$, with $\mathcal{I} := \{i \in [\ell] \mid \rho(i) \in S\}$ such that $\sum_{i \in \mathcal{I}} \mu_i \boldsymbol{M}_i \boldsymbol{v}^\top = v_1 = r\phi_2$. So the following linear combination reveals αs as: $k_1 c_1 - k_2 c_2 - k_3 c_3 + \sum_{i \in \mathcal{I}} \mu_i (k_{4,i} c_4 - k_{5,i} c_{5,\rho(i)} + k_{6,i} c_{6,\rho(i)}) = \alpha s - rw\phi_2 + \sum_{i \in \mathcal{I}} \mu_i (\boldsymbol{M}_i \boldsymbol{v}^\top w) = \alpha s$.

Figure 5: Pair Encoding Scheme 4: Used in Unbounded KP-ABE with Large Universes

- Param$(|\mathcal{U}|) \rightarrow |\mathcal{U}| + 1$. Let $\boldsymbol{h} := (\phi, \{h_i\}_{i \in \mathcal{U}})$.
- Enc1$(S \subseteq \mathcal{U}) \rightarrow \boldsymbol{k}(\alpha, \boldsymbol{r}, \boldsymbol{h}) = (k_1 := \alpha + \phi r, \{k_{2,x} := r h_x\}_{x \in S}, k_3 := r)$, where $\boldsymbol{r} := r$.
- Enc2$(\Gamma := (\boldsymbol{M}, \rho)) \rightarrow \boldsymbol{c}(\boldsymbol{s}, \boldsymbol{h}) = (c_1, \{c_{2,i}, c_{3,i}\}_{i \in [\ell]})$, where $\boldsymbol{M} \in \mathbb{Z}_N^{\ell \times k}$, $c_1 := s$, $c_{2,i} := \phi \boldsymbol{M}_i \boldsymbol{v}^\top + s'_i h_{\rho(i)}$, $c_{3,i} := s'_i$ and $\boldsymbol{s} := (s, v_2, \ldots, v_k, s'_1, \ldots, s'_\ell)$, $\boldsymbol{v} := (s, v_2, \ldots, v_k)$.
- Correctness: If $\Gamma(S) = \mathsf{True}$, we have $\sum_{i \in \mathcal{I}} \mu_i \boldsymbol{M}_i \boldsymbol{v}^\top = \alpha$. So the following linear combination reveals αs as: $k_1 c_1 + \sum_{i \in \mathcal{I}} \mu_i (k_3 c_{2,i} - k_{2,\rho(i)} c_{3,i}) = \alpha s$.

Figure 6: Pair Encoding Scheme 10: Used in CP-ABE with Small Universes

The author [2] extracted Pair Encoding Scheme 10 (given in Figure 6) from the fully secure CP-ABE [33]. Again the condition (1) is obvious. For the random variables $s, s'_1, \ldots, s'_\ell$, the condition 2(a) holds.

But, for $v_2, \ldots, v_k$, the condition 2(b) holds. For all the v_j, the unique $h_{i'}$ is ϕ (for a clear view, see the matrix $\boldsymbol{A}^\top$ in Example 3.7). So, we require that during setup ϕ is chosen to be co-prime to N. The condition (3) works similarly as that of Pair Encoding Scheme 4.

3.2 Dual Conversion of Pair Encodings

We illustrate the dual conversion technique [2, 6] for converting a pair encoding for $\sim$ to another pair encoding for its dual predicate (Definition 2.6) $\tilde{\sim}$.

Let P be a given pair encoding scheme for the predicate $\sim$. We construct a pair encoding scheme $\mathbb{D}(\mathsf{P})$ for the predicate $\tilde{\sim}$ as follows: For $(n, \boldsymbol{h}) \leftarrow \mathsf{Param}$, we define $\overline{\mathsf{Param}} := (n+1, \bar{\boldsymbol{h}})$, where $\bar{\boldsymbol{h}} := (\boldsymbol{h}, \phi)$ and ϕ is a new variable.

- $\overline{\mathsf{Enc1}}(x, N)$: It runs $(\boldsymbol{c}'_x(\boldsymbol{s}', \boldsymbol{h}), \omega_2) \leftarrow \mathsf{Enc2}(x, N)$, where $\boldsymbol{s}' := (s'_0, \ldots, s'_{\omega_2})$. Then sets $\boldsymbol{r} := \boldsymbol{s}'$ and $\boldsymbol{k}_x(\alpha, \boldsymbol{r}, \bar{\boldsymbol{h}}) := (\boldsymbol{c}'_x(\boldsymbol{s}', \boldsymbol{h}), \alpha + \phi.\boldsymbol{s}')$. Finally, it outputs $(\boldsymbol{k}_x(\alpha, \boldsymbol{r}, \bar{\boldsymbol{h}}), \omega_2)$, where α is new variable.
- $\overline{\mathsf{Enc2}}(y, N)$: Runs $(\boldsymbol{k}'_y(\alpha', \boldsymbol{r}', \boldsymbol{h}), m_2) \leftarrow \mathsf{Enc1}(y, N)$. Then sets $\boldsymbol{s} := (s_0, \boldsymbol{r}')$ and $\boldsymbol{c}_y(\boldsymbol{s}, \bar{\boldsymbol{h}}) := (\boldsymbol{k}'_y(\phi.s_0, \boldsymbol{s}, \bar{\boldsymbol{h}}), s_0)$, and returns $(\boldsymbol{c}_y(\boldsymbol{s}, \bar{\boldsymbol{h}}), m_2)$, where s_0 is a new variable.

The correctness is verified as follows: If $x \tilde{\sim} y$, then $y \sim x$, so from the correctness of P we have

$$\boldsymbol{k}'_y(\alpha', \boldsymbol{r}', \boldsymbol{h})\boldsymbol{E}'\boldsymbol{c}'^\top_x(\boldsymbol{s}', \boldsymbol{h}) = \alpha' s'_0 = (\phi \cdot s_0)s'_0.$$

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

Proposition 3.1. *([2]) If a pair encoding scheme* P *for* $\sim$ *is perfectly master-key hiding, then the pair encoding scheme* $\mathbb{D}(\mathsf{P})$ *for* $\tilde{\sim}$ *is also perfectly master-key hiding.*

Proposition 3.2. *([6]) If a pair encoding scheme* P *for* $\sim$ *is normal and* $(1,1)$*-co-selectively master-key hiding, then the pair encoding scheme* $\mathbb{D}(\mathsf{P})$ *for* $\tilde{\sim}$ *is* $(1,1)$*-selectively master-key hiding.*

Proposition 3.3. *([6]) If a pair encoding scheme* P *for* $\sim$ *is normal and* $(1,1)$*-selectively master-key hiding, then the pair encoding scheme* $\mathbb{D}(\mathsf{P})$ *for* $\tilde{\sim}$ *is* $(1,1)$*-co-selectively master-key hiding.*

Observation 3.2. We first note that the pair encoding scheme, $\mathbb{D}(\mathsf{P})$ satisfies the condition (1) of **Conditions 3.1** due to newly added variable s_0. Let us examine the 3rd condition. W.l.o.g, we set $c_{y,1} = s_0$ and $k_{x,1} = \alpha + \phi \cdot s'_0$. The correctness of $\mathbb{D}(\mathsf{P})$ says that $\boldsymbol{k}_x(\alpha, \boldsymbol{r}, \boldsymbol{h})\boldsymbol{E}\boldsymbol{c}^\top_y(\boldsymbol{s}, \boldsymbol{h}) = k_{x,1} \cdot c_{y,1} - \boldsymbol{k}'_y(\alpha', \boldsymbol{r}', \boldsymbol{h})\boldsymbol{E}'\boldsymbol{c}'^\top_x(\boldsymbol{s}', \boldsymbol{h}) = \alpha s_0$. If $\boldsymbol{E}'$ has dimension $(m'_1 \times \omega'_1)$, then the dimension of $\boldsymbol{E}$ is $(m_1 \times \omega_1)$, where $m_1 = \omega'_1 + 1$ and $\omega_1 = m'_1 + 1$. Hence, the matrix, $\boldsymbol{E}$ 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 $\mathbb{D}(\mathsf{P})$ satisfies the condition (3) of **Conditions 3.1**. We note that the condition (2) of **Conditions 3.1** is imposed on the $\mathsf{Enc2}$, similarly it could be defined over $\mathsf{Enc1}$ and let us call it condition $(\bar{2})$. One can verify that if a pair encoding scheme P for predicate, $\sim$ fulfills the condition $(\bar{2})$, then its dual, $\mathbb{D}(\mathsf{P})$ for $\tilde{\sim}$ satisfies the condition (2). So far, we check that dual of all the pair encoding schemes [2, 6, 52] satisfy **Conditions 3.1**. Therefore, all the pair encoding schemes of [2, 6, 52] and their dual satisfy **Conditions 3.1** and, have either computational security (CMH and SMH) or the PMH security.

3.3 Predicate Signature from Pair Encoding Scheme

Terminology: For fixed $\theta_1, \theta_2, \hbar \in \mathbb{Z}_N$ and $\boldsymbol{h} \in \mathbb{Z}_N^n$, we define $\boldsymbol{h}_{\mathsf{M}} := (\theta_1, \theta_2, \boldsymbol{h})$, $\boldsymbol{\theta} := (\theta_1, \theta_2, \hbar)$ and $c_0(z, \boldsymbol{\theta}) := z(\theta_1 \hbar + \theta_2)$, where z is an independent variable. Note that $\theta_1, \theta_2, \hbar$ and $\boldsymbol{h}$ will be understood from the context. For $(\boldsymbol{c}_y, \omega_2) \longleftarrow \mathsf{Enc2}(y, N)$, we define $\boldsymbol{c}_y^{\mathsf{M}} := (c_0, \boldsymbol{c}_y)$, so $|\boldsymbol{c}_y^{\mathsf{M}}| = \omega_1 + 1$ if $|\boldsymbol{c}_y| = \omega_1$. Therefore, we can write $\boldsymbol{c}_y^{\mathsf{M}}(\boldsymbol{s}, \boldsymbol{h}_{\mathsf{M}}) = (c_0(s_0, \boldsymbol{\theta}), \boldsymbol{c}_y(\boldsymbol{s}, \boldsymbol{h}))$ for $\boldsymbol{s} := (s_0, \ldots, s_{\omega_2}) \in \mathbb{Z}_N^{\omega_2+1}$. We define[5] $\mathbf{V}_{\mathsf{M}} := \{\boldsymbol{c}_y^{\mathsf{M}}(\boldsymbol{s}, \boldsymbol{h}_{\mathsf{M}}) \in \mathbb{Z}_N^{\omega_1+1} \mid \boldsymbol{s} := (s_0, \ldots, s_{\omega_2}) \in \mathbb{Z}_N^{\omega_2+1}\}$. Now, we define an orthogonal set to be $(\mathbf{V}_{\mathsf{M}})^{\perp} := \{\boldsymbol{v}_{\mathsf{sp}} \in \mathbb{Z}_N^{\omega_1+1} \mid \ < \boldsymbol{v}_{\mathsf{sp}}, \boldsymbol{u} >= 0 \ \forall \ \boldsymbol{u} \in \mathbf{V}_{\mathsf{M}}\}$. The process of sampling from $(\mathbf{V}_{\mathsf{M}})^{\perp}$ is given in Section 3.4.

Let $\mathsf{P} := (\mathsf{Param}, \mathsf{Enc1}, \mathsf{Enc2}, \mathsf{Pair})$ be a primitive pair encoding scheme which satisfies **Conditions 3.1**.

- $\mathsf{Setup}(1^\kappa, \boldsymbol{j})$: It executes $\mathcal{J} := (N := p_1p_2p_3, \mathbb{G}, \mathbb{G}_T, e) \longleftarrow \mathcal{G}_{\mathsf{cbg}}(1^\kappa)$ and chooses $g \xleftarrow{\mathrm{U}} \mathbb{G}_{p_1}; Z_3 \xleftarrow{\mathrm{U}} \mathbb{G}_{p_3}$. Then runs $n \longleftarrow \mathsf{Param}(\boldsymbol{j})$ and picks $\boldsymbol{h} \xleftarrow{\mathrm{U}} \mathbb{Z}_N^n$. Again picks $\alpha, \theta_1, \theta_2 \xleftarrow{\mathrm{U}} \mathbb{Z}_N$ and sets $\boldsymbol{h}_{\mathsf{M}} := (\theta_1, \theta_2, \boldsymbol{h}) \in \mathbb{Z}_N^{n+2}$. Let $H : \{0,1\}^* \longrightarrow \mathbb{Z}_N$ be a hash function. The public parameters and master secret are given by

$$\mathcal{PP} := (\mathcal{J}, g, g^{\boldsymbol{h}_{\mathsf{M}}}, g_T^{\alpha} := e(g,g)^{\alpha}, Z_3, H) \text{ and } \mathcal{MSK} := (\alpha).$$

- $\mathsf{KeyGen}(\mathcal{PP}, \mathcal{MSK}, x)$: It runs $(\boldsymbol{k}_x, m_2) \longleftarrow \mathsf{Enc1}(x, N)$. Let $|\boldsymbol{k}_x| = m_1$. Picks $\boldsymbol{r} \xleftarrow{\mathrm{U}} \mathbb{Z}_N^{m_2}$ and $\boldsymbol{R}_3 \xleftarrow{\mathrm{U}} \mathbb{G}_{p_3}^{m_1}$. It outputs the secret key

$$\mathcal{SK}_x := (x, \ \boldsymbol{K}_x := g^{\boldsymbol{k}_x(\alpha, \boldsymbol{r}, \boldsymbol{h})} \cdot \boldsymbol{R}_3).$$

- $\mathsf{Sign}(\mathcal{PP}, m, \mathcal{SK}_x, y)$: If $x \not\sim y$, returns $\perp$. Let $\mathcal{SK}_x = (x, \ \boldsymbol{K}_x)$. It runs[6] $\boldsymbol{K}_x := g^{\boldsymbol{k}_x(\alpha, \boldsymbol{r}, \boldsymbol{h})} \cdot \boldsymbol{R}_3 \longleftarrow \mathsf{Re\text{-}Randomize}(\boldsymbol{K}_x)$ and $\mathsf{Pair}(x, y) \longrightarrow \boldsymbol{E} \in \mathbb{Z}_N^{m_1 \times \omega_1}$. Then, computes $\hbar := H(m, y)$. It picks $\tau \xleftarrow{\mathrm{U}} \mathbb{Z}_N$, $\boldsymbol{v}_{\mathsf{sp}} \xleftarrow{\mathrm{U}} (\mathbf{V}_{\mathsf{M}})^{\perp}$ and $\boldsymbol{R}_3' \xleftarrow{\mathrm{U}} \mathbb{G}_{p_3}^{\omega_1+1}$. Sets $\boldsymbol{v} := (-\tau, \boldsymbol{\psi} + \boldsymbol{k}_x(\alpha, \boldsymbol{r}, \boldsymbol{h})\boldsymbol{E}) \in \mathbb{Z}_N^{\omega_1+1}$, where $\boldsymbol{\psi} := (\tau(\theta_1\hbar + \theta_2), 0, \ldots, 0) \in \mathbb{Z}_N^{\omega_1}$. The signature is given by

$$\boldsymbol{\delta}_y := g^{\boldsymbol{v} + \boldsymbol{v}_{\mathsf{sp}}} \cdot (1_{\mathbb{G}}, \boldsymbol{R}_3^{\boldsymbol{E}}) \cdot \boldsymbol{R}_3' \in \mathbb{G}^{\omega_1+1}.$$

  where $1_{\mathbb{G}}$ is the zero element of the source group $\mathbb{G}$. We note that $\boldsymbol{\delta}_y$ can be easily computed from $\mathcal{SK}_x$, $g^{\boldsymbol{h}_{\mathsf{M}}}$, $\boldsymbol{E}$ and the random coins involved in the sign algorithm. In fact, $\boldsymbol{\delta}_y$ is computed as follows:

$$\boldsymbol{\delta}_y = \underbrace{(g^{-\tau}, 1_{\mathbb{G}}, \ldots, 1_{\mathbb{G}})}_{\mathbb{G}^{\omega_1+1}} \cdot \underbrace{(1_{\mathbb{G}}, (g^{\theta_1})^{\tau\hbar} \cdot (g^{\theta_2})^{\tau}, 1_{\mathbb{G}}, \ldots, 1_{\mathbb{G}})}_{\mathbb{G}^{\omega_1+1}} \cdot \underbrace{(1_{\mathbb{G}}, \boldsymbol{K}_x^{\boldsymbol{E}})}_{\mathbb{G}^{\omega_1+1}} \cdot \underbrace{g^{\boldsymbol{v}_{\mathsf{sp}}}}_{\mathbb{G}^{\omega_1+1}} \cdot \underbrace{\boldsymbol{R}_3'}_{\mathbb{G}^{\omega_1+1}} .$$

- $\mathsf{Ver}(\mathcal{PP}, m, \boldsymbol{\delta}_y, y)$: It runs $(\boldsymbol{c}_y, \omega_2) \longleftarrow \mathsf{Enc2}(y, N)$ and picks $\boldsymbol{s} := (s_0, s_1, \ldots, s_{\omega_2}) \xleftarrow{\mathrm{U}} \mathbb{Z}_N^{\omega_2+1}$. Computes $\boldsymbol{c}_y^{\mathsf{M}}(\boldsymbol{s}, \boldsymbol{h}_{\mathsf{M}}) := (c_0(s_0, \boldsymbol{\theta}), \boldsymbol{c}_y(\boldsymbol{s}, \boldsymbol{h})) \in \mathbb{Z}_N^{\omega_1+1}$, where $|\boldsymbol{c}_y| = \omega_1$, $\boldsymbol{\theta} := (\theta_1, \theta_2, \hbar)$, $\hbar := H(m, y)$ and $c_0(s_0, \boldsymbol{\theta}) := s_0(\theta_1\hbar + \theta_2)$. Then computes a verification text, $\mathcal{V} := (\mathcal{V}_{\mathsf{INT}} := g_T^{\alpha s_0}, \boldsymbol{\mathcal{V}}_y := g^{\boldsymbol{c}_y^{\mathsf{M}}(\boldsymbol{s}, \boldsymbol{h}_{\mathsf{M}})})$. It returns 1 if $e(\boldsymbol{\delta}_y, \boldsymbol{\mathcal{V}}_y) = \mathcal{V}_{\mathsf{INT}}$ else 0.

[5] We note that the set $\mathbf{V}_{\mathsf{M}}$ depends on $\boldsymbol{c}_y^{\mathsf{M}}$. A natural notation for the set could be $\mathbf{V}_{\boldsymbol{c}_y^{\mathsf{M}}}$, but for simplicity, we use $\mathbf{V}_{\mathsf{M}}$.

[6] The linear property of the pair encodings guarantees the re-randomization of the keys. In fact, let $\boldsymbol{K}_x = g^{\boldsymbol{k}_x(\alpha, \tilde{\boldsymbol{r}}, \boldsymbol{h})} \cdot \widetilde{\boldsymbol{R}_3}$, where $\tilde{\boldsymbol{r}} \in \mathbb{Z}_N^{m_2}$, $\widetilde{\boldsymbol{R}_3} \in \mathbb{G}_{p_3}^{m_1}$ and $(\boldsymbol{k}_x, m_2) \longleftarrow \mathsf{Enc1}(x, N)$. $\mathsf{Re\text{-}Randomize}$ picks $\boldsymbol{r}' \xleftarrow{\mathrm{U}} \mathbb{Z}_N^{m_2}$ and $\boldsymbol{R}_3' \xleftarrow{\mathrm{U}} \mathbb{G}_{p_3}^{m_1}$ and sets $\boldsymbol{K}_x := g^{\boldsymbol{k}_x(\alpha, \tilde{\boldsymbol{r}}, \boldsymbol{h})} \cdot \widetilde{\boldsymbol{R}_3} \cdot g^{\boldsymbol{k}_x(0, \boldsymbol{r}', \boldsymbol{h})} \cdot \boldsymbol{R}_3' = g^{\boldsymbol{k}_x(\alpha, \boldsymbol{r}, \boldsymbol{h})} \cdot \boldsymbol{R}_3$, where $\boldsymbol{r} := \tilde{\boldsymbol{r}} + \boldsymbol{r}' \in \mathbb{Z}_N^{m_2}$ and $\boldsymbol{R}_3 := \widetilde{\boldsymbol{R}}_3 \cdot \boldsymbol{R}_3' \in \mathbb{G}_{p_3}^{m_1}$.

Correctness: For $x \sim_N y$ ($\Rightarrow x \sim_{p_1} y$ by domain-transferability), we have

$$\begin{aligned}
e(\boldsymbol{\delta}_y, \boldsymbol{\mathcal{V}}_y) &= g_T^{<\boldsymbol{v}+\boldsymbol{v}_{\mathsf{sp}},\ \boldsymbol{c}_y^{\mathsf{M}}(\boldsymbol{s},\boldsymbol{h}_{\mathsf{M}})>} && \begin{pmatrix}\text{by orthogonality}\\ \text{of CBG}\end{pmatrix}\\
&= g_T^{<\boldsymbol{v},\ \boldsymbol{c}_y^{\mathsf{M}}(\boldsymbol{s},\boldsymbol{h}_{\mathsf{M}})>} && (\text{since } \boldsymbol{v}_{\mathsf{sp}} \in (\mathbf{V}_{\mathsf{M}})^{\perp})\\
&= g_T^{<(-\tau,\ 0,\ldots,0)+(0,\ \boldsymbol{\psi})+(0,\ \boldsymbol{k}_x(\alpha,\boldsymbol{r},\boldsymbol{h})\boldsymbol{E},\ \boldsymbol{c}_y^{\mathsf{M}}(\boldsymbol{s},\boldsymbol{h}_{\mathsf{M}})>} && (\text{by definition of } \boldsymbol{v})\\
&= g_T^{-\tau c_0(s,\boldsymbol{\theta})+\tau(\theta_1\hbar+\theta_2)c_{y,1}(\boldsymbol{s},\boldsymbol{h})+<\boldsymbol{k}_x(\alpha,\boldsymbol{r},\boldsymbol{h})\boldsymbol{E},\ \boldsymbol{c}_y(\boldsymbol{s},\boldsymbol{h}_{\mathsf{M}})>}\\
&= g_T^{-\tau s_0(\theta_1\hbar+\theta_2)+\tau s_0(\theta_1\hbar+\theta_2)+\boldsymbol{k}_x(\alpha,\boldsymbol{r},\boldsymbol{h})\boldsymbol{E}\boldsymbol{c}_y^{\top}(\boldsymbol{s},\boldsymbol{h}_{\mathsf{M}})} && (\text{since } c_{y,1}(\boldsymbol{s},\boldsymbol{h}) = s_0)\\
&= g_T^{\alpha s_0} && (\text{by correctness of } \mathsf{P})
\end{aligned}$$

Remark 3.3. In Sign algorithm, two random coins, τ and $\boldsymbol{v}_{\mathsf{sp}}$ are used, among them $\boldsymbol{v}_{\mathsf{sp}}$ is assigned only for signer privacy and τ is the only coin that provides randomness in unforgeability. If signer privacy is not required, we can ignore $\boldsymbol{v}_{\mathsf{sp}}$.

Fact 3.4. We note that size of the signature for a message (m, y) is $\omega_1 + 1$, where $|\boldsymbol{c}_y| = \omega_1$ and number of pairings in Ver is $\omega_1 + 1$. **Therefore, if $\boldsymbol{c}_y$ of the underlying pair encoding scheme is of constant-size, then the corresponding signature will be constant-size and the number of pairings in verification will be constant-size**. One example of such pair encodings is Pair Encoding Scheme 5 of [2].

3.4 How to Uniformly Sample from $(\mathbf{V}_{\mathsf{M}})^{\perp}$

Let $\mathbf{V} := \{\boldsymbol{c}_y(\boldsymbol{s}, \boldsymbol{h}) \in \mathbb{Z}_N^{\omega_1} \mid \boldsymbol{s} := (s_0, \ldots, s_{\omega_2}) \in \mathbb{Z}_N^{\omega_2+1}\}$ and $\mathbf{V}^{\perp} := \{\boldsymbol{v} \in \mathbb{Z}_N^{\omega_1} \mid\ <\boldsymbol{v}, \boldsymbol{u}>= 0\ \forall\ \boldsymbol{u} \in \mathbf{V}\}$. Note that there is no known method to sample uniformly from $\mathbf{V}^{\perp}$ for arbitrary pair encoding schemes. However, it is possible if we put a condition on $\mathsf{Enc2}$ of P. Condition (2) of **Conditions 3.1** is such a condition. Let $\boldsymbol{s} = (s_0, \ldots, s_{\omega_2})$ and $\boldsymbol{h} = (h_1, \ldots, h_n)$. Write $\boldsymbol{c}_y(\boldsymbol{s}, \boldsymbol{h}) = \boldsymbol{c}(\boldsymbol{s}, \boldsymbol{h}) = (c_1(\boldsymbol{s}, \boldsymbol{h}), \ldots, c_{\omega_1}(\boldsymbol{s}, \boldsymbol{h}))$, where $c_\iota(\boldsymbol{s}, \boldsymbol{h})$ is given by

$$c_\iota(\boldsymbol{s}, \boldsymbol{h}) := \sum_{j\in[0,\omega_2]} a_{\iota,j} s_j + \sum_{\substack{j\in[0,\omega_2]\\ i\in[n]}} a_{\iota,j,i} h_i s_j.$$

Then, $\boldsymbol{c}_y^{\top}(\boldsymbol{s}, \boldsymbol{h})$ can be written as $\boldsymbol{c}_y^{\top}(\boldsymbol{s}, \boldsymbol{h}) = \boldsymbol{A}\boldsymbol{s}^{\top}$, where the matrix $\boldsymbol{A} \in \mathbb{Z}_N^{\omega_1\times(\omega_2+1)}$ is given by:

$$\boldsymbol{A} := \left(a_{\iota,j} + \sum_{i\in[n]} a_{\iota,j,i} h_i\right)_{\substack{1\le\iota\le\omega_1\\ 0\le j\le\omega_2}}$$

For simplicity of the description, we assign the label for the columns of $\boldsymbol{A}$ from 0 to ω_2. The matrix $\boldsymbol{A}$, we call the associated matrix for $\boldsymbol{c}_y(\boldsymbol{s}, \boldsymbol{h})$. The matrix $\boldsymbol{A}$ is described by $a_{\iota,j}$, $a_{\iota,j,i}$ and h_i, where $\iota \in [\omega_1]$, $j \in [0, \omega_2]$ and $i \in [n]$. Note that $a_{\iota,j}$'s and $a_{\iota,j,i}$'s are the coefficients of the polynomials c_ι's with $(\boldsymbol{c}_y, \omega_2) \longleftarrow \mathsf{Enc2}(y, N)$. Therefore, the matrix $\boldsymbol{A}$ is completely determined by $y \in \mathcal{Y}$ and $\boldsymbol{h}$. Since the part, $\boldsymbol{h}$ is fixed, we say that $\boldsymbol{A}$ is associated with $y \in \mathcal{Y}$. Then from the definition of $\mathbf{V}^{\perp}$, we have

$$\begin{aligned}
\mathbf{V}^{\perp} &= \{\boldsymbol{v} \in \mathbb{Z}_N^{\omega_1} \mid\ <\boldsymbol{v}, \boldsymbol{u}>= 0\ \forall\ \boldsymbol{u} \in \mathbf{V}\}\\
&= \{\boldsymbol{v} \in \mathbb{Z}_N^{\omega_1} \mid \boldsymbol{v}\boldsymbol{c}_y^{\top}(\boldsymbol{s}, \boldsymbol{h}) = 0\ \forall\ \boldsymbol{s} \in \mathbb{Z}_N^{\omega_2+1}\}\\
&= \{\boldsymbol{v} \in \mathbb{Z}_N^{\omega_1} \mid \boldsymbol{v}\boldsymbol{A}\boldsymbol{s}^{\top} = 0\ \forall\ \boldsymbol{s} \in \mathbb{Z}_N^{\omega_2+1}\}\\
&= \{\boldsymbol{v} \in \mathbb{Z}_N^{\omega_1} \mid \boldsymbol{v}\boldsymbol{A} = \mathbf{0}\}\\
&= \{\boldsymbol{v} \in \mathbb{Z}_N^{\omega_1} \mid \boldsymbol{A}^{\top}\boldsymbol{v}^{\top} = \mathbf{0}\}.
\end{aligned}$$

Now, sampling from $\mathbf{V}^{\perp}$ boils down to solving the homogeneous system, $\boldsymbol{A}^{\top}\boldsymbol{X} = \mathbf{0}$, with $\boldsymbol{X}^{\top} := (x_1, \ldots, x_{\omega_1})$. **Before proceeding further, we note that the sampling of $\mathbf{V}^{\perp}$ gives rise to the sampling of $(\mathbf{V}_{\mathsf{M}})^{\perp}$ if $c_1(\boldsymbol{s}, \boldsymbol{h}) = s_0$.** It is assured using Theorem 2.3, where $\boldsymbol{A}_{\mathsf{M}}^{\top}$ is defined from $\boldsymbol{A}^{\top}$ and $t := \theta_1\hbar + \theta_2$.

Our goal is to compute $g^{\boldsymbol{v}}$, where $\boldsymbol{v} \xleftarrow{\text{U}} \mathbf{V}^{\perp}$. Note that $g^{\boldsymbol{h}}$ is given but not $\boldsymbol{h}$. If each component v_j of $\boldsymbol{v}$ is linear combination of h_i's, then we will be able to compute $g^{\boldsymbol{v}}$. In fact, for each $\iota \in [\omega_1]$, if $v_\iota = \sum_{i=1}^{n} \chi_{\iota,i} h_i$, where $\chi_{\iota,i} \in \mathbb{Z}_N$ for $i \in [n]$, then g^{v_ι} can be computed as $(g^{h_1})^{\chi_{\iota,1}} \cdots (g^{h_n})^{\chi_{\iota,n}}$.

Since h_i's are not known, we are not able to compute h_i^{-1} required for the elementary operations of type-2 (for details of the elementary operations, refer to Section 2.4). It may even happen that the h_i's are not invertible in $\mathbb{Z}_N$. So the only information of $\boldsymbol{A}$ available in the process of elementary operations are a_ι's, $a'_{\iota,i}$'s, $a_{\iota,j}$'s and $a_{\iota,j,i}$'s. Therefore, throughout the elementary operations, we treat h_i's as symbols, where the symbols h_i^{-1}'s are not known. But, if we find some row of $\boldsymbol{A}^{\top}$ is a multiple of h_i, then we can multiple the row by h_i^{-1} (provided it exists in $\mathbb{Z}_N$) to make the row h_i free. Under these multiplications solution of the system remains unchanged.

Suppose $\boldsymbol{M}$ is obtained by applying say n elementary column operations on $\boldsymbol{A}^{\top}$, then we have $\boldsymbol{A}^{\top}\boldsymbol{\mathcal{E}}_1^{\top}\boldsymbol{\mathcal{E}}_2^{\top} \cdots \boldsymbol{\mathcal{E}}_n^{\top} = \boldsymbol{M}$, where $\boldsymbol{\mathcal{E}}_i$'s are elementary matrices. If the column operations are other than the type-1 operation, then there is a chance that h_i may appear in the elementary matrix $\boldsymbol{\mathcal{E}}_j^{\top}$. Since for each solution $\boldsymbol{v} := (v_1, \ldots, v_{\omega_1})^{\top}$ of $\boldsymbol{M}\boldsymbol{X} = \mathbf{0}$, $\boldsymbol{\mathcal{E}}_1^{\top}\boldsymbol{\mathcal{E}}_2^{\top} \cdots \boldsymbol{\mathcal{E}}_n^{\top}\boldsymbol{v}$ is a solution of $\boldsymbol{A}^{\top}\boldsymbol{X} = \mathbf{0}$ and v_ι's are linear combination of h_i's, the terms like $h_{i_1}h_{i_2} \cdots h_{i_k}$ may appear in $\boldsymbol{v}$ to complicate things. For this reason, we avoid the elementary column operations in the sampling process.

Below, we define the leading h-free column of a matrix which comes in connection with h-free coins (Definition 3.1). The definition says that for each h-free coin s_j, there is a unique leading h-free column of the matrix $\boldsymbol{A}^{\top}$.

Definition 3.2. A ι^{th} column of $\boldsymbol{A}^{\top}$ is said to be "leading h-free" column if there exists a $j \in [0, \omega_2]$ such that all the entries of the ι^{th} column of $\boldsymbol{A}^{\top}$ are 0 except $A^{\top}_{j,\iota} = a_{\iota,j}$.

For Examples 3.6 and 3.7, the leading h-free columns of $\boldsymbol{A}^{\top}$ are $\{1, 4, 6, 8, 10\}$ and $\{1, 3, 5, 7, 9\}$ respectively.

More Notations. We define $S_{\mathsf{hf}} := \{\iota \in [\omega_1] \mid \exists\ j \in [0, \omega_2]$ such that $c_\iota(\boldsymbol{s}, \boldsymbol{h}) = a_{\iota,j}s_j\}$ and $T_{\mathsf{hf}} := \{j \in [0, \omega_2] \mid \exists \iota \in [\omega_1]$ such that $c_\iota(\boldsymbol{s}, \boldsymbol{h}) = a_{\iota,j}s_j\}$. We remark that S_{hf} and T_{hf} are respectively the collection of indices for h-free columns and h-free coins. Let $S_{\mathsf{non-hf}} := [\omega_1] \setminus S_{\mathsf{hf}}$ and $T_{\mathsf{non-hf}} := [0, \omega_2] \setminus T_{\mathsf{hf}}$. The main task is to find which variables are free and which are not among $x_1, \ldots, x_{\omega_1}$ with $\boldsymbol{X} := (x_1, \ldots, x_{\omega_1})^{\top}$ for the homogeneous system, $\boldsymbol{A}^{\top}\boldsymbol{X} = \mathbf{0}$. Let S_{fv} and $S_{\mathsf{non-fv}}$ respectively represent the indices for free variables and non-free variables.

Remark 3.5. Since, the factorization problem is assumed to be intractable, all $a_{\iota,j}$'s appearing in condition 2(a) are invertible in $\mathbb{Z}_N$ (as discussed in Section 2.4). For most of the existing pair coding schemes, $a_{\iota,j}$'s are found to be 1. When all the variables are h-free, then $T_{\mathsf{non-hf}} = \emptyset$.

Algorithm for sampling. As discussed above that sampling from $\mathbf{V}^{\perp}$ boils down to solving $\boldsymbol{A}^{\top}\boldsymbol{X} = \mathbf{0}$ with $\boldsymbol{X}^{\top} = (x_1, \ldots, x_{\omega_1})$. The matrix $\boldsymbol{A}$ is completely determined by $a_{\iota,j}$, $a_{\iota,j,i}$ and h_i, where $\iota \in [\omega_1]$, $j \in [0, \omega_2]$ and $i \in [n]$. Since h_i's are not known, the input matrix $\boldsymbol{A}$ to the algorithm is supplied by $a_{\iota,j}$, $a_{\iota,j,i}$ and g^{h_i}, where $\iota \in [\omega_1]$, $j \in [0, \omega_2]$ and $i \in [n]$. We call this form of input for the matrix $\boldsymbol{A}$ as implicit form of $\boldsymbol{A}$. The algorithm returns $(g^{x_1}, \ldots, g^{x_{\omega_1}})$, where $(x_1, \ldots, x_{\omega_1})$ is a uniform solution of $\boldsymbol{A}^{\top}\boldsymbol{X} = \mathbf{0}$ which we call implicit form of solution for the system. We describe Algorithm 1 for sampling in details which takes as input the matrix $\boldsymbol{A}$ in implicit form associated with some $y \in \mathcal{Y}$ and outputs a uniform solution in implicit form of the system $\boldsymbol{A}^{\top}\boldsymbol{X} = \mathbf{0}$. Algorithm 1 separately handles two cases, all s_j's are

h-free and not all s_j's are h-free. The additional comments for the statements of Algorithm 1 are described in details below.

- **(All s_j's are h-free.)** From line no.2 to 11 represents the case that all s_j's involved in $\boldsymbol{c}_y(\boldsymbol{s},\boldsymbol{h})$ are h-free. For this case, we do not require any elementary operation. In this case, $\mathsf{Null}(\boldsymbol{A}^\top) = \omega_1 - (\omega_2+1)$. For better understanding this case, we refer to Example 3.6.
  - From line no.7 to 9: For each $\iota \in S_{\mathsf{hf}}$, there is a unique $j \in [0,\omega_2]$ such that $c_\iota(\boldsymbol{s},\boldsymbol{h}) = a_{\iota,j}s_j$ by condition 2(a). The condition 2(a) guarantees that no non-free variable contributes during the computation of others.
- **(Not all s_j's are h-free.)** From line no.12 to 36 represents the case that not all s_j's involved in $\boldsymbol{c}_y(\boldsymbol{s},\boldsymbol{h})$ are h-free. In this case, $\mathsf{Null}(\boldsymbol{A}^\top) \leq \omega_1 - (\omega_2 + 1)$. For better understanding, we refer to Example 3.7.
  - For line no.14: For each $j \in T_{\mathsf{non-hf}}$, there is a unique i' such that $a_{\iota,j,i'} \neq 0$ and for all $\iota \in [\omega_1]$, $i \in [n]$ with $i \neq i'$, $a_{\iota,j,i} = 0$, $a_{\iota,j} = 0$ by condition 2(b). On line no.14, the j^{th} row of $\boldsymbol{A}$ is multiplied by $h_{i'}^{-1}$ symbolically to makes each element of the j^{th} row free from h-term. Under these changes the h-free variables remain h-free as the corresponding leading h-free columns are unaffected. Since, $h_{i'}$ is invertible (by condition 2(b)), the solutions of the system $\boldsymbol{A}^\top\boldsymbol{X} = \boldsymbol{0}$ remain unaltered
  - From line no.16 to 25: It applies the elementary row operations of type-2 and type-3 until each row $j \in T_{\mathsf{non-hf}}$ becomes row reduced.
  - From line no.18 to 20: Solves the factorization problem in polynomial time in κ and aborts. In this case, $\gcd(k, N)$ is a factor of N.
  - For line no.23: It applies the elementary row operations of type-3 to reduce all other elements of the column containing the leading 1 to 0.
  - For line no.25: Under the elementary row operations of type-2 and type-3 used in line no.22 and 23, the h-free variables remain h-free as the corresponding leading h-free columns are unaffected, but some non-h-free variables become h-free. These new h-free variables make the free variables to non-free variables.
  - For line no.26: $S_{\mathsf{non-fv}}$ is the set of new non-free variables.
  - From line no.32 to 34: Note that the set of non-free variables to the system, $\boldsymbol{MX} = \boldsymbol{0}$ is $S_{\mathsf{non-fv}} := S_{\mathsf{hf}} \cup S_{\mathsf{new}}$. As in first case, for each $\iota \in S_{\mathsf{hf}}$, there is a unique $j \in [0,\omega_2]$ such that $c_\iota(\boldsymbol{s},\boldsymbol{h}) = a_{\iota,j}s_j$ by condition 2(a). For each $\iota \in S_{\mathsf{new}}$, there is a unique $j \in [0,\omega_2]$ such that $c_\iota(\boldsymbol{s},\boldsymbol{h}) = a_{\iota,j}s_j$ by line no.16 to 25.

Example 3.6. For better understanding of Algorithm 1, we work out Pair Encoding Scheme 4 (given in Section 3.1). We customize a set of attributes to be $S := \{y_2, y_3, y_4\} \subset \mathbb{Z}_N$. $\mathsf{Enc2}(S) \to \boldsymbol{c}(\boldsymbol{s},\boldsymbol{h}) = (c_1 := s, c_2 := s\eta, c_3 := s\phi_1 + w\phi_2, c_4 := w, \{c_{5,y}, c_{6,y}\}_{y\in S})$, where $c_{5,y} := w\phi_3 + s_y(h_0 + h_1 y)$, $c_{6,y} := s_y$ and

Algorithm 1: An algorithm for uniform sampling from $\mathbf{V}^{\perp}$.

```
Input: $a_{\iota,j}$, $a_{\iota,j,i}$ and $g^{h_i}$, where $\iota \in [\omega_1]$, $j \in [0, \omega_2]$ and $i \in [n]$ ($\boldsymbol{A}$ in implicit form).
Output: $g^{(x_1,\dots,x_{\omega_1})}$, where $(x_1, \dots, x_{\omega_1})^\top$ is a uniform solution of the system $\boldsymbol{A}^\top \boldsymbol{X} = \mathbf{0}$.
It computes the sets $S_{\mathsf{hf}}$, $S_{\mathsf{non-hf}}$ and $T_{\mathsf{non-hf}}$ ;
if All $s_j$'s are h-free then
    $S_{\mathsf{non-fv}} := S_{\mathsf{hf}}$ and $S_{\mathsf{fv}} := [\omega_1] \setminus S_{\mathsf{non-fv}}$ ;
    for $i \in S_{\mathsf{fv}}$ do
        $x_i := \chi_i \xleftarrow{\mathrm{U}} \mathbb{Z}_N$ ;                      // free variables are assigned uniformly
    end
    for $\iota \in S_{\mathsf{non-fv}}$ do
        $x_\iota := -a_{\iota,j}^{-1} \sum_{i \in S_{\mathsf{fv}}} A_{j,i}^\top \chi_i$ ;      // for each $\iota \in S_{\mathsf{non-fv}}$ $\exists! j \in [0, \omega_2]$
    end
    return $(g^{x_1}, \dots, g^{x_{\omega_1}})$ ;                                 // refer to Example 3.6
end
else
    for $j \in T_{\mathsf{non-hf}}$ do
        $\boldsymbol{A}_j^\top \leftarrow h_{i'}^{-1} \boldsymbol{A}_j^\top$ ;            // $j^{th}$ row of $\boldsymbol{A}^\top$ is multiplied by $h_{i'}^{-1}$
    end
    for $j \in T_{\mathsf{non-hf}}$ do
        $k$ := the first non-zero (leading) element of the $j^{th}$ row ;
        if $\gcd(k, N) > 1$ then
            return $\gcd(k, N)$ ;                          // solves factorization problem for $N$
        end
        else
            $\boldsymbol{A}_j^\top \leftarrow k^{-1} \boldsymbol{A}_j^\top$ ;                     // make the leading element to 1
            all other elements of the column containing the leading 1 are changed to 0 ;
        end
    end
    ;                  // $\boldsymbol{M}$ := matrix obtained by applying above operations on $\boldsymbol{A}^\top$
    $S_{\mathsf{new}} := \{\iota \in S_{\mathsf{non-hf}} \mid \exists j \in T_{\mathsf{non-hf}}$ s.t $M_{i\iota} = \delta_{i,j}\}$ ;
    $S_{\mathsf{non-fv}} := S_{\mathsf{hf}} \cup S_{\mathsf{new}}$ ;                  // set of non-free variables for $\boldsymbol{MX} = \mathbf{0}$
    $S_{\mathsf{fv}} := [\omega_1] \setminus S_{\mathsf{non-fv}}$ ;                      // set of free variables for $\boldsymbol{MX} = \mathbf{0}$
    for $i \in S_{\mathsf{fv}}$ do
        $x_i := \chi_i \xleftarrow{\mathrm{U}} \mathbb{Z}_N$ ;                      // free variables are assigned uniformly
    end
    for $\iota \in S_{\mathsf{non-fv}}$ do
        $x_\iota := -(M_{j,\iota})^{-1} \sum_{i \in S_{\mathsf{fv}}} M_{j,i} \chi_i$;      // for each $\iota \in S_{\mathsf{non-fv}}$ $\exists! j \in [0, \omega_2]$
    end
    return $(g^{x_1}, \dots, g^{x_{\omega_1}})$;                                  // refer to Example 3.7
end
```

$\boldsymbol{s} := (s_0 := s, s_1 := w, s_2, s_3, s_4)$ with $s_i := s_{y_i}$ for $i \geq 2$. The matrix[7] of the system $\boldsymbol{A}^\top \boldsymbol{X} = \mathbf{0}$ is given by:

$$\boldsymbol{A}^\top = \begin{array}{c} \\ s_0 \\ s_1 \\ s_2 \\ s_3 \\ s_4 \end{array} \begin{array}{c} \begin{array}{cccccccccc} c_1 & c_2 & c_3 & c_4 & c_{5,y_2} & c_{6,y_2} & c_{5,y_3} & c_{6,y_3} & c_{5,y_4} & c_{6,y_4} \end{array} \\ \begin{pmatrix} \boxed{1} & \eta & \phi_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \phi_2 & \boxed{1} & \phi_3 & 0 & \phi_3 & 0 & \phi_3 & 0 \\ 0 & 0 & 0 & 0 & h_0 + h_1 y_2 & \boxed{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & h_0 + h_1 y_3 & \boxed{1} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & h_0 + h_1 y_4 & \boxed{1} \end{pmatrix} \end{array}$$

This is a case, where **all the coins are h-free**. Here $\omega_1 = 10$, $\omega_2 = 4$, $S_{\mathsf{hf}} := \{1, 4, 6, 8, 10\}$ and $T_{\mathsf{hf}} := \{0, 1, 2, 3, 4\}$. Therefore, $S_{\mathsf{non-fv}} := S_{\mathsf{hf}} = \{1, 4, 6, 8, 10\}$ and $S_{\mathsf{fv}} := [10] \setminus S_{\mathsf{non-fv}} = \{2, 3, 5, 7, 9\}$. For each $i \in S_{\mathsf{fv}}$, $x_i := \chi_i \xleftarrow{\text{U}} \mathbb{Z}_N$. The non-free variables are computed as: $x_1 := -\eta\chi_2 - \phi_2\chi_3$, $x_4 := -\phi_2\chi_3 - \phi_3(\chi_5 + \chi_7 + \chi_9)$, $x_6 := -(h_0 + h_1 y_2)\chi_5$, $x_8 := -(h_0 + h_1 y_3)\chi_7$, $x_{10} := -(h_0 + h_1 y_4)\chi_9$. Therefore, $(x_1, \ldots, x_{10})^\top$ is a solution of the system $\boldsymbol{A}^\top \boldsymbol{X} = \mathbf{0}$. If $\boldsymbol{v} = (x_1, \ldots, x_{10})$, then $g^{\boldsymbol{v}}$ is computed as follows. $g^{x_1} := (g^\eta)^{-\chi_2}.(g^{\phi_2})^{-\chi_3}$, $g^{x_2} := g^{\chi_2}$, $g^{x_3} := g^{\chi_3}$, $g^{x_4} := (g^{\phi_2})^{-\chi_3}.(g^{\phi_3})^{-(\chi_5+\chi_7+\chi_9)}$, $g^{x_5} := g^{\chi_5}$, $g^{x_6} := (g^{h_0})^{-\chi_5}.(g^{h_1})^{-y_2\chi_5}$, $g^{x_7} := g^{\chi_7}$, $g^{x_8} := (g^{h_0})^{-\chi_7}.(g^{h_1})^{-y_3\chi_7}$, $g^{x_9} := g^{\chi_9}$ and $g^{x_{10}} := (g^{h_0})^{-\chi_9}.(g^{h_1})^{-y_4\chi_9}$

Example 3.7. We also consider Pair Encoding Scheme 10 (described in Section 3.1) which explains other case of Algorithm 1. Let $\Gamma := (\boldsymbol{M}, \rho)$ be a span program, where $\rho : [4] \to \mathcal{U}$ is some row labeling function and $\boldsymbol{M}$ is given below:

$$\boldsymbol{M} = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \\ 3 & 2 & 1 \\ 3 & 1 & 3 \end{pmatrix}$$

If we run $\mathsf{Enc2}$ of Pair Encoding Scheme 10 on Γ, we have following output: $\boldsymbol{c}(\boldsymbol{s}, \boldsymbol{h}) = (c_1, \{c_{2,i}, c_{3,i}\}_{i \in [4]})$, where $c_1 := s$, $c_{2,i} := \phi \boldsymbol{M}_i \boldsymbol{v}^\top + s'_i h_{\rho(i)}$, $c_{3,i} := s'_i$ and $\boldsymbol{s} := (s_0 := s, s_1 := v_2, s_2 := v_3, s_3 := s'_1, s_4 := s'_2, s_5 := s'_3, s_6 := s'_4)$, $\boldsymbol{v} := (s, v_2, v_3)$. The matrix of the system $\boldsymbol{A}^\top \boldsymbol{X} = \mathbf{0}$ is given by:

$$\boldsymbol{A}^\top = \begin{array}{c} \\ s_0 \\ s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \end{array} \begin{array}{c} \begin{array}{ccccccccc} c_1 & c_{2,1} & c_{3,1} & c_{2,2} & c_{3,2} & c_{2,3} & c_{3,3} & c_{2,4} & c_{3,4} \end{array} \\ \begin{pmatrix} \boxed{1} & \phi & 0 & 2\phi & 0 & 3\phi & 0 & 3\phi & 0 \\ 0 & 2\phi & 0 & 3\phi & 0 & 2\phi & 0 & \phi & 0 \\ 0 & 3\phi & 0 & 4\phi & 0 & \phi & 0 & 3\phi & 0 \\ 0 & h_{\rho(1)} & \boxed{1} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & h_{\rho(2)} & \boxed{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & h_{\rho(3)} & \boxed{1} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & h_{\rho(4)} & \boxed{1} \end{pmatrix} \end{array}$$

This is a case, where **all the coins are not h-free**. For all the non-h-free coins (there are only two non-h-free coins, v_2 and v_3), there is a unique h-term which is ϕ. Here $\omega_1 = 9$, $\omega_2 = 6$, $S_{\mathsf{hf}} := \{1, 3, 5, 7, 9\}$, $S_{\mathsf{non-hf}} := [9] \setminus S_{\mathsf{hf}} = \{2, 4, 6, 8\}$, $T_{\mathsf{hf}} := \{0, 3, 4, 5, 6\}$ and $T_{\mathsf{non-hf}} := [0, 6] \setminus T_{\mathsf{hf}} = \{1, 2\}$. Note that the labeling of the rows are started with 0. For each $j \in T_{\mathsf{non-hf}}$, the j^{th} row is multiplied by ϕ^{-1} to make the j^{th} row free from ϕ. We now apply the following elementary row operations of type-2 and type-3 to make each row $j \in T_{\mathsf{non-hf}}$ of $\boldsymbol{A}^\top$ row reduced: $R_2 \leftarrow 2^{-1}R_2$, $R_1 \leftarrow R_1 + (-\phi)R_2$, $R_3 \leftarrow R_3 + (-3)R_2$, $R_4 \leftarrow R_4 + (-h_{\rho(1)})R_2$, $R_3 \leftarrow (-2)R_3$, $R_1 \leftarrow R_1 + (-\phi/2)R_3$, $R_2 \leftarrow R_2 + (-3/2)R_3$, $R_4 \leftarrow R_4 + 3h_{\rho(1)}/2R_3$ and $R_5 \leftarrow R_5 + (-h_{\rho(2)})R_3$. Let $\boldsymbol{M}$ (given below) be the matrix obtained from $\boldsymbol{A}^\top$ after applying the

[7] The box in the j^{th} row indicates that the coin s_j is h-free and the corresponding column containing the box is leading h-free column.

above elementary row operations. The elements appearing in the double boxes of the row reduced rows of $\boldsymbol{M}$ are the new leading elements of the corresponding rows.

$$\boldsymbol{M} = \begin{pmatrix} \boxed{1} & 0 & 0 & 0 & 0 & 0 & 0 & 4\phi & 0 \\ 0 & \boxed{\boxed{1}} & 0 & 0 & 0 & -5 & 0 & 5 & 0 \\ 0 & 0 & 0 & \boxed{\boxed{1}} & 0 & 4 & 0 & -3 & 0 \\ 0 & 0 & \boxed{1} & 0 & 0 & 5h_{\rho(1)} & 0 & -5h_{\rho(1)} & 0 \\ 0 & 0 & 0 & 0 & \boxed{1} & -4h_{\rho(2)} & 0 & 3h_{\rho(2)} & 0 \\ 0 & 0 & 0 & 0 & 0 & h_{\rho(3)} & \boxed{1} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & h_{\rho(4)} & \boxed{1} \end{pmatrix}$$

$S_{\mathsf{new}} := \{2,4\}$. So $S_{\mathsf{non-fv}} := S_{\mathsf{hf}} \cup S_{\mathsf{new}} = \{1,2,3,4,5,7,9\}$ and $S_{\mathsf{fv}} := \{6,8\}$. For each $i \in S_{\mathsf{fv}}$, $x_i := \chi_i \xleftarrow{\mathrm{U}} \mathbb{Z}_N$. The non-free variables are computed as: $x_1 := -4\phi\chi_8$, $x_2 := 5(\chi_6 - \chi_8)$, $x_3 := -5h_{\rho(1)}(\chi_6 - \chi_8)$, $x_4 := -4\chi_6 + 3\chi_8$, $x_5 := h_{\rho(2)}(4\chi_6 - 3\chi_8)$, $x_7 := -h_{\rho(3)}\chi_6$ and $x_9 := -h_{\rho(4)}\chi_8$. Therefore, $(x_1, \dots, x_{10})^\top$ is a solution of the system $\boldsymbol{M}\boldsymbol{X} = \boldsymbol{0}$ and hence, a solution of the system $\boldsymbol{A}^\top \boldsymbol{X} = \boldsymbol{0}$. If $\boldsymbol{v} = (x_1, \dots, x_9)$, then $g^{\boldsymbol{v}}$ is computed as follows. $g^{x_1} := (g^{\phi})^{-4\chi_8}$, $g^{x_2} := g^{5(\chi_6 - \chi_8)}$, $g^{x_3} := (g^{h_{\rho(1)}})^{-5(\chi_6-\chi_8)}$, $g^{x_4} := g^{-4\chi_6+3\chi_8}$, $g^{x_5} := (g^{h_{\rho(2)}})^{4\chi_6 - 3\chi_8}$, $g^{x_6} := g^{\chi_6}$, $g^{x_7} := (g^{h_{\rho(3)}})^{-\chi_6}$, $g^{x_8} := g^{\chi_8}$ and $g^{x_9} := (g^{h_{\rho(4)}})^{-\chi_8}$.

4 Security Proof of Proposed Predicate Signature

4.1 Signer Privacy

Theorem 4.1. *Our proposed PS scheme in Section 3.3 is perfectly private (Definition 2.13).*

Proof. For $\boldsymbol{s} := (s_0, \dots, s_{\omega_2}) \in \mathbb{Z}_N^{\omega_2+1}$, we define $(\mathbf{V}_{\mathsf{M}})_{\alpha s_0} := \{\boldsymbol{v} \in \mathbb{Z}_N^{\omega_1+1} \mid \; < \boldsymbol{v}, \boldsymbol{c}_y^{\mathsf{M}}(\boldsymbol{s}, \boldsymbol{h}_{\mathsf{M}}) >= \alpha s_0\}$. One can easily check that for arbitrary $\tilde{\boldsymbol{v}} \in (\mathbf{V}_{\mathsf{M}})_{\alpha s_0}$, $\tilde{\boldsymbol{v}} + (\mathbf{V}_{\mathsf{M}})^{\perp} = (\mathbf{V}_{\mathsf{M}})_{\alpha s_0}$. Since, the distribution of a signature for (m, y) is

$$\boldsymbol{\delta}_y = g^{\boldsymbol{v}+\boldsymbol{v}_{\mathsf{sp}}} \cdot \boldsymbol{R}_3 \in \mathbb{G}^{\omega_1+1}$$

where $\boldsymbol{v} \in (\mathbf{V}_{\mathsf{M}})_{\alpha s_0}$ for some $\boldsymbol{s} = (s_0, \dots, s_{\omega_2}) \in \mathbb{Z}_N^{\omega_2+1}$. So, it is sufficient to prove that $\boldsymbol{v} + \boldsymbol{v}_{\mathsf{sp}}$ is uniformly distributed over $(\mathbf{V}_{\mathsf{M}})_{\alpha s_0}$ for each $\boldsymbol{s} \in \mathbb{Z}_N^{\omega_2+1}$. Since, $\boldsymbol{v}_{\mathsf{sp}}$ is chosen uniformly and independently from $(\mathbf{V}_{\mathsf{M}})^{\perp}$ and $\boldsymbol{v} + (\mathbf{V}_{\mathsf{M}})^{\perp} = (\mathbf{V}_{\mathsf{M}})_{\alpha s_0}$, so we are done. □

4.2 The Proof of Unforgeability

To prove the unforgeability of the proposed construction in Section 3.3, we apply the signature variant of the dual system methodology [50] deployed in [2]. This signature variant of the dual system is similar to the style of [41, 43]. In this variant, the original unforgeability game is changed to the final game through some intermediate games. These changes are made under three subgroup decision problems and CMH or PMH-security of the underlying pair encoding scheme. In the final game, $\mathcal{V}_{\mathsf{INT}}$ of the verification text is sampled uniformly and independently from $\mathbb{G}_T$. Therefore, the forgery in the final game will be invalid. If ν_1 and ν_2 are respectively the number of key query and signature query made by $\mathscr{A}$, then the reduction cost is $\mathcal{O}(\nu_1 + \nu_2)$. We use the abbreviations 'vText' and 'sf-type' respectively for verification text and semi-functional type. For all the games, we define the semi-functional keys, signatures and verification texts of various type as follow.

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

- $\mathsf{SFKeyGen}(\mathcal{PP}, \mathcal{MSK}, x, g_2, \mathsf{type}, \hat{\boldsymbol{h}})$: It runs $(\boldsymbol{k}_x, m_2) \longleftarrow \mathsf{Enc1}(x, N)$ with $|\boldsymbol{k}_x| = m_1$. It chooses $\hat{\alpha} \xleftarrow{\text{U}} \mathbb{Z}_N$, $\boldsymbol{r}, \hat{\boldsymbol{r}} \xleftarrow{\text{U}} \mathbb{Z}_N^{m_2}$ and $\boldsymbol{R}_3 \xleftarrow{\text{U}} \mathbb{G}_{p_3}^{m_1}$. It outputs the semi-functional key $\mathcal{SK}_x := (x, \boldsymbol{K}_x)$, where $\boldsymbol{K}_x$ is given by:
$$\boldsymbol{K}_x := \begin{cases} g^{\boldsymbol{k}_x(\alpha, \boldsymbol{r}, \boldsymbol{h})} \cdot g_2^{\boldsymbol{k}_x(0, \hat{\boldsymbol{r}}, \hat{\boldsymbol{h}})} \cdot \boldsymbol{R}_3 & \text{if } \mathsf{type}= 1 \\ g^{\boldsymbol{k}_x(\alpha, \boldsymbol{r}, \boldsymbol{h})} \cdot g_2^{\boldsymbol{k}_x(\hat{\alpha}, \hat{\boldsymbol{r}}, \hat{\boldsymbol{h}})} \cdot \boldsymbol{R}_3 & \text{if } \mathsf{type}= 2 \\ g^{\boldsymbol{k}_x(\alpha, \boldsymbol{r}, \boldsymbol{h})} \cdot g_2^{\boldsymbol{k}_x(\hat{\alpha}, \boldsymbol{0}, \boldsymbol{0})} \cdot \boldsymbol{R}_3 & \text{if } \mathsf{type}= 3. \end{cases}$$

- $\mathsf{SFSign}(\mathcal{PP}, m, \mathcal{SK}_x, y, g_2, \mathsf{type})$: If $x \not\sim y$, returns $\perp$. It runs $\boldsymbol{\delta}_y \longleftarrow \mathsf{Sign}(\mathcal{PP},\ m,\ \mathcal{SK}_x,\ y)$. Note that $\boldsymbol{\delta}_y = g^{\boldsymbol{v} + \boldsymbol{v}_{\mathsf{sp}}} \cdot \boldsymbol{R}_3$ with $\boldsymbol{R}_3 \in \mathbb{G}_{p_3}^{\omega_1 + 1}$. It picks $b, \iota \xleftarrow{\text{U}} \mathbb{Z}_N$ and returns the semi-functional signature $\boldsymbol{\delta}_y \cdot g_2^{\hat{\boldsymbol{v}}}$, where $\hat{\boldsymbol{v}} \in \mathbb{Z}_N^{\omega_1 + 1}$ is given by:
$$\hat{\boldsymbol{v}} := \begin{cases} (b, \iota, 0, \ldots, 0) & \text{if } \mathsf{type}= 1 \\ (0, \iota, 0, \ldots, 0) & \text{if } \mathsf{type}= 2. \end{cases}$$

- $\mathsf{SFVText}(\mathcal{PP}, m, y, g_2, \mathsf{type}, \hat{\boldsymbol{h}}_{\mathsf{M}})$: It runs $(\boldsymbol{c}_y, \omega_2) \longleftarrow \mathsf{Enc2}(y, N)$ and picks $\boldsymbol{s} := (s_0, \ldots, s_{\omega_2}), \hat{\boldsymbol{s}} := (\hat{s}_0, \ldots, \hat{s}_{\omega_2}) \xleftarrow{\text{U}} \mathbb{Z}_N^{\omega_2 + 1}$. Computes $\boldsymbol{c}_y^{\mathsf{M}}(\boldsymbol{s}, \boldsymbol{h}_{\mathsf{M}}) := (c_0(s_0, \boldsymbol{\theta}),\ \boldsymbol{c}_y(\boldsymbol{s}, \boldsymbol{h})) \in \mathbb{G}^{\omega_1 + 1}$ and $\boldsymbol{c}_y^{\mathsf{M}}(\hat{\boldsymbol{s}}, \hat{\boldsymbol{h}}_{\mathsf{M}}) := (c_0(\hat{s}_0, \hat{\boldsymbol{\theta}}), \boldsymbol{c}_y(\hat{\boldsymbol{s}}, \hat{\boldsymbol{h}})) \in \mathbb{G}^{\omega_1 + 1}$, where $|\boldsymbol{c}_y| = \omega_1$, $\boldsymbol{\theta} := (\theta_1, \theta_2, \hbar)$, $\hat{\boldsymbol{\theta}} := (\hat{\theta}_1, \hat{\theta}_2, \hbar)$, $\hbar := H(m, y)$, $c_0(s_0, \boldsymbol{\theta}) := s_0(\theta_1 \hbar + \theta_2)$ and $c_0(\hat{s}_0, \hat{\boldsymbol{\theta}}) := \hat{s}_0(\hat{\theta}_1 \hbar + \hat{\theta}_2)$. It returns the semi-function verification text as:
$$\mathcal{V} := \begin{cases} \left(\mathcal{V}_{\mathsf{INT}} := g_T^{\alpha s_0}, \boldsymbol{\mathcal{V}}_y := g^{\boldsymbol{c}_y^{\mathsf{M}}(\boldsymbol{s}, \boldsymbol{h}_{\mathsf{M}})} \cdot g_2^{\boldsymbol{c}_y^{\mathsf{M}}(\hat{\boldsymbol{s}}, \hat{\boldsymbol{h}}_{\mathsf{M}})}\right) & \text{if } \mathsf{type}= 1 \\ \left(\mathcal{V}_{\mathsf{INT}} \xleftarrow{\text{U}} \mathbb{G}_T, \boldsymbol{\mathcal{V}}_y := g^{\boldsymbol{c}_y^{\mathsf{M}}(\boldsymbol{s}, \boldsymbol{h}_{\mathsf{M}})} \cdot g_2^{\boldsymbol{c}_y^{\mathsf{M}}(\hat{\boldsymbol{s}}, \hat{\boldsymbol{h}}_{\mathsf{M}})}\right) & \text{if } \mathsf{type}= 2. \end{cases}$$

Condition 4.1. To go through some of the arguments in the unforgeability proof, we assume the following condition on the pair encodings. This condition is imposed on $\boldsymbol{k}_x$ and $\boldsymbol{E}$ of the pair encodings. We note that this is weaker than the condition (3) of **Conditions 3.1**.

> For $(x, y) \in \mathcal{X} \times \mathcal{Y}$ with $x \sim y$, $(\boldsymbol{k}_x, m_2) \longleftarrow \mathsf{Enc1}(x, N)$ and $\boldsymbol{E} \longleftarrow \mathsf{Pair}(x, y, N)$, it holds that $\boldsymbol{k}_x(\alpha, \boldsymbol{0}, \boldsymbol{0})\boldsymbol{E} = (*, 0, \ldots, 0)$, where $*$ is any entry from $\mathbb{Z}_N$.

Remark 4.2. (Construction of sf-type 2 signature from sf-type 3 key.) We apply this construction (to Lemma A.8) for reaching to Game$_{Final}$ using DSG3 assumption. We see later that the sf-type 3 key is easily computed from the instance of the DSG3 assumption. But, the computation of sf-type 2 signature is not possible unless we assume **Condition 4.1**. Although one can directly compute the sf-type 2 signature from the parameters of DSG3 (without computing sf-type 3 key), for simplicity of the simulation we show the construction of sf-type 2 signature from sf-type 3 key. If we apply the Sign algorithm to the sf-type 3 key, $\mathcal{SK}_x$, we have the following distribution (viz., the $\mathbb{G}_{p_2}$ part):

$$\begin{aligned} \boldsymbol{\delta}_y\big|_{\mathbb{G}_{p_2}} &= (g_2^0, g_2^{\boldsymbol{k}_x(\hat{\alpha}, \boldsymbol{0}, \boldsymbol{0})\boldsymbol{E}}) \\ &= (g_2^0, g_2^{(*, 0, \ldots, 0)}) && \text{(by Condition 4.1)} \\ &= g_2^{(0, *, 0, \ldots, 0)} \end{aligned}$$

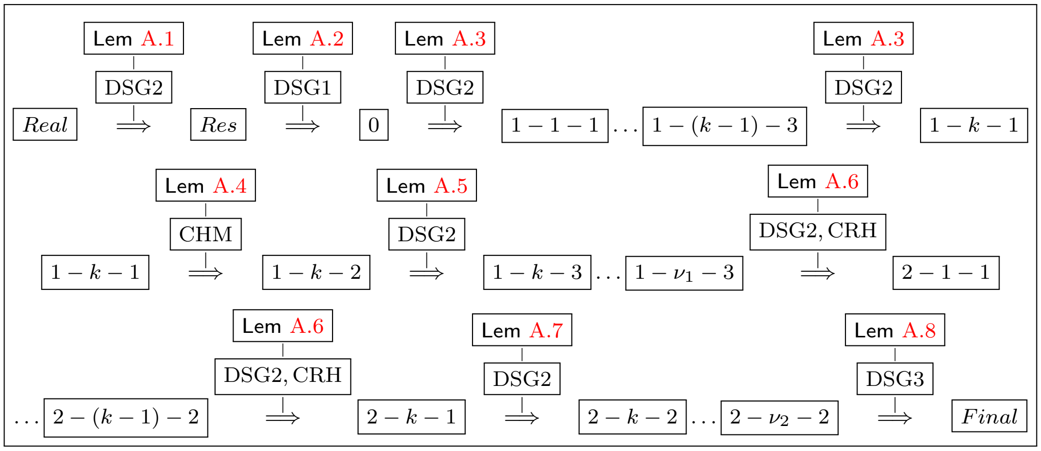
Then, randomize it by composing $g_2^{(0,\iota',0,\ldots,0)} \in \mathbb{G}_{p_2}^{\omega_1+1}$ for $\iota' \xleftarrow{\mathrm{U}} \mathbb{Z}_N$ and finally what we get is the sf-type 2 signature.

Theorem 4.2. *Let* P *be a pair encoding scheme for a predicate* $\sim$ *which satisfies* **Conditions 3.1** *and* $\sim$ *is domain-transferable. Suppose* P *has* CMH*-security, the assumptions, DSG1, DSG2 and DSG3 hold in* $\mathcal{J}$ *and* H *is a collision resistant hash function, then the proposed predicate signature scheme* PS *in Section 3.3 for the predicate* $\sim$ *is adaptive-predicate existential unforgeable (Definition 2.14).*

Proof. Suppose there are at most ν_1 (resp. ν_2) key (resp. signature) queries made by an adversary $\mathscr{A}$. Then the security proof consists of hybrid argument over a sequence of $3\nu_1 + 2\nu_2 + 4$ games which are defined below:

- Game$_{Real}$:= Original AP-UF-CMA game of predicate signature scheme.
- Game$_{Res}$:= This is same as Game$_{Real}$ except $x \not\sim_N y^*$ is replaced by $x \not\sim_{p_2} y^*$ for each key query x made by $\mathscr{A}$.
- Game$_0$ (= Game$_{1-0-3}$) is just like Game$_{Res}$ except that the vText is of sf-type 1.
- In Game$_{1-k-1}$ (for $1 \le k \le \nu_1$) is same as Game$_{1-(k-1)-3}$ except the k^{th} key is sf-type 1.
- Game$_{1-k-2}$ (for $1 \le k \le \nu_1$) is same as Game$_{1-k-1}$ except the k^{th} key is sf-type 2.
- Game$_{1-k-3}$ (for $1 \le k \le \nu_1$) is same as Game$_{1-k-2}$ except the k^{th} key is sf-type 3.
- In Game$_{2-k-1}$ (for $1 \le k \le \nu_2$) is same as Game$_{2-(k-1)-2}$ except the k^{th} signature is of sf-type 1. In this sequel, we define Game$_{2-0-2}$ = Game$_{1-\nu_1-3}$.
- Game$_{2-k-2}$ (for $1 \le k \le \nu_2$) is same as Game$_{2-k-1}$ except the k^{th} signature is of sf-type 2.
- Game$_{Final}$ is similar to Game$_{2-\nu_2-2}$ except that the vText is of sf-type 2.

In Game$_{Final}$, the part, $\mathcal{V}_{\mathsf{INT}}$ is chosen independently and uniformly at random from $\mathbb{G}_T$. This implies that the forgery will be invalid with respect to the vText. Therefore, the adversary $\mathscr{A}$ has no advantage in Game$_{Final}$. The outline of the hybrid arguments over the games is given below, where Lem stands for Lemma.



Using the lemmas referred in the above box (for details of the lemmas, refer to Appendix A), we have the following reduction:

$$\mathsf{Adv}^{\mathrm{AP-UF-CMA}}_{\mathscr{A},\mathrm{PS}}(\kappa) \leq \mathsf{Adv}^{\mathrm{DSG1}}_{\mathscr{B}_1}(\kappa) + (2\nu_1 + 2\nu_2 + 1)\mathsf{Adv}^{\mathrm{DSG2}}_{\mathscr{B}_2}(\kappa) + \nu_1\mathsf{Adv}^{\mathrm{CMH}}_{\mathscr{B}_3,\mathrm{P}}(\kappa) + \nu_2\mathsf{Adv}^{\mathrm{CRH}}_{\mathscr{B}_4}(\kappa) + \mathsf{Adv}^{\mathrm{DSG3}}_{\mathscr{B}_5}(\kappa)$$

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

Theorem 4.3. *Let* P *be a pair encoding scheme for a predicate* $\sim$ *which satisfies* **Conditions 3.1** *and* $\sim$ *is domain-transferable. Suppose* P *has* PMH*-security, the assumptions, DSG1, DSG2 and DSG3 hold in* $\mathcal{J}$ *and* H *is a collision resistant hash function, then the proposed predicate signature scheme* PS *in Section 3.3 for the predicate* $\sim$ *is adaptive-predicate existential unforgeable.*

Proof. Similar to the proof of Theorem 4.2. The reduction of the proof is given by

$$\mathsf{Adv}^{\mathrm{AP-UF-CMA}}_{\mathscr{A},\mathrm{PS}}(\kappa) \leq \mathsf{Adv}^{\mathrm{DSG1}}_{\mathscr{B}_1}(\kappa) + (2\nu_1 + 2\nu_2 + 1)\mathsf{Adv}^{\mathrm{DSG2}}_{\mathscr{B}_2}(\kappa) + \nu_2\mathsf{Adv}^{\mathrm{CRH}}_{\mathscr{B}_3}(\kappa) + \mathsf{Adv}^{\mathrm{DSG3}}_{\mathscr{B}_4}(\kappa)$$

where $\mathscr{B}_1, \mathscr{B}_2, \mathscr{B}_3$ and $\mathscr{B}_4$ are PPT algorithms whose running times are same as that of $\mathscr{A}$. □

5 Instantiations of Predicate Signature

In this section, we instantiate different predicate signature schemes from various pair encoding schemes. The different variants of PS with many new features which did not exist earlier in the literature are presented here. Also we show that some existing PS schemes can be obtained by applying our framework. If the underlying pair encoding scheme with either PMH or CMH-security satisfies the sufficient **Conditions 3.1**, then our construction of predicate signature in Section 3.3 guarantees the signer privacy and adaptive-predicate unforgeability. For instantiations, we consider only the pair encoding schemes[8] presented in [2, 6, 52] as they are having either PMH or CMH-security and satisfy the aforementioned conditions. Other reasons for considering the pair encoding schemes mainly from [2, 6, 52] are that they are available in ready-made forms and many PS schemes with new features can be derived from them. In the following, we briefly describe the instantiations of predicate signature using the pair encodings of [2, 6, 52].

Our framework provides predicate signature scheme for regular languages in key-policy and signature-policy forms. The KP-PS and SP-PS for regular languages are instantiated from Pair Encoding Schemes 3 and 7 of [2] respectively. These are the first practical non-trivial predicate signature schemes beyond ABS.

We can derive an unbounded KP-ABS with large universes from Pair Encoding Scheme 4 of [2]. Here unbounded means there is no restriction on the sizes of policies and attribute sets, and the repetition of attributes in a policy. An ABS with large universes will have super-polynomial size attribute universe. The universe of attributes is considered to be $\mathbb{Z}_N$ and size of the public parameters is constant. The only known adaptive-predicate unforgeable ABS with large universes available in the literature are the construction of [41, 37], among them only ABS of [37] has the feature, unbounded. However, these constructions are known to have signature-policy form. Therefore, the proposed ABS scheme is the first unbounded KP-ABS with

[8] Since, the predicate encodings of [52] have the similar structure with the pair encodings of [2], w.l.o.g we refer the predicate encodings of [52] as pair encodings in the paper. All the pair encoding schemes of [52] are perfectly master-key hiding.

PS	Form	Feature	Pair Encoding	SPES
PS	*KP*	*Regular Languages*	PES 3 [2]	CMH
PS	*SP*	*Regular Languages*	PES 7 [2]	CMH
ABS	*KP*	*Unbounded, Large Universes*	PES 4 [2]	CMH
ABS	*SP*	*Unbounded, Large Universes*	Dual[6] of PES 4 [2]	CMH
ABS	*KP*	*Constant-size signatures*	PES 5 [2]	CMH
ABS	*SP*	*Constant-size keys*	Dual[6] of PES 5 [2]	CMH
KP-DSS	*KP*	*It generalizes KP-ABS*	PES 6 [2]	CMH
SP-DSS	*SP*	*It generalizes SP-ABS*	Dual[6] of PES 6 [2]	CMH
ABS	*KP*	*Cost Free*	PES 8 [2]	PMH
ABS	*SP*	*Cost Free*	PES 10 [2]	PMH
ABS	*KP*	*Cost Free, Large Universes*	PES 12 [2]	PMH
ABS	*SP*	*Cost Free, Large Universes*	PES 13 [2]	PMH
IPS	*NA*	*Cost Free, Const-size signatures*	PES [52]	PMH
IPS	*NA*	*Cost Free, Const-size keys*	PES [52]	PMH
NIPS	*NA*	*Cost Free, Const-size signatures*	PES [52]	PMH
NIPS	*NA*	*Cost Free, Const-size keys*	PES [52]	PMH
SS	*KP*	*Cost Free, Const-size signatures*	PES [52]	PMH
SS	*SP*	*Cost Free, Const-size keys*	Dual [2] of PES [52]	PMH
DSS	*NA*	*Cost Free*	PES 14 [2]	PMH
NSS	*NA*	*Cost Free*	PES 15 [2]	PMH

Table 1: Instantiations of predicate signature using existing pair encodings

large universes which is unforgeable in adaptive-predicate model. We can also instantiate an unbounded SP-ABS with large universes from dual [6] of Pair Encoding Scheme 4 of [2], but it is less efficient than SP-ABS of [37].

We can achieve a KP-ABS with constant-size signatures using Pair Encoding Scheme 5 of [2]. The unforgeability of the only known constant-size signature [4] for non-monotone access structures was proven in the selective-predicate model. Therefore, the proposed ABS scheme is the first ABS with constant-size signature which is existential unforgeable in the adaptive-predicate model. Similarly, by applying our framework on dual [6] of Pair Encoding Scheme 5 of [2], we achieve adaptive-predicate unforgeable SP-ABS with constant-size keys.

The authors [2, 6] proposed new encryption schemes for policy over doubly-spatial relation (see Section 2.9) in key-policy and ciphertext-policy forms. These predicate encryption schemes are called key-policy over doubly-spatial encryption (KP-DSE) and ciphertext-policy over doubly-spatial encryption (CP-DSE) respectively. These predicate encryption schemes work in similar manner as ABE except the equality relation is replaced by doubly-spatial relation [28]. The signature analogue of KP-DSE and CP-DSE are called key-policy over doubly-spatial signature (KP-DSS) and signature-policy over doubly-spatial signature (SP-DSS) respectively. If we apply our framework on Pair Encoding Scheme 6 and its dual, we can obtain KP-DSS and SP-DSS respectively. Similar to KP-DSE (resp. CP-DSE), KP-DSS (resp. SP-DSS) generalizes the existing class, KP-ABS (resp. SP-ABS).

By applying our framework on Pair Encoding Schemes 8 and 9 of [2], we can obtain KP-ABS and SP-ABS with small universes respectively, where a restriction is imposed only on the polices. Since, the underlying pair encodings are perfectly master-key hiding, both the ABS schemes are cost free. The SP-ABS of [43] can be viewed by the proposed SP-ABS.

Attrapadung [2] constructed new cost free ABE schemes with large universes in key-policy and ciphertext-policy forms. The KP-ABE and CP-ABE were constructed from Pair Encoding Schemes 12 and 13 respectively. These pair encoding schemes were constructed based on cover-free families [22, 31]. Analogously, by applying our framework on Pair Encoding Schemes 12 and 13, we obtain cost free KP-ABS and SP-ABS with large universes. Unlike ABS with small universes, bounds on both, the sizes of attribute sets and the sizes of access structures are imposed.

We also instantiate many other cost free predicate signatures as follows. A doubly-spatial signature (DSS) scheme (as a signature analogue of DSE [28]) can be derived using Pair Encoding Scheme 14 of [2]. The signature analogue of negated spatial encryption [5] is called negated spatial signature (NSS). An NSS can be instantiated from Pair Encoding Scheme 15 of [2]. Using the pair encodings of [52] for inner product predicate, we can obtain inner product signature (IPS) schemes with constant-size keys and constant-size signatures respectively. We can also instantiate non-zero inner product signature (NIPS) schemes with constant-size keys and constant-size signatures respectively using the pair encodings of [52] for non-zero inner product predicate. We note that non-zero inner product predicate is a special case of negated spatial predicate. We can also obtain a spatial signature scheme with constant-size signatures using the pair encoding of [52].

A summary of the instantiations of the predicate signature using the pair encodings of [2, 6, 52] is provided in Table 1. The abbreviations NA, KP, SP, PES and SPES stand for not-applicable, key-policy, signature-policy, Pair Encoding Scheme and security of pair encoding scheme respectively. All the pair encodings shown in Table 1 are either perfectly (PMH) secure or computationally (both, SMH and CMH) secure. The right most column stands for the security of the corresponding pair encoding scheme. The security given in Table 1 are used for unforgeability of the predicate signatures. The notations, DSS, KP-DSS, SP-DSS, IPS, NIPS, SS and NSS respectively denote doubly-spatial signature, key-policy over DSS, signature-policy over DSS, inner product signature, non-zero IPS, spatial signature and negated spatial signature.

6 Framework for CCA Secure PE

In the traditional techniques [53, 54, 39] for CCA conversion, the primitive CPA-secure PE schemes must have either verifiability or delegation property. One good side towards this direction is that if the underlying pair encoding scheme fulfills the condition (1) of **Conditions 3.1**, then the fully secure construction in Section 4.3 of [2] always satisfies verifiability (Definition 2.9). In fact, the verifiability is defined as follows.

Verifiability. In the following, we define the algorithm, verify where y is the data-index implicitly contained in C_{cpa}, and x and $\tilde{x}$ are key-indices. Let $\boldsymbol{E} := \mathsf{Pair}(x, y)$ and $\boldsymbol{E}' := \mathsf{Pair}(\tilde{x}, y)$.

$$\mathsf{Verify}(\mathcal{PP}, C_{\mathsf{cpa}}, x, \tilde{x}) := \begin{cases} \perp & \text{if } x \not\sim y \text{ or } \tilde{x} \not\sim y \\ 1 & \text{if } \mathsf{Event} \\ 0 & \text{otherwise} \end{cases}$$

$$\mathsf{Event} := \begin{cases} e(g^{\boldsymbol{k}_x(0,\boldsymbol{1}_i,\boldsymbol{h})\boldsymbol{E}},\ \boldsymbol{C}_y) = 1\ \forall\ i \in [m_2] & (1) \\ e(g^{\boldsymbol{k}_{\tilde{x}}(0,\boldsymbol{1}_i,\boldsymbol{h})\boldsymbol{E}'},\ \boldsymbol{C}_y) = 1\ \forall\ i \in [m_2'] & (2) \\ e(g^{\boldsymbol{k}_x(1,\boldsymbol{0},\boldsymbol{h})\boldsymbol{E}},\ \boldsymbol{C}_y) = e(g^{\boldsymbol{k}_{\tilde{x}}(1,\boldsymbol{0},\boldsymbol{h})\boldsymbol{E}'},\ \boldsymbol{C}_y) = e(g, C_{y,1}) & (3) \\ \text{For } R_3 \in \mathbb{G}_{p_3},\quad e(R_3, C_{y,\iota}) = 1\ \forall\ \iota \in [\omega_1] & (4) \end{cases}$$

where $\boldsymbol{1}_i$ is a vector whose i^{th} position is 1 and rest are 0.

Soundness of verifiability. Suppose $\mathsf{Verify}(\mathcal{PP}, C_{\mathsf{cpa}}, x, \tilde{x}) = 1$, then we show that both the keys, $\mathcal{SK}_x$ and $\mathcal{SK}_{\tilde{x}}$ output the same message on decryption. Let $\Delta_d := \mathsf{Decrypt}(\mathcal{PP}, C_{\mathsf{cpa}}, \mathcal{SK}_x)$.

$$\begin{aligned}
\Delta_d &= C_{\mathsf{INT}}/e(\boldsymbol{K}_x^{\boldsymbol{E}}, \boldsymbol{C}_y) \\
&= C_{\mathsf{INT}}/e(g^{\boldsymbol{k}_x(\alpha,\boldsymbol{r},\boldsymbol{h})\boldsymbol{E}}, \boldsymbol{C}_y) && \text{(by (4))} \\
&= C_{\mathsf{INT}}/e(g^{(\boldsymbol{k}_x(0,\boldsymbol{r},\boldsymbol{h})+\boldsymbol{k}_x(\alpha,\boldsymbol{0},\boldsymbol{h}))\boldsymbol{E}}, \boldsymbol{C}_y) && \text{(by linearity of } \mathsf{P}\text{)} \\
&= C_{\mathsf{INT}}/e(g^{(\sum_{i\in[m_2]} r_i \boldsymbol{k}_x(0,\boldsymbol{1}_i,\boldsymbol{h})+\alpha\boldsymbol{k}_x(1,\boldsymbol{0},\boldsymbol{h}))\boldsymbol{E}}, \boldsymbol{C}_y) && \text{(by linearity of } \mathsf{P}\text{)} \\
&= C_{\mathsf{INT}}/\big(\prod_{i\in[m_2]} e(g^{\boldsymbol{k}_x(0,\boldsymbol{1}_i,\boldsymbol{h})\boldsymbol{E}}, \boldsymbol{C}_y)^{r_i}.e(g^{\boldsymbol{k}_x(1,\boldsymbol{0},\boldsymbol{h})\boldsymbol{E}}, \boldsymbol{C}_y)^{\alpha}\big) \\
&= C_{\mathsf{INT}}/\big(\prod_{i\in[m_2]} 1^{r_i}.e(g, C_{y,1})^{\alpha}\big) && \text{(by (1) and (3))} \\
&= C_{\mathsf{INT}}/e(g, C_{y,1})^{\alpha}.
\end{aligned}$$

Since x is arbitrary, similarly we have $\mathsf{Decrypt}(\mathcal{PP}, C_{\mathsf{cpa}}, \mathcal{SK}_{\tilde{x}}) = C_{\mathsf{INT}}/e(g, C_{y,1})^{\alpha}$.

Completeness of verifiability. It follows from the correctness of the pair encoding scheme P, orthogonality of CBG and the assumption $c_{y,1}(\boldsymbol{s}, \boldsymbol{h}) = s_0$.

Therefore, most of the fully CPA-secure schemes in [2, 6, 52] can be converted to fully CCA-secure schemes using the conversions [53, 54, 39] as the underlying pair encoding schemes satisfy condition (1) of **Conditions 3.1**.

6.1 Direct CCA-secure Predicate Encryption from Pair Encoding Scheme

We explore a direct CCA-secure construction of predicate encryptions from the pair encodings. Using this construction, we achieve CCA security of all the predicate encryptions found in [2, 6, 52] directly from the pair encodings of [2, 6, 52] with almost the same cost of CPA construction of [2]. In fact, the difference between the construction of ours and [2] is that, we use an extra component in ciphertext and three extra pairing computations in decryption.

Terminology: For $(\boldsymbol{c}_y, \omega_2) \longleftarrow \mathsf{Enc2}(y, N)$, we define $\boldsymbol{c}_y^{\mathsf{M}} := (c_0, \boldsymbol{c}_y)$, so $|\boldsymbol{c}_y^{\mathsf{M}}| = \omega_1 + 1$, where $c_0(z, \boldsymbol{\theta}) := z(\theta_1\hbar + \theta_2)$, $\boldsymbol{\theta} := (\theta_1, \theta_2, \hbar) \in \mathbb{Z}_N^3$, z is the independent variable and $|\boldsymbol{c}_y| = \omega_1$.

Let $\mathsf{P} := (\mathsf{Param}, \mathsf{Enc1}, \mathsf{Enc2}, \mathsf{Pair})$ be a primitive pair encoding scheme with the following condition (already defined in **Conditions 3.1**)

Here we assume that for some $\iota \in [\omega_1]$, $c_{y,\iota}(\boldsymbol{s}, \boldsymbol{h}) = s_0$. W.l.o.g, we assume that $c_{y,1}(\boldsymbol{s}, \boldsymbol{h}) = s_0$.

- $\mathsf{Setup}(1^{\kappa}, \boldsymbol{j})$: Same as the Setup in Section 3.3.

- $\mathsf{KeyGen}(\mathcal{PP}, \mathcal{MSK}, x)$: Same as the KeyGen in Section 3.3.

- $\mathsf{Encrypt}(\mathcal{PP}, m, y)$: It runs $(\boldsymbol{c}_y, \omega_2) \longleftarrow \mathsf{Enc2}(y, N)$ and picks $\boldsymbol{s} := (s_0, \ldots, s_{\omega_2}) \xleftarrow{\text{U}} \mathbb{Z}_N^{\omega_2+1}$. Then computes $C_{\mathsf{cpa}} := (y, \boldsymbol{C}_y := g^{\boldsymbol{c}_y(\boldsymbol{s},\boldsymbol{h})}, C_{\mathsf{INT}} := m \cdot g_T^{\alpha s_0})$ and $\hbar := H(C_{\mathsf{cpa}})$. It sets $\boldsymbol{c}_y^{\mathsf{M}}(\boldsymbol{s}, \boldsymbol{h}_{\mathsf{M}}) := (c_0(s_0, \boldsymbol{\theta}), \boldsymbol{c}_y(\boldsymbol{s}, \boldsymbol{h})) \in \mathbb{Z}_N^{\omega_1+1}$, where $|\boldsymbol{c}_y| = \omega_1$, $\boldsymbol{\theta} := (\theta_1, \theta_2, \hbar)$, and $c_0(s_0, \boldsymbol{\theta}) := s_0(\theta_1\hbar + \theta_2)$. Returns $\mathsf{CT} := (y, \boldsymbol{C}_y^{\mathsf{M}} := g^{\boldsymbol{c}_y^{\mathsf{M}}(\boldsymbol{s},\boldsymbol{h}_{\mathsf{M}})}, C_{\mathsf{INT}})$.

- $\mathsf{Decrypt}(\mathcal{PP}, \mathsf{CT}, \mathcal{SK}_x)$: It parses CT as $(y, \boldsymbol{C}_y^{\mathsf{M}} = (C_0, \boldsymbol{C}_y), C_{\mathsf{INT}})$ with $C_0 = g^{c_0(s_0, \boldsymbol{\theta})}$ and $\boldsymbol{C}_y = g^{\boldsymbol{c}_y(\boldsymbol{s}, \boldsymbol{h})}$. Then sets $C_{\mathsf{cpa}} := (y, \boldsymbol{C}_y, C_{\mathsf{INT}})$ and computes $\hbar := H(C_{\mathsf{cpa}})$. It chooses $R \xleftarrow{\mathrm{U}} \mathbb{G}_{p_3}$. If $x \not\sim y$ or $e(g \cdot R, C_0) \neq e(g^{\theta_1 \hbar + \theta_2}, C_1)$, returns $\perp$. Otherwise, it sets $\mathcal{SK}_x^{\mathsf{M}} := (K_0, \boldsymbol{\Psi} \cdot \boldsymbol{K}_x^{\boldsymbol{E}}) \in \mathbb{G}^{\omega_1 + 1}$, where $K_0 := g^{-\tau} \cdot R_0$, $\boldsymbol{\Psi} := g^{\boldsymbol{\psi}}$ with $\boldsymbol{\psi} := (\tau(\theta_1 \hbar + \theta_2), 0, \ldots, 0) \in \mathbb{Z}_N^{\omega_1}$, $\tau \xleftarrow{\mathrm{U}} \mathbb{Z}_N$, $R_0 \xleftarrow{\mathrm{U}} \mathbb{G}_{p_3}$ and $\boldsymbol{E} \leftarrow \mathsf{Pair}(x, y)$. It returns $C_{\mathsf{INT}} / e(\mathcal{SK}_x^{\mathsf{M}}, \boldsymbol{C}_y^{\mathsf{M}})$.

Correctness: Let $\Delta := e(\mathcal{SK}_x^{\mathsf{M}}, \boldsymbol{C}_y^{\mathsf{M}})$. For $x \sim_N y$ ($\Rightarrow x \sim_{p_1} y$ by domain-transferability), we have

$$\begin{aligned}
\Delta &= g_T^{<(-\tau,\ \boldsymbol{\psi} + \boldsymbol{k}_x(\alpha, \boldsymbol{r}, \boldsymbol{h})\boldsymbol{E}),\ \boldsymbol{c}_y^{\mathsf{M}}(\boldsymbol{s}, \boldsymbol{h}_{\mathsf{M}})>} && \text{(by orthogonality of CBG)}\\
&= g_T^{<(-\tau,\ 0,\ldots,0) + (0,\ \boldsymbol{\psi}) + (0,\ \boldsymbol{k}_x(\alpha, \boldsymbol{r}, \boldsymbol{h})\boldsymbol{E}),\ \boldsymbol{c}_y^{\mathsf{M}}(\boldsymbol{s}, \boldsymbol{h}_{\mathsf{M}})>} && \text{(by linearity)}\\
&= g_T^{-\tau c_0(s, \boldsymbol{\theta}) + \tau(\theta_1 \hbar + \theta_2) c_{y,1}(\boldsymbol{s}, \boldsymbol{h}) + <\boldsymbol{k}_x(\alpha, \boldsymbol{r}, \boldsymbol{h})\boldsymbol{E},\ \boldsymbol{c}_y(\boldsymbol{s}, \boldsymbol{h}_{\mathsf{M}})>} && \\
&= g_T^{-\tau s_0(\theta_1 \hbar + \theta_2) + \tau s_0(\theta_1 \hbar + \theta_2) + \boldsymbol{k}_x(\alpha, \boldsymbol{r}, \boldsymbol{h})\boldsymbol{E}\boldsymbol{c}_y^{\top}(\boldsymbol{s}, \boldsymbol{h}_{\mathsf{M}})} && \text{(by assumption: } c_{y,1}(\boldsymbol{s}, \boldsymbol{h}) = s_0)\\
&= g_T^{\alpha s}. && \text{(by correctness of } \mathsf{P})
\end{aligned}$$

Remark 6.1. Note that the CCA secure ciphertext is also represented as $\mathsf{CT} = (C_{\mathsf{cpa}}, C_0)$, where the algorithm $\mathsf{Encrypt}$ can be thought as algorithm $\mathsf{Encrypt}^*$ (the $\mathsf{Encrypt}$ of CPA construction in [2]), then followed by the computation of C_0.

Remark 6.2. The key $\mathcal{SK}_x^{\mathsf{M}}$ defined in $\mathsf{Decrypt}$, we call the *alternative key* (in short alt-key) whose distribution is exactly the same as the signature $\boldsymbol{\delta}_y$, if we ignore the random coin $\boldsymbol{v}_{\mathsf{sp}}$ used for signer privacy. Using this alternative key if we run $\mathsf{AltDecrypt}$ (defined later), we have the same message as in $\mathsf{Decrypt}$ using the original key $\mathcal{SK}_x$.

Fact 6.3. We note that size of the ciphertext is $\omega_1 + 2$, where $|\boldsymbol{c}_y| = \omega_1$ and number pairings in $\mathsf{Decrypt}$ is $\omega_1 + 1$. **Therefore, if $\boldsymbol{c}_y$ of the underlying pair encoding scheme is of constant-size, then the corresponding ciphertext will be constant-size and the number of pairings in decryption will be constant-size**.

6.2 Security Proof of Proposed Predicate Encryption

The proof strategy is based on the dual system style of [50, 2]. Again to pass the argument in Lemma B.11, we assume the **Condition 4.1** (which is implied by condition (3) of **Conditions 3.1**). In the following, we define the algorithms either to generate the semi-functional objects or answer the oracle queries.

- $\mathsf{SFSetup}(1^\kappa, \boldsymbol{j})$: Same as the $\mathsf{SFSetup}$ in Section 4.2.

- $\mathsf{SFKeyGen}(\mathcal{PP}, \mathcal{MSK}, x, g_2, \mathsf{type}, \hat{\boldsymbol{h}})$: Same as the $\mathsf{SFKeyGen}$ in Section 4.2.

- $\mathsf{SFEncrypt}(\mathcal{PP}, m, y, g_2, \mathsf{type}, \hat{\boldsymbol{h}}_{\mathsf{M}})$: It runs $(\boldsymbol{c}_y, \omega_2) \longleftarrow \mathsf{Enc2}(y, N)$ and picks $\boldsymbol{s} := (s_0, \ldots, s_{\omega_2}), \hat{\boldsymbol{s}} := (\hat{s}_0, \ldots, \hat{s}_{\omega_2}) \xleftarrow{\mathrm{U}} \mathbb{Z}_N^{\omega_2 + 1}$. Computes $\boldsymbol{c}_y^{\mathsf{M}}(\boldsymbol{s}, \boldsymbol{h}_{\mathsf{M}}) := (c_0(s_0, \boldsymbol{\theta}),\ \boldsymbol{c}_y(\boldsymbol{s}, \boldsymbol{h})) \in \mathbb{G}^{\omega_1 + 1}$ and $\boldsymbol{c}_y^{\mathsf{M}}(\hat{\boldsymbol{s}}, \hat{\boldsymbol{h}}_{\mathsf{M}}) := (c_0(\hat{s}_0, \hat{\boldsymbol{\theta}}), \boldsymbol{c}_y(\hat{\boldsymbol{s}}, \hat{\boldsymbol{h}})) \in \mathbb{G}^{\omega_1 + 1}$, where $|\boldsymbol{c}_y| = \omega_1$, $\boldsymbol{\theta} := (\theta_1, \theta_2, \hbar)$, $\hat{\boldsymbol{\theta}} := (\hat{\theta}_1, \hat{\theta}_2, \hbar)$, $\hbar := H(C_{\mathsf{cpa}})$, $C_{\mathsf{cpa}} := (y, \boldsymbol{C}_y := g^{\boldsymbol{c}_y(\boldsymbol{s}, \boldsymbol{h})}, C_{\mathsf{INT}} := m \cdot g_T^{\alpha s_0})$, $c_0(s_0, \boldsymbol{\theta}) := s_0(\theta_1 \hbar + \theta_2)$ and $c_0(\hat{s}_0, \hat{\boldsymbol{\theta}}) := \hat{s}_0(\hat{\theta_1} \hbar + \hat{\theta_2})$. It picks $g_t \xleftarrow{\mathrm{U}} \mathbb{G}_T$ and returns the following semi-functional ciphertext CT:

$$\mathsf{CT} := \begin{cases} \left(y, \boldsymbol{C}_y^{\mathsf{M}} := g^{\boldsymbol{c}_y^{\mathsf{M}}(\boldsymbol{s}, \boldsymbol{h}_{\mathsf{M}})} \cdot g_2^{\boldsymbol{c}_y^{\mathsf{M}}(\hat{\boldsymbol{s}}, \hat{\boldsymbol{h}}_{\mathsf{M}})}, C_{\mathsf{INT}} := m \cdot g_T^{\alpha s_0}\right) & \text{if } \mathsf{type} = 1 \\ \left(y, \boldsymbol{C}_y^{\mathsf{M}} := g^{\boldsymbol{c}_y^{\mathsf{M}}(\boldsymbol{s}, \boldsymbol{h}_{\mathsf{M}})} \cdot g_2^{\boldsymbol{c}_y^{\mathsf{M}}(\hat{\boldsymbol{s}}, \hat{\boldsymbol{h}}_{\mathsf{M}})}, C_{\mathsf{INT}} := m \cdot g_t\right) & \text{if } \mathsf{type} = 2. \end{cases}$$

- $\mathsf{SFAltKeyGen}(\mathcal{PP}, \mathcal{MSK}, \mathsf{CT}, x, g_2, \mathsf{type})$: It parses CT as (C_{cpa}, C_0), computes $\hbar := H(C_{\mathsf{cpa}})$ and picks $\tau \xleftarrow{\mathrm{U}} \mathbb{Z}_N$, $R_0 \xleftarrow{\mathrm{U}} \mathbb{G}_{p_3}$. It first generates the normal key, $\mathcal{SK}_x := [x,\ \boldsymbol{K}_x := g^{\boldsymbol{k}_x(\alpha, \boldsymbol{r}, \boldsymbol{h})} \cdot \boldsymbol{R}_3]$. Then, it creates the alt-key $\mathcal{SK}_x^{\mathsf{M}} := (K_0, \boldsymbol{\Psi} \cdot \boldsymbol{K}_x^{\boldsymbol{E}}) \in \mathbb{G}^{\omega_1+1}$, where $K_0 := g^{-\tau} \cdot R_0$, $\boldsymbol{\Psi} := g^{\boldsymbol{\psi}}$ with $\boldsymbol{\psi} := (\tau(\theta_1\hbar + \theta_2), 0, \ldots, 0) \in \mathbb{Z}_N^{\omega_1}$ and $\boldsymbol{E} \leftarrow \mathsf{Pair}(x, y)$. It picks $b, \iota \xleftarrow{\mathrm{U}} \mathbb{Z}_N$ and returns the semi-functional alt-key $\mathcal{SK}_x^{\mathsf{M}} \cdot g_2^{\hat{\boldsymbol{v}}}$, where $\hat{\boldsymbol{v}} \in \mathbb{Z}_N^{\omega_1+1}$ is given by:

$$\hat{\boldsymbol{v}} := \begin{cases} (b, \iota, 0, \ldots, 0) & \text{if } \mathsf{type} = 1 \\ (0, \iota, 0, \ldots, 0) & \text{if } \mathsf{type} = 2. \end{cases}$$

- $\mathsf{AltDecrypt}(\mathcal{PP}, \mathsf{CT}, \mathcal{SK}_x^{\mathsf{M}})$: This is same as $\mathsf{Decrypt}$ algorithm, but here we do not need to compute the alt-key as it is supplied. For sake of completeness: It picks $R \xleftarrow{\mathrm{U}} \mathbb{G}_{p_3}$. If $x \not\sim y$ or $e(g \cdot R, C_0) \neq e(g^{\theta_1\hbar+\theta_2}, C_1)$, it returns $\perp$ else $C_{\mathsf{INT}}/e(\mathcal{SK}_x^{\mathsf{M}}, \boldsymbol{C}_y^{\mathsf{M}})$.

Remark 6.4. If we identify the challenge ciphertext and alt-keys respectively with the verification text and queried signatures in the unforgeability proof of the predicate signature scheme in Section 3.3, then most of the part of CCA-security proof of the proposed predicate encryption scheme in Section 6.1 will follow the unforgeability proof in Section 4.2.

Theorem 6.1. *Let* P *be a pair encoding scheme for a predicate* $\sim$ *which satisfies the conditions (1) and (3) of* **Conditions 3.1** *and* $\sim$ *is domain-transferable. Suppose* P *has both the security,* SMH *and* CMH*, the assumptions, DSG1, DSG2 and DSG3 hold in* $\mathcal{J}$ *and* H *is a collision resistant hash function, then the proposed predicate encryption scheme* PE *in Section 6.1 for the predicate* $\sim$ *is AP-IND-CCA secure (Definition 2.10).*

Proof. Suppose there are at most q (resp. ν) key (resp. decrypt) queries made by an adversary $\mathscr{A}$. 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 and q_2 respectively be the number of pre-challenged and post-challenged key queries. The games are defined below:

- Game_{Real} := Original AP-IND-CCA game of predicate encryption scheme.

- Game_{Res} := This is same as Game_{Real} except $x \not\sim_N y^*$ is replaced by $x \not\sim_{p_2} y^*$ for each key query x made by $\mathscr{A}$.

- Game_0 ($= \mathrm{Game}_{1-0-3}$) is just like Game_{Res} except that the challenge ciphertext is of sf-type 1.

- In Game_{1-k-1} (for $1 \leq k \leq q_1$) same as $\mathrm{Game}_{1-(k-1)-3}$ except the k^{th} queried key is sf-type 1.

- Game_{1-k-2} (for $1 \leq k \leq q_1$) is same as Game_{1-k-1} except the k^{th} queried key is sf-type 2.

- Game_{1-k-3} (for $1 \leq k \leq q_1$) is same as Game_{1-k-2} except the k^{th} queried key is sf-type 3.

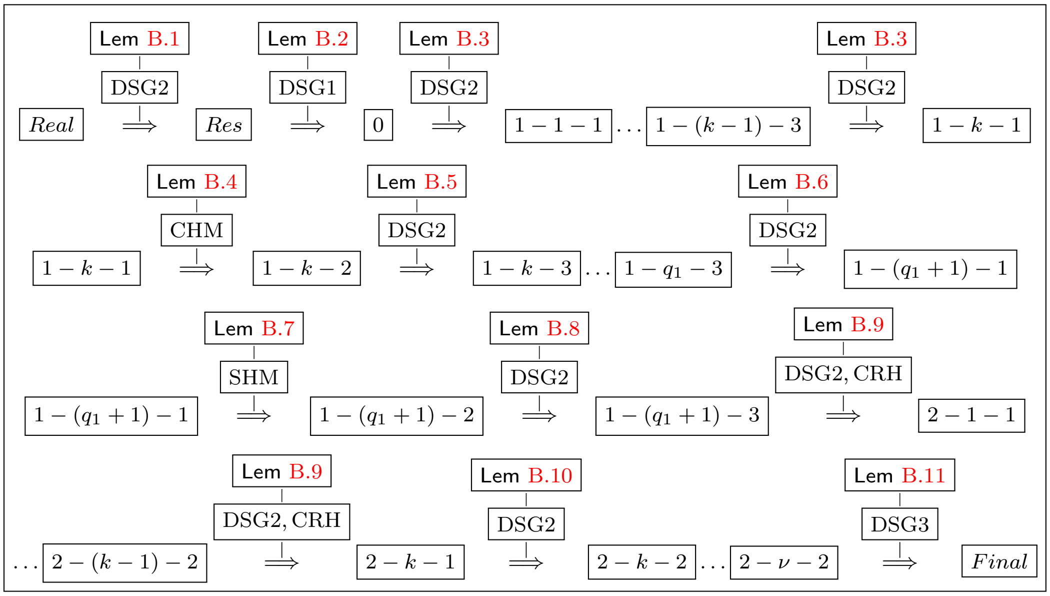
- $\mathrm{Game}_{1-(q_1+1)-i}$ (for $1 \leq i \leq 3$) is same as Game_{1-q_1-3} except the last q_2 queried keys are of sf-type i. Similar to [2], we restrict the form of queried keys of type-2 and type-3 after the challenge phase. That is, for all the post key queries x_j ($q_1 + 1 \leq j \leq q = q_1 + q_2$), $\hat{\alpha}$'s appearing in $\mathbb{G}_{p_2}$ components of $\mathcal{SK}_{x_j}$ are identical.

- In Game_{2-k-1} (for $1 \leq k \leq \nu$) is same as $\mathrm{Game}_{2-(k-1)-2}$ except the k^{th} decrypt query is answered by sf-type 1 alt-key. In this sequel, we define $\mathrm{Game}_{2-0-2} = \mathrm{Game}_{1-(q_1+1)-3}$.

- Game$_{2-k-2}$ (for $1 \leq k \leq \nu$) is same as Game$_{2-k-1}$ except the k^{th} decrypt query is answered by sf-type 2 alt-key.

- Game$_{Final}$ is similar to Game$_{2-\nu-2}$ except that the challenge ciphertext is of sf-type 2.

In Game$_{Final}$, the challenge message m_b is masked with an independently and uniformly chosen element from $\mathbb{G}_T$ implying the component C_{INT} does not leak any information about the challenge message m_b. Therefore, the adversary $\mathscr{A}$ has no advantage in Game$_{Final}$. The outline of the hybrid arguments over the games is given below, where Lem stands for Lemma.



Using the lemmas referred in the above box (for details of the lemmas, refer to Appendix B), we have the following reduction:

$$\begin{aligned}\mathsf{Adv}^{\mathrm{AP-IND-CCA}}_{\mathscr{A},\mathrm{PE}}(\kappa) \leq \mathsf{Adv}^{\mathrm{DSG1}}_{\mathscr{B}_1}(\kappa) + (2q_1 + 2\nu + 3)\mathsf{Adv}^{\mathrm{DSG2}}_{\mathscr{B}_2}(\kappa) + q_1\mathsf{Adv}^{\mathrm{CMH}}_{\mathscr{B}_3,\mathrm{P}}(\kappa) + \\ \mathsf{Adv}^{\mathrm{SMH}}_{\mathscr{B}_4,\mathrm{P}}(\kappa) + \nu\mathsf{Adv}^{\mathrm{CRH}}_{\mathscr{B}_5}(\kappa) + \mathsf{Adv}^{\mathrm{DSG3}}_{\mathscr{B}_6}(\kappa)\end{aligned}$$

where $\mathscr{B}_1, \mathscr{B}_2, \mathscr{B}_3, \mathscr{B}_4, \mathscr{B}_5$ and $\mathscr{B}_6$ are PPT algorithms whose running times are same as that of $\mathscr{A}$. This completes the proof. □

Theorem 6.2. *Let* P *be a pair encoding scheme for a predicate* $\sim$ *which satisfies the conditions (1) and (3) of* **Conditions 3.1** *and* $\sim$ *is domain-transferable. Suppose* P *has* PMH *security, the assumptions, DSG1, DSG2 and DSG3 hold in* $\mathcal{J}$ *and* H *is a collision resistant hash function, then the proposed predicate encryption scheme* PE *in Section 6.1 for the predicate* $\sim$ *is AP-IND-CCA secure (Definition 2.10).*

Proof. Similar to the proof of Theorem 6.1. The reduction of the proof is given by

$$\begin{aligned}\mathsf{Adv}^{\mathrm{AP-IND-CCA}}_{\mathscr{A},\mathrm{PE}}(\kappa) \leq \mathsf{Adv}^{\mathrm{DSG1}}_{\mathscr{B}_1}(\kappa) + (2q + 2\nu + 1)\mathsf{Adv}^{\mathrm{DSG2}}_{\mathscr{B}_2}(\kappa) + \\ \nu\mathsf{Adv}^{\mathrm{CRH}}_{\mathscr{B}_3}(\kappa) + \mathsf{Adv}^{\mathrm{DSG3}}_{\mathscr{B}_4}(\kappa)\end{aligned}$$

where q and ν respectively be the number of key and decrypt queries made $\mathscr{A}$ and $\mathscr{B}_1, \mathscr{B}_2, \mathscr{B}_3, \mathscr{B}_4$ are PPT algorithms whose running times are same as that of $\mathscr{A}$. □

7 Framework for Predicate Signcryption

In this section, we present a construction of predicate signcryption using the pair encodings as black-box. The proposed signcryption supports the combined-setup, i.e., the distributions of public parameters and keys of the implicit primitive schemes, PS and PE are identical. Here we consider signcryption in $\mathcal{C}t\mathcal{E}\&\mathcal{S}t\mathcal{S}$-paradigm of [43] to guarantee the faster execution of the algorithms in $\mathsf{Signcrypt}$ and $\mathsf{Unsigncrypt}$, stronger security and public verifiability.

Let $\mathsf{PE} := (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{PE.Encrypt}, \mathsf{PE.Decrypt})$ and $\mathsf{PS} := (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{PS.Sign}, \mathsf{PS.Ver})$ be respectively the predicate encryption scheme in Section 6.1 and predicate signature scheme in Section 3.3 constructed from pair encoding scheme. Let $\mathsf{OTS} := (\mathsf{OTS.Gen}, \mathsf{OTS.Sign}, \mathsf{OTS.Ver})$ and $\mathcal{C} := (\mathsf{CSetup}, \mathsf{Commit}, \mathsf{Open})$ be the one-time signature scheme (for details of OTS, refer to Appendix 2.7) and commitment scheme respectively. To distinguish the hash values $\hbar_s$ and $\hbar_e$ involved in PS and PE, we keep the first argument of the hash function H to be 1 and 0 respectively, viz., $\hbar_s := H(1, \mathsf{vk}, y_s)$ and $\hbar_e := H(0, \mathsf{com}, \boldsymbol{\delta}_{y_s}, \mathsf{vk}, C_{\mathsf{cpa}})$

- $\mathsf{Setup}(1^\kappa, \boldsymbol{j})$: Same as the Setup in Section 3.3 except the $\mathcal{PP}$ additionally contains the public commitment key, $\mathcal{CK}$.

- $\mathsf{KeyGen}(\mathcal{PP}, \mathcal{MSK}, x)$: Same as the KeyGen in Section 3.3.

- $\mathsf{Signcrypt}(m, \mathcal{SK}_x, y_s, y_e) := \left(\begin{array}{c} \boxed{(\mathsf{com}, \mathsf{decom}) \longleftarrow \mathsf{Commit}(m); \parallel (\mathsf{vk}, \mathsf{signk}) \longleftarrow \mathsf{OTS.Gen}(1^\kappa);} \\ \boxed{\boldsymbol{\delta}_{y_s} \longleftarrow \mathsf{PS.Sign}(\mathsf{vk}, \mathcal{SK}_x, y_s); \parallel C_{\mathsf{cpa}} \longleftarrow \mathsf{PE.Encrypt}^*(\mathsf{decom}, y_e);} \\ \boxed{\hbar_e := H(0, \mathsf{com}, \boldsymbol{\delta}_{y_s}, \mathsf{vk}, C_{\mathsf{cpa}});} \\ \boxed{C_0 := g^{s_0(\theta_1 \hbar_e + \theta_2)}, \text{where } s_0 \text{ is a randomness in } \mathsf{PE.Encrypt}^*;} \\ \boxed{\delta_o \longleftarrow \mathsf{OTS.Sign}(C_0 || y_s, \mathsf{signk});} \\ \boxed{\text{returns } \mathsf{U} := (\mathsf{com}, \delta := (\boldsymbol{\delta}_{y_s}, \delta_o, \mathsf{vk}), \mathsf{CT} := (C_{\mathsf{cpa}}, C_0))} \end{array} \right)$

- $\mathsf{Unsigncrypt}(\mathsf{U}, \mathcal{SK}_x, y_s) := \begin{cases} m & \text{if } \left(\begin{array}{c} \boxed{\mathsf{OTS.Ver}(C_0 || y_s, \delta_o, \mathsf{vk}) = 1;} \\ \boxed{\mathsf{PS.Ver}(\mathsf{vk}, \boldsymbol{\delta}_{y_s}, y_s) = 1; \parallel \mathsf{decom} \longleftarrow \mathsf{PE.Decrypt}(\mathsf{CT}, \mathcal{SK}_x);} \\ \boxed{\text{let } m \longleftarrow \mathsf{Open}(\mathsf{com}, d)} \end{array} \right) \\ \perp & \text{otherwise.} \end{cases}$

Correctness. Following the correctness of primitive predicate signature scheme PS, predicate encryption scheme PE, commitment scheme $\mathcal{C}$ and one-time signature scheme OTS, we have the correctness of the proposed construction.

Fact 7.1. Following Facts 3.4 and 6.3, we have if $|\boldsymbol{c}_y| = \omega_1$, then size of the signcryption (mainly the #group elements) is $2\omega_1 + 3$ and the number of pairings in $\mathsf{Unsigncrypt}$ is $2(\omega_1 + 1)$. Since Ver and $\mathsf{Decrypt}$ run in parallel in $\mathsf{Unsigncrypt}$, the number of pairings is counted to be $\omega_1 + 1$. **Therefore, if $\boldsymbol{c}_y$ of the underlying pair encoding scheme has constant-size, then the corresponding signcryption will be of constant-size and the number of pairings in $\mathsf{Unsigncrypt}$ is constant.**

Remark 7.2. The $\mathsf{Signcrypt}$ and $\mathsf{Unsigncrypt}$ work almost in black-box manner except the $\hbar_e$ is computed as $H(0, \mathsf{com}, \boldsymbol{\delta}_{y_s}, \mathsf{vk}, C_{\mathsf{cpa}})$ instead of $H(C_{\mathsf{cpa}})$ in $\mathsf{Encrypt}$ and $\mathsf{Decrypt}$.

Discussion 7.3. We see later that for confidentiality of the proposed signcryption, we require hiding property of the primitive commitment scheme $\mathcal{C}$. But, for unforgeability, we do not assume any property of the commitment scheme $\mathcal{C}$.

Variants of signcryption. Below, we discuss the simple variants of our proposed signcryption.

- If we ignore the objects related to commitment from the proposed construction and make a minor change, $C_{\mathsf{cpa}} \leftarrow \mathsf{PE.Encrypt}^*(m, y_e)$, then we obtain a simple predicate signcryption. This variant of signcryption has the same performance as the proposed signcryption except the simple variant does not attain the public verifiability.
- If we ignore the OTS scheme and apply the modification, $\boldsymbol{\delta}_{y_s} \leftarrow \mathsf{PS.Sign}(\mathsf{com}||y_e, \mathcal{SK}_x, y_e)$ to the original construction, we obtain a new signcryption. This variant has the same performance as the proposed signcryption except the new variant does not guarantee the strong unforgeability. For proving the weak unforgeability of this new signcryption, we have to assume the relax-binding property of $\mathcal{C}$.

8 Security of the Proposed Predicate Signcryption

The algorithms of PE and PS run almost as black-boxes in $\mathsf{Signcrypt}$ and $\mathsf{Unsigncrypt}$ algorithms of the proposed PSC. But, the confidentiality and unforgeability of the proposed PSC is proven without assuming PE and PS as black-boxes respectively. The reason is the following. All the signcrypt (resp. unsigncrypt) oracle queries made by $\mathscr{A}$ in APs-IND-CCA (resp. APs-UF-CMA) model could not be answered using black-box proof of PE in Section 6.1 (resp. PS in Section 3.3).

Theorem 8.1. *Our proposed predicate signcryption scheme in Section 7 is perfectly private (Definition 2.17).*

Proof. Since the $\mathcal{SK}_x$ is only used to generate $\boldsymbol{\delta}_{y_s}$ and the PS scheme in Section 3.3 is perfectly private, so we are done. □

Theorem 8.2. *Let* P *be a pair encoding scheme for a predicate* $\sim$ *which satisfies* **Conditions 3.1** *and* $\sim$ *is domain-transferable. Suppose* P *has both the security,* SMH *and* CMH*, the assumptions, DSG1, DSG2 and DSG3 hold in* $\mathcal{J}$*, the one-time signature scheme* OTS *has strong unforgeability, the commitment scheme* $\mathcal{C}$ *has the hiding property and* H *is a collision resistant hash function, then the proposed predicate signcryption scheme* PSC *in Section 7 for the predicate* $\sim$ *is IND-CCA secure in adaptive-predicates model (Definition 2.18).*

Proof. For proof, refer to Appendix C □

Theorem 8.3. *Let* P *be a pair encoding scheme for a predicate* $\sim$ *which satisfies* **Conditions 3.1** *and* $\sim$ *is domain-transferable. Suppose* P *has the* CMH *security, the assumptions, DSG1, DSG2 and DSG3 hold in* $\mathcal{J}$*, the one-time signature scheme* OTS *has strong unforgeability and* H *is a collision resistant hash function, then the proposed predicate signcryption scheme* PSC *in Section 7 for the predicate* $\sim$ *is strongly existential unforgeable in adaptive-predicates model (Definition 2.22).*

Proof. For proof, refer to Appendix C □

9 Instantiations of Predicate Signcryption

Using the different pair encoding schemes of [2, 6, 52], we instantiate many new results which are given below.

– Our framework for PSC provides predicate signcryptions for regular languages in both policies, key-policy (KP) and signcryption-policy (SCP). Both the signcryptions support large universe alphabet set. The KP-PSC and SCP-PSC are derived using Pair Encoding Schemes 3 and 7 of [2] respectively.

– Unbounded KP-ABSC and SCP-ABSC schemes with large universes can be instantiated using Pair Encoding Scheme 4 of [2] and its dual [6] respectively.

– We provide a KP-ABSC scheme with constant-size signcryptions and the number of pairings required to unsigncrypt is also constant. This signcryption is instantiated from Pair Encoding Scheme 5 of [2]. Since $|\boldsymbol{c}_y| = 6$, following **Fact 7.1**, we have $|\mathsf{U}| = 15$ and the number of pairings in Unsigncrypt is 14. Similarly, by applying dual [6] on Pair Encoding Scheme 5 of [2], we can obtain an SCP-ABSC scheme with constant-size keys.

– If we consider signcryption analogue of KP-DSS, then we have a new predicate signcryption, called key-policy over doubly-spatial signcryption (KP-DSSC). A KP-DSSC is instantiated from Pair Encoding Scheme 6 of [2]. The dual version of KP-DSSC is called signcryption-policy over doubly-spatial signcryption (SCP-DSSC). By applying our framework on dual [6] of Pair Encoding Scheme 6 of [2], we can instantiate an SCP-DSSC. Similar to KP-DSS (resp. SP-DSS), KP-DSSC (resp. SCP-DSSC) generalizes the existing class, KP-ABSC (resp. SCP-ABSC).

– Cost-free ABSC schemes in both KP and SCP forms with small universes can be derived from Pair Encoding Schemes 8 and 10 of [2] respectively.

– Similar to the cost free ABS with large universes (in Section 5), we can also obtain cost free ABSC schemes with large universes in signcryption-policy and key-policy forms. The KP-ABSC and SCP-ABSC with large universes are instantiated from Pair Encoding Schemes 12 and 13 of [2] respectively.

– (Other Cost Free Signcryptions.) Similar to the other cost free signatures (in Section 5), we instantiate other cost free signcryptions as follows. A doubly-spatial signcryption (DSSC) (special case of KP-DSSC) is instantiated using Pair Encoding Scheme 14 of [2]. A predicate signcryption for negated spatial predicate [5] is obtained using Pair Encoding Scheme 15 of [2]. Using the pair encodings of [52] for the inner product predicate [30], we obtain inner product signcryption (IPSC) schemes with constant-size keys and constant-size signcryptions respectively. We generate non-zero inner product signcryption (NIPSC) schemes with respectively constant-size keys and constant-size signcryptions using the pair encodings of [52] for non-zero inner product predicate. A spatial signcryption (special case of DSSC) scheme with constant-size signcryptions can be obtained using a pair encoding of [52].

10 Conclusion

In this paper, for the first time we showed that the pair encodings provide the adaptively unforgeable predicate signatures with prefect privacy. Then, we have shown that the pair encodings can also be applied to construct fully CCA-secure predicate encryption with almost the same cost as the CPA-secure PE of [2]. Finally, we explored a generic framework for predicate signcryptions using the pair encodings. The proposed signcryptions have been shown to be strongly unforgeable and IND-CCA secure in the adaptive-predicates models and achieve signer privacy. Since, the Sign (resp. Ver) and Encrypt (resp. Decrypt) run in parallel in Signcrypt (resp. Unsigncrypt), the execution is comparatively faster. For all the frameworks, we have instantiated many schemes with new features using the existing pair encoding schemes. In future, similar to the prime order variants [3, 19] of [2, 52], we will be focusing on the prime order variant of all the constructions presented in this paper.

Acknowledgements. We would like to thank Prof. Rana Barua, Subhabrata Samajder and the anonymous reviewers of PKC, 2016 for their comments and suggestions that helped in polishing the technical and editorial content of this paper.

References

[1] Jee Hea An, Yevgeniy Dodis, and Tal Rabin. On the security of joint signature and encryption. In *EUROCRYPT*, volume 2332 of *LNCS*, pages 83–107. Springer, 2002.

[2] Nuttapong Attrapadung. Dual system encryption via doubly selective security: Framework, fully secure functional encryption for regular languages, and more. In *EUROCRYPT*, volume 8441 of *LNCS*, pages 557–577. Springer, 2014.

[3] Nuttapong Attrapadung. Dual system encryption framework in prime-order groups. Cryptology ePrint Archive, Report 2015/390, 2015. http://eprint.iacr.org/.

[4] Nuttapong Attrapadung, Goichiro Hanaoka, and Shota Yamada. Conversions among Several Classes of Predicate Encryption and Applications to ABE with Various Compactness Tradeoffs. In *ASIACRYPT*, volume 9452 of *LNCS*, pages 575–601. Springer, 2015.

[5] Nuttapong Attrapadung and Benoît Libert. Functional encryption for inner product: Achieving constant-size ciphertexts with adaptive security or support for negation. In *PKC*, volume 6056 of *LNCS*, pages 384–402. Springer, 2010.

[6] Nuttapong Attrapadung and Shota Yamada. Duality in ABE: Converting attribute based encryption for dual predicate and dual policy via computational encodings. In *CT-RSA*, volume 9048 of *LNCS*, pages 616–637. Springer, 2015.

[7] Mihir Bellare and Georg Fuchsbauer. Policy-based signatures. In *PKC*, volume 8383 of *LNCS*, pages 520–537. Springer, 2014.

[8] Johannes Blömer and Gennadij Liske. Construction of fully cca-secure predicate encryptions from pair encoding schemes. In *CT-RSA*, volume 9610 of *LNCS*, pages 431–447. Springer, 2016.

[9] Dan Boneh and Xavier Boyen. Efficient selective-ID secure identity-based encryption without random oracles. In *EUROCRYPT*, volume 3027 of *LNCS*, pages 223–238. Springer, 2004.

[10] Dan Boneh and Xavier Boyen. Secure identity based encryption without random oracles. In *CRYPTO*, volume 3152 of *LNCS*, pages 443–459. Springer, 2004.

[11] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-dnf formulas on ciphertexts. In *TCC*, volume 3378 of *LNCS*, pages 325–341. Springer, 2005.

[12] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In *TCC*, volume 6597 of *LNCS*, pages 253–273. Springer, 2011.

[13] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: a new vision for public-key cryptography. *Communications of the ACM*, 55(11):56–64, 2012.

[14] Xavier Boyen. Mesh signatures. In *EUROCRYPT*, volume 4515 of *LNCS*, pages 210–227. Springer, 2007.

[15] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In *PKC*, volume 8383 of *LNCS*, pages 501–519. Springer, 2014.

[16] David Chaum and Eugéne van Heyst. Group signatures. In *EUROCRYPT*, volume 547 of *LNCS*, pages 257–265. Springer, 1991.

[17] Cheng Chen, Jie Chen, Hoon Wei Lim, Zhenfeng Zhang, and Dengguo Feng. Combined public-key schemes: The case of ABE and ABS. In *PROVSEC*, volume 7496 of *LNCS*, pages 53–69. Springer, 2012.

[18] Jie Chen and Hoeteck Wee. Doubly spatial encryption from DBDH. Cryptology ePrint Archive, Report 2014/199, 2014. http://eprint.iacr.org/.

[19] Jie Chen and Hoeteck Wee. Improved dual system ABE in prime-order groups via predicate encodings. In *EUROCRYPT*, volume 9057 of *LNCS*, pages 595–624. Springer, 2015.

[20] Pratish Datta, Ratna Dutta, and Sourav Mukhopadhyay. Functional signcryption: Notion, construction, and applications. In *PROVSEC*, volume 9451 of *LNCS*, pages 268–288. Springer, 2015.

[21] Keita Emura, Atsuko Miyaji, and Mohammad Shahriar Rahman. Dynamic attribute-based signcryption without random oracles. *International Journal of Applied Cryptography*, 2(11):199–211, 2012.

[22] Paul Erdös, Peter Frankl, and Zoltán Füredi. Families of finite sets in which no set is covered by the union of r others. *Israel Journal of Mathematics*, 51(1-2):79–89, 1985.

[23] Alex Escala, Javier Herranz, and Paz Morillo. Revocable attribute-based signatures with adaptive security in the standard model. In *AFRICACRYPT*, volume 6737 of *LNCS*, pages 224–241. Springer, 2011.

[24] Martin Gagné, Shivaramakrishnan Narayan, and Reihaneh Safavi-Naini. Threshold attribute-based signcryption. In *SCN*, volume 6280 of *LNCS*, pages 154–171. Springer, 2010.

[25] Jens Groth, Rafail Ostrovsky, and Amit Sahai. New techniques for non-interactive zero-knowledge. *Journal of ACM*, 59(3):11:1–11:35, June, 2012.

[26] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In *EUROCRYPT*, volume 4965 of *LNCS*, pages 415–432. Springer, 2008.

[27] Stuart Haber and Benny Pinkas. Securely combining public-key cryptosystems. In *ACM Conference on Computer and Communications Security*, pages 215–224. ACM, 2001.

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

[29] Kenneth Hoffman and Ray Kunze. *Linear Algebra*. Prentice Hall of India, New Delhi, 2nd edition, 1991.

[30] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *EUROCRYPT*, volume 4965 of *LNCS*, pages 146–162. Springer, 2008.

[31] Ravi Kumar, Sridhar Rajagopalan, and Amit Sahai. Coding constructions for blacklisting problems without computational assumptions. In *CRYPTO*, volume 1666 of *LNCS*, pages 609–623. Springer, 1999.

[32] Allison Lewko and Brent Waters. New techniques for dual system encryption and fully secure HIBE with short ciphertexts. In *TCC*, volume 5978 of *LNCS*, pages 455–479. Springer, 2010.

[33] Allison B. Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In *EUROCRYPT*, volume 6110 of *LNCS*, pages 62–91. Springer, 2010.

[34] Jin Li, Man Ho Au, Willy Susilo, Dongqing Xie, and Kui Ren. Attribute-based signature and its applications. In *ACM Conference on Computer and Communications Security*, pages 60–69. ACM, 2010.

[35] Benoît Libert and Jean-Jacques Quisquater. Efficient signcryption with key privacy from gap diffie-hellman groups. In *PKC*, volume 2947 of *LNCS*, pages 187–200. Springer, 2004.

[36] Hemanta Maji, Manoj Prabhakaran, and Mike Rosulek. Attribute-based signatures: Achieving attribute-privacy and collusion-resistance. Cryptology ePrint Archive, Report 2008/328, 2008. http://eprint.iacr.org/.

[37] Hemanta Maji, Manoj Prabhakaran, and Mike Rosulek. Attribute-based signatures. In *CT-RSA*, volume 6558 of *LNCS*, pages 376–392. Springer, 2011.

[38] Takahiro Matsuda, Kanta Matsuura, and Jacob C. N. Schuldt. Efficient constructions of signcryption schemes and signcryption composability. In *INDOCRYPT*, volume 5922 of *LNCS*, pages 321–342. Springer, 2009.

[39] Mridul Nandi and Tapas Pandit. Generic conversions from CPA to CCA secure functional encryption. Cryptology ePrint Archive, Report 2015/457, 2015. http://eprint.iacr.org/, submited to journal.

[40] Tatsuaki Okamoto and Katsuyuki Takashima. Hierarchical predicate encryption for inner-products. In *ASIACRYPT*, volume 5912 of *LNCS*, pages 214–231. Springer, 2009.

[41] Tatsuaki Okamoto and Katsuyuki Takashima. Efficient attribute-based signatures for non-monotone predicates in the standard model. In *PKC*, volume 6571 of *LNCS*, pages 35–52. Springer, 2011.

[42] Tatsuaki Okamoto and Katsuyuki Takashima. Decentralized attribute-based signatures. In *PKC*, volume 7778 of *LNCS*, pages 125–142. Springer, 2013.

[43] Tapas Pandit, Sumit Kumar Pandey, and Rana Barua. Attribute-based signcryption: Signer privacy, strong unforgeability and IND-CCA2 security in adaptive-predicates attack. In *PROVSEC*, volume 8782 of *LNCS*, pages 274–290. Springer, 2014.

[44] Y. Sreenivasa Rao and Ratna Dutta. Expressive bandwidth-efficient attribute based signature and signcryption in standard model. In *ACISP*, volume 8544 of *LNCS*, pages 209–225. Springer, 2014.

[45] Yusuke Sakai, Nuttapong Attrapadung, and Goichiro Hanaoka. Attribute-based signatures for circuits from bilinear map. In *PKC*, volume 9614 of *LNCS*, pages 283–300. Springer, 2016.

[46] Siamak F Shahandashti and Reihaneh Safavi-Naini. Threshold attribute-based signatures and their application to anonymous credential systems. In *AFRICACRYPT*, volume 5580 of *LNCS*, pages 198–216. Springer, 2009.

[47] Guo Shaniqng and Zeng Yingpei. Attribute-based signature scheme. In *ISA*, pages 509–511. IEEE, 2008.

[48] María Isabel González Vasco, Florian Hess, and Rainer Steinwandt. Combined (identity-based) public key schemes. Cryptology ePrint Archive, Report 2008/466, 2008. http://eprint.iacr.org/.

[49] Changji Wang and Jiasen Huang. Attribute-based signcryption with ciphertext-policy and claim-predicate mechanism. In *Computational Intelligence and Security (CIS), 2011 Seventh International Conference on*, pages 905–909. IEEE, 2011.

[50] Brent Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In *CRYPTO*, volume 5677 of *LNCS*, pages 619–636. Springer, 2009.

[51] Brent Waters. Functional encryption for regular languages. In *CRYPTO*, volume 7417 of *LNCS*, pages 218–235. Springer, 2012.

[52] Hoeteck Wee. Dual system encryption via predicate encodings. In *TCC*, volume 8349 of *LNCS*, pages 455–479. Springer, 2014.

[53] Shota Yamada, Nuttapong Attrapadung, Goichiro Hanaoka, and Noboru Kunihiro. Generic constructions for chosen-ciphertext secure attribute based encryption. In *PKC*, volume 6571 of *LNCS*, pages 71–89. Springer, 2011.

[54] Shota Yamada, Nuttapong Attrapadung, Bagus Santoso, Jacob C. N. Schuldt, Goichiro Hanaoka, and Noboru Kunihiro. Verifiable predicate encryption and applications to CCA security and anonymous predicate authentication. In *PKC*, volume 7293 of *LNCS*, pages 243–261. Springer, 2012.

[55] Piyi Yang, Zhenfu Cao, , and Xiaolei Dong. Fuzzy identity based signature. Cryptology ePrint Archive, Report 2008/02, 2008. http://eprint.iacr.org/.

[56] Yuliang Zheng. Digital signcryption or how to achieve cost(signature & encryption) ≪ cost(signature) + cost(encryption). In *CRYPTO*, volume 1294 of *LNCS*, pages 165–179. Springer, 1997.

A Lemmas Used in the Proof of Theorem 4.2

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

$$|\mathsf{Adv}^{\mathrm{Real}}_{\mathscr{A},\mathrm{PS}}(\kappa) - \mathsf{Adv}^{\mathrm{Res}}_{\mathscr{A},\mathrm{PS}}(\kappa)| \leq \mathsf{Adv}^{\mathrm{DSG2}}_{\mathscr{B}}(\kappa).$$

Proof. Suppose an adversary can distinguish the games with a non-negligible probability. Then we will establish a PPT simulator $\mathscr{B}$ for breaking the DSG2 assumption with the same probability. An instance of DSG2, $(\mathcal{J}, g, Z_1Z_2, W_2W_3, Z_3, T_\beta)$ with $\beta \xleftarrow{\mathrm{U}} \{0,1\}$ is given to $\mathscr{B}$. The only difference between the games, Game$_{Real}$ and Game$_{Res}$ 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 $T_\beta^B \overset{?}{=} 1_{\mathbb{G}}$, $\mathscr{B}$ 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 $Y_2 := (W_2W_3)^B = W_2^{p_3}$ else $Y_2 := (Z_1Z_2)^B = Z_2^{p_1}$. In both case, we have $Y_2 \in \mathbb{G}_{p_2}$, then by checking $e(T_\beta, Y_2) \overset{?}{=} 1$, $\mathscr{B}$ can break the DSG2 assumption. □

Lemma A.2. Game$_{Res}$ *and* Game$_0$ *are indistinguishable under the DSG1 assumption. That is, for every adversary $\mathscr{A}$, there exists a PPT algorithm $\mathscr{B}$ such that*

$$|\mathsf{Adv}_{\mathscr{A},\mathrm{PS}}^{\mathrm{Res}}(\kappa) - \mathsf{Adv}_{\mathscr{A},\mathrm{PS}}^{0}(\kappa)| \leq \mathsf{Adv}_{\mathscr{B}}^{\mathrm{DSG1}}(\kappa).$$

Proof. We establish a PPT simulator $\mathscr{B}$ who receives an instance of DSG1, $(\mathcal{J},\ g,\ Z_3,\ T_\beta)$ with $\beta \xleftarrow{\mathrm{U}} \{0,1\}$ and depending on the distribution of β, it simulates either Game$_{Res}$ or Game$_0$.

Setup: $\mathscr{B}$ chooses $\alpha, \theta_1, \theta_2 \xleftarrow{\mathrm{U}} \mathbb{Z}_N$, $\boldsymbol{h} \xleftarrow{\mathrm{U}} \mathbb{Z}_N^n$ and sets $\boldsymbol{h}_{\mathsf{M}} := (\theta_1, \theta_2, \boldsymbol{h})$. Let $H : \{0,1\}^* \longrightarrow \mathbb{Z}_N$ be a hash function. Then, it provides $\mathcal{PP} := (\mathcal{J}, g, g^{\boldsymbol{h}_{\mathsf{M}}}, g_T^{\alpha} := e(g,g)^{\alpha}, Z_3, H)$ to $\mathscr{A}$ and keeps $\mathcal{MSK} := (\alpha)$ to itself. It implicitly sets $\hat{\boldsymbol{h}}_{\mathsf{M}} :\equiv \boldsymbol{h}_{\mathsf{M}} \mod p_2$. By Chinese Remainder Theorem (CRT), $\hat{\boldsymbol{h}}_{\mathsf{M}}$ is independent from $\boldsymbol{h}_{\mathsf{M}} \mod p_1$ and so $\hat{\boldsymbol{h}}_{\mathsf{M}}$ is perfectly distributed.

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

- $\mathsf{KeyGen}(x)$: It is a query for normal key. $\mathscr{B}$ can handle the key query of $\mathscr{A}$, since the $\mathcal{MSK}$ is known to him.
- $\mathsf{Sign}(m, x, y)$: If $x \not\sim y$, it returns $\perp$. It is a query for normal signature. $\mathscr{B}$ can answer the query of $\mathscr{A}$, since he can construct $\mathcal{SK}_x$ using the $\mathcal{MSK}$ known to him.

Forgery: $\mathscr{A}$ outputs a signature $\boldsymbol{\delta}_{y^*}$ for (m^*, y^*). Then, $\mathscr{B}$ prepares a vText for (m^*, y^*) as follows: It computes $\hbar^* := H(m^*, y^*)$. Runs $(\boldsymbol{c}_{y^*}, \omega_2) \longleftarrow \mathsf{Enc2}(y^*, N)$ with $|\boldsymbol{c}_{y^*}| = \omega_1$ and sets $\boldsymbol{c}_{y^*}^{\mathsf{M}} := (c_0, \boldsymbol{c}_{y^*})$. Then picks $\boldsymbol{s}' := (s'_0, \ldots, s'_{\omega_2}) \xleftarrow{\mathrm{U}} \mathbb{Z}_N^{\omega_2+1}$. Finally, computes the vText as $\mathcal{V} := (\mathcal{V}_{\mathsf{INT}} := e(g^{\alpha}, T_\beta)^{s'_0}, \boldsymbol{\mathcal{V}}_{y^*} := T_\beta^{\boldsymbol{c}_{y^*}^{\mathsf{M}}(\boldsymbol{s}', \boldsymbol{h}_{\mathsf{M}})})$. $\mathscr{B}$ returns 1 if $e(\boldsymbol{\delta}_{y^*}, \boldsymbol{\mathcal{V}}_{y^*}) = \mathcal{V}_{\mathsf{INT}}$ else 0.

Analysis: We will show that all the objects are perfectly distributed as required. $\mathscr{B}$ implicitly sets $g^{t_1} := T_\beta\big|_{\mathbb{G}_{p_1}}$ and for $\beta = 1$, $g_2^{t_2} := T_\beta\big|_{\mathbb{G}_{p_2}}$. Then by linearity of P, we have $g^{t_1\boldsymbol{c}_{y^*}^{\mathsf{M}}(\boldsymbol{s}', \boldsymbol{h}_{\mathsf{M}})} = g^{\boldsymbol{c}_{y^*}^{\mathsf{M}}(t_1\boldsymbol{s}', \boldsymbol{h}_{\mathsf{M}})}$ and $g_2^{t_2\boldsymbol{c}_{y^*}^{\mathsf{M}}(\boldsymbol{s}', \boldsymbol{h}_{\mathsf{M}})} = g_2^{\boldsymbol{c}_{y^*}^{\mathsf{M}}(t_2\boldsymbol{s}', \hat{\boldsymbol{h}}_{\mathsf{M}})}$. $\mathscr{B}$ implicitly sets $\boldsymbol{s} :\equiv t_1\boldsymbol{s}' \mod p_1$ and for $\beta = 1$, $\hat{\boldsymbol{s}} :\equiv t_2\boldsymbol{s}' \mod p_2$. By CRT, $\boldsymbol{s}' \mod p_1$ is independent from $\boldsymbol{s}' \mod p_2$ and therefore $\boldsymbol{s}$ and $\hat{\boldsymbol{s}}$ are perfectly distributed as required. Altogether, we have that the joint distribution of all the objects simulated by $\mathscr{B}$ is identical to that of Game$_{Res}$ if $\beta = 0$ else Game$_0$. □

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

$$|\mathsf{Adv}_{\mathscr{A},\mathrm{PS}}^{1-(k-1)-3}(\kappa) - \mathsf{Adv}_{\mathscr{A},\mathrm{PS}}^{1-k-1}(\kappa)| \leq \mathsf{Adv}_{\mathscr{B}}^{\mathrm{DSG2}}(\kappa) \ \ \textit{for} \ \ 1 \leq k \leq \nu_1.$$

Proof. We establish a PPT simulator $\mathscr{B}$ who receives an instance of DSG2, $(\mathcal{J},\ g,\ Z_1Z_2,\ W_2W_3,\ Z_3,\ T_\beta)$ with $\beta \xleftarrow{\mathrm{U}} \{0,1\}$ and depending on the distribution of β, it simulates either Game$_{1-(k-1)-3}$ or Game$_{1-k-1}$.

Setup: $\mathscr{B}$ chooses $\alpha, \theta_1, \theta_2 \xleftarrow{\text{U}} \mathbb{Z}_N$, $\boldsymbol{h} \xleftarrow{\text{U}} \mathbb{Z}_N^n$ and sets $\boldsymbol{h}_{\mathsf{M}} := (\theta_1, \theta_2, \boldsymbol{h})$. Let $H : \{0,1\}^* \longrightarrow \mathbb{Z}_N$ be a hash function. Then, it provides $\mathcal{PP} := (\mathcal{J}, g, g^{\boldsymbol{h}_{\mathsf{M}}}, g_T^{\alpha} := e(g,g)^{\alpha}, Z_3, H)$ to $\mathscr{A}$ and keeps $\mathcal{MSK} := (\alpha)$ to itself. It implicitly sets $\hat{\boldsymbol{h}}_{\mathsf{M}} :\equiv \boldsymbol{h}_{\mathsf{M}} \mod p_2$. By CRT, $\hat{\boldsymbol{h}}_{\mathsf{M}}$ is independent from $\boldsymbol{h}_{\mathsf{M}} \mod p_1$ and so $\hat{\boldsymbol{h}}_{\mathsf{M}}$ is perfectly distributed.

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

– $\mathsf{KeyGen}(x)$: Let x_j be the j^{th} query key-index. $\mathscr{B}$ answers the key $\mathcal{SK}_{x_j}$ as follows:

- If $j > k$, then $\mathscr{B}$ runs the KeyGen algorithm and gives the normal key to $\mathscr{A}$.
- If $j < k$, then it is of sf-type 3 key. $\mathscr{B}$ runs $(\boldsymbol{k}_{x_j}, m_2) \longleftarrow \mathsf{Enc1}(x_j, N)$ with $|\boldsymbol{k}_{x_j}| = m_1$. Picks $\alpha'_j \xleftarrow{\text{U}} \mathbb{Z}_N$, $\boldsymbol{r}_j \xleftarrow{\text{U}} \mathbb{Z}_N^{m_2}$ and $\boldsymbol{R}_3 \xleftarrow{\text{U}} \mathbb{G}_{p3}^{m_1}$. It computes the sf-type 3 key as defined below:

  $$\mathcal{SK}_{x_j} := g^{\boldsymbol{k}_{x_j}(\alpha, \boldsymbol{r}_j, \boldsymbol{h})} \cdot (W_2W_3)^{\boldsymbol{k}_{x_j}(\alpha'_j, \boldsymbol{0}, \boldsymbol{0})} \cdot \boldsymbol{R}_3.$$

  It implicitly sets $\hat{\alpha}_j := w_2\alpha'_j$, where $W_2W_3 = g_2^{w_2}g_3^{w_3}$. So, $\mathcal{SK}_{x_j}$ is properly distributed sf-type 3 key.
- If $j = k$ then it is either normal or sf-type 1 key. $\mathscr{B}$ runs $(\boldsymbol{k}_{x_k}, m_2) \longleftarrow \mathsf{Enc1}(x_k, N)$ with $|\boldsymbol{k}_{x_k}| = m_1$. Picks $\boldsymbol{r}'_k, \hat{\boldsymbol{r}}'_k \xleftarrow{\text{U}} \mathbb{Z}_N^{m_2}$ and $\boldsymbol{R}_3 \xleftarrow{\text{U}} \mathbb{G}_{p3}^{m_1}$. It generates the following $\mathcal{SK}_{x_j}$ using T_β of the instance of DSG2:

  $$\mathcal{SK}_{x_k} := g^{\boldsymbol{k}_{x_k}(\alpha, \boldsymbol{r}'_k, \boldsymbol{h})} \cdot T_\beta^{\boldsymbol{k}_{x_k}(0, \hat{\boldsymbol{r}}'_k, \boldsymbol{h})} \cdot \boldsymbol{R}_3.$$

  $\mathscr{B}$ implicitly sets $g^{t_1} := T_\beta\big|_{\mathbb{G}_{p_1}}$ and for $\beta = 1$, $g_2^{t_2} := T_\beta\big|_{\mathbb{G}_{p_2}}$. Then by linearity of P, we have $g^{\boldsymbol{k}_{x_k}(\alpha,\ \boldsymbol{r}'_k,\ \boldsymbol{h})} \cdot g^{t_1\boldsymbol{k}_{x_k}(0,\ \hat{\boldsymbol{r}}'_k,\ \boldsymbol{h})} = g^{\boldsymbol{k}_{x_k}(\alpha,\ \boldsymbol{r}'_k + t_1\hat{\boldsymbol{r}}'_k,\ \boldsymbol{h})}$ and $g_2^{t_2\boldsymbol{k}_{x_k}(0,\ \hat{\boldsymbol{r}}'_k,\ \boldsymbol{h})} = g_2^{\boldsymbol{k}_{x_k}(0,\ t_2\hat{\boldsymbol{r}}'_k,\ \hat{\boldsymbol{h}})}$. $\mathscr{B}$ implicitly sets $\boldsymbol{r}_k := \boldsymbol{r}'_k + t_1\hat{\boldsymbol{r}}'_k$ and $\hat{\boldsymbol{r}}_k := t_2\hat{\boldsymbol{r}}'_k$. Since $\boldsymbol{r}'_k$ and $\hat{\boldsymbol{r}}'_k$ are chosen uniformly and independently from $\mathbb{Z}_N^{m_2}$, then so are $\boldsymbol{r}_k$ and $\hat{\boldsymbol{r}}_k$. Therefore, $\mathcal{SK}_{x_k}$ is perfectly distributed normal (resp. sf-type 1) key if $\beta = 0$ (resp. $\beta = 1$).

– $\mathsf{Sign}(m, x, y)$: If $x \not\sim y$, it returns $\perp$. It is a query for normal signature. $\mathscr{B}$ can answer the query of $\mathscr{A}$ as the $\mathcal{MSK}$ is known to him.

Forgery: $\mathscr{A}$ outputs a signature $\boldsymbol{\delta}_{y^*}$ for (m^*, y^*). Then, $\mathscr{B}$ prepares a vText for (m^*, y^*) as follows: It computes $\hbar^* := H(m^*, y^*)$. Runs $(\boldsymbol{c}_{y^*}, \omega_2) \longleftarrow \mathsf{Enc2}(y^*, N)$ with $|\boldsymbol{c}_{y^*}| = \omega_1$ and sets $\boldsymbol{c}_{y^*}^{\mathsf{M}} := (c_0, \boldsymbol{c}_{y^*})$. Then picks $\boldsymbol{s}' := (s'_0, \ldots, s'_{\omega_2}) \xleftarrow{\text{U}} \mathbb{Z}_N^{\omega_2+1}$. Finally, computes $\mathcal{V} := (\mathcal{V}_{\mathsf{INT}} := e(g^{\alpha}, Z_1Z_2)^{s'_0}, \boldsymbol{\mathcal{V}}_{y^*} := (Z_1Z_2)^{\boldsymbol{c}_{y^*}^{\mathsf{M}}(\boldsymbol{s}', \boldsymbol{h}_{\mathsf{M}})})$. $\mathscr{B}$ returns 1 if $e(\boldsymbol{\delta}_{y^*}, \boldsymbol{\mathcal{V}}_{y^*}) = \mathcal{V}_{\mathsf{INT}}$ else 0.

Analysis: We will show that all the objects are perfectly distributed as required. Let $Z_1Z_2 = g^{z_1}g_2^{z_2}$. Then by linearity of P, we have $g^{z_1\boldsymbol{c}_{y^*}^{\mathsf{M}}(\boldsymbol{s}', \boldsymbol{h}_{\mathsf{M}})} = g^{\boldsymbol{c}_{y^*}^{\mathsf{M}}(z_1\boldsymbol{s}', \boldsymbol{h}_{\mathsf{M}})}$ and $g_2^{z_2\boldsymbol{c}_{y^*}^{\mathsf{M}}(\boldsymbol{s}', \boldsymbol{h}_{\mathsf{M}})} = g_2^{\boldsymbol{c}_{y^*}^{\mathsf{M}}(z_2\boldsymbol{s}', \hat{\boldsymbol{h}}_{\mathsf{M}})}$. $\mathscr{B}$ implicitly sets $\boldsymbol{s} :\equiv z_1\boldsymbol{s}' \mod p_1$ and $\hat{\boldsymbol{s}} :\equiv z_2\boldsymbol{s}' \mod p_2$. By CRT, $\boldsymbol{s}' \mod p_1$ is independent from $\boldsymbol{s}' \mod p_2$ and therefore $\boldsymbol{s}$ and $\hat{\boldsymbol{s}}$ are perfectly distributed as required. Altogether, we have that the joint distribution of all the objects simulated by $\mathscr{B}$ is identical to that of $\text{Game}_{1-(k-1)-3}$ if $\beta = 0$ else Game_{1-k-1}. □

Lemma A.4. Game_{1-k-1} *and* Game_{1-k-2} *are indistinguishable under the* CMH *security of primitive pair encoding scheme,* P*. That is, for every adversary* $\mathscr{A}$*, there exists a PPT algorithm* $\mathscr{B}$ *such that*

$$|\mathsf{Adv}_{\mathscr{A},\text{PS}}^{1-k-1}(\kappa) - \mathsf{Adv}_{\mathscr{A},\text{PS}}^{1-k-2}(\kappa)| \leq \mathsf{Adv}_{\mathscr{B},\text{P}}^{\text{CMH}}(\kappa) \;\; \textit{for} \;\; 1 \leq k \leq \nu_1.$$

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

Setup: The challenger $\mathcal{CH}$ of P gives $(g, g_2, g_3) \in \mathbb{G}_{p_1} \times \mathbb{G}_{p_2} \times \mathbb{G}_{p_3}$ to $\mathscr{B}$. $\mathscr{B}$ chooses $\alpha, \theta_1, \theta_2 \xleftarrow{\text{U}} \mathbb{Z}_N$, $\boldsymbol{h} \xleftarrow{\text{U}} \mathbb{Z}_N^n$ and sets $\boldsymbol{h}_{\mathsf{M}} := (\theta_1, \theta_2, \boldsymbol{h})$. Let $H : \{0,1\}^* \longrightarrow \mathbb{Z}_N$ be a hash function. Then, it provides $\mathcal{PP} := (\mathcal{J}, g, g^{\boldsymbol{h}_{\mathsf{M}}}, g_T^{\alpha} := e(g,g)^{\alpha}, Z_3 := g_3, H)$ to $\mathscr{A}$ and keeps $\mathcal{MSK} := (\alpha)$ and g_2 to itself.

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

– $\mathsf{KeyGen}(x)$: Let x_j be the j^{th} query key-index. $\mathscr{B}$ answers the key $\mathcal{SK}_{x_j}$ as follows:

- If $j > k$, then $\mathscr{B}$ runs the KeyGen algorithm and gives the normal key to $\mathscr{A}$.
- If $j < k$, then it is of sf-type 3 key. Using $\mathcal{PP}$, $\mathcal{MSK}$ and g_2, $\mathscr{B}$ can generate the required key.
- If $j = k$ then it is either of sf-type 1 or sf-type 2 key. $\mathscr{B}$ runs $(\boldsymbol{k}_{x_k}, m_2) \longleftarrow \mathsf{Enc1}(x_k, N)$ with $|\boldsymbol{k}_{x_k}| = m_1$. Picks $\boldsymbol{r}_k \xleftarrow{\text{U}} \mathbb{Z}_N^{m_2}$ and $\boldsymbol{R}_3 \xleftarrow{\text{U}} \mathbb{G}_{p_3}^{m_1}$. It makes a query with x_k to $\mathcal{CH}$ and let $T := g_2^{\boldsymbol{k}_{x_k}(\beta, \hat{\boldsymbol{r}}_k, \hat{\boldsymbol{h}})}$ be the reply, where $\beta = 0$ or random element from $\mathbb{Z}_N$. Then $\mathscr{B}$ returns the following key
$$\mathcal{SK}_{x_k} := g^{\boldsymbol{k}_{x_k}(\alpha, \boldsymbol{r}_k, \boldsymbol{h})} \cdot T \cdot \boldsymbol{R}_3$$
to $\mathscr{A}$. Therefore, $\mathcal{SK}_{x_j}$ is perfectly distributed sf-type 1 key if $\beta = 0$ else sf-type 2.

– $\mathsf{Sign}(m, x, y)$: If $x \not\sim y$, it returns $\perp$. It is a query for normal signature. $\mathscr{B}$ can answer the query of $\mathscr{A}$ as the $\mathcal{MSK}$ is known to him.

Forgery: $\mathscr{A}$ outputs a signature $\boldsymbol{\delta}_{y^*}$ for (m^*, y^*). Then, $\mathscr{B}$ prepares a vText for (m^*, y^*) as follows: It computes $\hbar^* := H(m^*, y^*)$. Runs $(\boldsymbol{c}_{y^*}, \omega_2) \longleftarrow \mathsf{Enc2}(y^*, N)$ with $|\boldsymbol{c}_{y^*}| = \omega_1$ and sets $\boldsymbol{c}_{y^*}^{\mathsf{M}} := (c_0, \boldsymbol{c}_{y^*})$. Then, picks $\boldsymbol{s} := (s_0, \ldots, s_{\omega_2}) \xleftarrow{\text{U}} \mathbb{Z}_N^{\omega_2+1}$. Then, it makes a query with y^* to $\mathcal{CH}$ and let $D := g_2^{\boldsymbol{c}_{y^*}(\hat{\boldsymbol{s}}, \hat{\boldsymbol{h}})}$ be the reply. Finally, computes a vText as $\mathcal{V} := \big(\mathcal{V}_{\mathsf{INT}} := e(g,g)^{\alpha s_0}, \boldsymbol{\mathcal{V}}_{y^*} := g^{\boldsymbol{c}_{y^*}^{\mathsf{M}}(\boldsymbol{s}, \boldsymbol{h}_{\mathsf{M}})} \cdot g_2^{\boldsymbol{c}_{y^*}^{\mathsf{M}}(\hat{\boldsymbol{s}}, \hat{\boldsymbol{h}}_{\mathsf{M}})}\big)$, where $g_2^{\boldsymbol{c}_{y^*}^{\mathsf{M}}(\hat{\boldsymbol{s}}, \hat{\boldsymbol{h}}_{\mathsf{M}})} := (g_2^{\hat{s}(\theta_1 \hbar^* + \theta_2)}, D)$. $\mathscr{B}$ returns 1 if $e(\boldsymbol{\delta}_{y^*}, \boldsymbol{\mathcal{V}}_{y^*}) = \mathcal{V}_{\mathsf{INT}}$ else 0.

Analysis:

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

– $\mathsf{Perfectness}$: By the assumption: $c_{y^*,1}(\hat{\boldsymbol{s}}, \hat{\boldsymbol{h}}) = \hat{s}_0$, the first component of D is $g_2^{\hat{s}_0}$. So, the first component of $g_2^{\boldsymbol{c}_{y^*}^{\mathsf{M}}(\hat{\boldsymbol{s}}, \hat{\boldsymbol{h}}_{\mathsf{M}})}$ can be computed as $g_2^{\hat{s}_0(\theta_1 \hbar^* + \theta_2)} = (g_2^{\hat{s}_0})^{\theta_1 \hbar^* + \theta_2}$. $\mathscr{B}$ implicitly sets $(\hat{\theta}_1, \hat{\theta}_2) :\equiv (\theta_1, \theta_2)$ mod p_2. By CRT, $(\hat{\theta}_1, \hat{\theta}_2)$ is independent from $(\theta_1, \theta_2) \mod p_1$ and therefore $\mathcal{V}$ is perfectly distributed sf-type 1 vText. Altogether, we have that the joint distribution of all the objects simulated by $\mathscr{B}$ is identical to that of Game_{1-k-1} if $\beta = 0$ else Game_{1-k-2}.

□

Lemma A.5. Game_{1-k-2} *and* Game_{1-k-3} *are indistinguishable under the DSG2 assumption. That is, for every adversary* $\mathscr{A}$*, there exists a PPT algorithm* $\mathscr{B}$ *such that*
$$|\mathsf{Adv}_{\mathscr{A},\text{PS}}^{1-k-2}(\kappa) - \mathsf{Adv}_{\mathscr{A},\text{PS}}^{1-k-3}(\kappa)| \leq \mathsf{Adv}_{\mathscr{B}}^{\text{DSG2}}(\kappa) \quad \textit{for} \quad 1 \leq k \leq \nu_1.$$

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

The k^{th} key is either sf-type 2 or sf-type 3. $\mathscr{B}$ runs $(\boldsymbol{k}_{x_k}, m_2) \longleftarrow \mathsf{Enc1}(x_k, N)$ with $|\boldsymbol{k}_{x_k}| = m_1$. Picks $\boldsymbol{r}'_k, \hat{\boldsymbol{r}}'_k \xleftarrow{\text{U}} \mathbb{Z}_N^{m_2}$ and $\boldsymbol{R}_3 \xleftarrow{\text{U}} \mathbb{G}_{p3}^{m_1}$. It generates the following $\mathcal{SK}_{x_k}$ using T_β of the instance of DSG2:

$$\mathcal{SK}_{x_k} := g^{\boldsymbol{k}_{x_k}(\alpha, \boldsymbol{r}'_k, \boldsymbol{h})} \cdot (W_2W_3)^{\boldsymbol{k}_{x_k}(\alpha'_k, \boldsymbol{0}, \boldsymbol{0})} \cdot T_\beta^{\boldsymbol{k}_{x_j}(0, \hat{\boldsymbol{r}}'_k, \boldsymbol{h})} \cdot \boldsymbol{R}_3.$$

If $W_2W_3 = g_2^{w_2}g_3^{w_3}$ and $T_\beta = g^{t_1}g_2^{t_2}g_3^{t_3}$ (for $\beta = 1$), then $\mathscr{B}$ implicitly sets $\hat{\alpha}_k := w_2\alpha'_k$, $\boldsymbol{r}_k := \boldsymbol{r}'_k + t_1\hat{\boldsymbol{r}}'_k$ and $\hat{\boldsymbol{r}}_k := t_2\hat{\boldsymbol{r}}'_k$. Note that here we use the linearity and param-vanishing properties of the pair encoding, P. Since $\boldsymbol{r}'_k$ and $\hat{\boldsymbol{r}}'_k$ are chosen uniformly and independently from $\mathbb{Z}_N^{m_2}$, then so are $\boldsymbol{r}_k$ and $\hat{\boldsymbol{r}}_k$. Therefore, $\mathcal{SK}_{x_k}$ is perfectly distributed sf-type 2 (resp. sf-type 3) key if $\beta = 1$ (resp. $\beta = 0$).

□

Lemma A.6. $\text{Game}_{2-(k-1)-2}$ *and* Game_{2-k-1} *are indistinguishable under DSG2 assumption and collision resistant property of H. That is, for every adversary $\mathscr{A}$, there exists a PPT algorithm $\mathscr{B}$ such that*

$$|\mathsf{Adv}_{\mathscr{A},\text{PS}}^{2-(k-1)-2}(\kappa) - \mathsf{Adv}_{\mathscr{A},\text{PS}}^{2-k-1}(\kappa)| \leq \mathsf{Adv}_{\mathscr{B}}^{\text{DSG2}}(\kappa) + \mathsf{Adv}_{\mathscr{B}}^{\text{CRH}}(\kappa) \text{ for } 1 \leq k \leq \nu_2.$$

Proof. We establish a PPT simulator $\mathscr{B}$ who receives an instance of DSG2, $(\mathcal{J},\ g,\ Z_1Z_2,\ W_2W_3,\ Z_3,\ T_\beta)$ with $\beta \xleftarrow{\text{U}} \{0,1\}$ and depending on the distribution of β, it simulates either $\text{Game}_{2-(k-1)-2}$ or Game_{2-k-1}.

Setup: Same as Lemma A.3.

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

- $\mathsf{KeyGen}(x)$: Here all the keys are of sf-type 3 and simulation of the keys are same as the sf-type 3 keys of Lemma A.3.
- $\mathsf{Sign}(m, x, y)$: If $x \not\sim y$, it returns $\perp$. Let (m_j, x_j, y_j) be the j^{th} signature query made $\mathscr{A}$. $\mathscr{B}$ answers the signature $\boldsymbol{\delta}_{y_j}$ as follows:
  - If $j > k$, it is normal signature. $\mathscr{B}$ can answer the queries of $\mathscr{A}$ as the $\mathcal{MSK}$ is known to him.
  - If $j < k$, it is sf-type 2 signature. $\mathscr{B}$ first computes the normal signature $\boldsymbol{\delta}_{y_j}$, picks $\iota'_j \xleftarrow{\text{U}} \mathbb{Z}_N$ and then returns
  $$\tilde{\boldsymbol{\delta}}_{y_j} := \boldsymbol{\delta}_{y_j} \cdot (W_2W_3)^{(0,\ \iota'_j,\ 0,\ldots,0)}.$$
  If $W_2W_3 = g_2^{w_2}g_3^{w_3}$, then $\mathscr{B}$ implicitly sets $\iota_j := w_2\iota'_j$. So, $\tilde{\boldsymbol{\delta}}_{y_j}$ is properly distributed sf-type 2 signature.
  - If $j = k$, it is either normal signature or sf-type 1 signature. $\mathscr{B}$ runs $(\boldsymbol{k}_{x_k}, m_2) \longleftarrow \mathsf{Enc1}(x_k, N)$ and $\mathsf{Pair}(x_k, y_k) \longrightarrow \boldsymbol{E} \in \mathbb{Z}_N^{m_1 \times \omega_1}$. Picks $\boldsymbol{v}_{\mathsf{sp}} \xleftarrow{\text{U}} (\mathbf{V}_{\mathsf{M}})^\perp$, $\boldsymbol{r} \xleftarrow{\text{U}} \mathbb{Z}_N^{m_2}$ and $\boldsymbol{R}_3 \xleftarrow{\text{U}} \mathbb{G}_{p3}^{\omega_1+1}$. It computes $\hbar_k := H(m_k, y_k)$ and then returns the following signature:
  $$\boldsymbol{\delta}_{y_k} := g^{(0,\ \boldsymbol{k}_{x_k}(\alpha, \boldsymbol{r}, \boldsymbol{h})\boldsymbol{E})} \cdot g^{\boldsymbol{v}_{\mathsf{sp}}} \cdot T_\beta^{(-1,\ 0,\ldots,0)} \cdot T_\beta^{(0,\ \theta_1\hbar_k + \theta_2,\ldots,0)} \cdot \boldsymbol{R}_3.$$

Let $g^{\tau} := T_\beta\big|_{\mathbb{G}_{p_1}}$ and for $\beta = 1$, $g_2^{t_2} := T_\beta\big|_{\mathbb{G}_{p_2}}$. Then, the $\mathbb{G}_{p_1}$ component of $\boldsymbol{\delta}_{y_k}$ can be written as $g^{\boldsymbol{v}+\boldsymbol{v}_{\mathsf{sp}}}$, where $\boldsymbol{v} := (-\tau, \boldsymbol{\psi} + \boldsymbol{k}_{x_k}(\alpha, \boldsymbol{r}, \boldsymbol{h})\boldsymbol{E})$ and $\boldsymbol{\psi} := (\tau(\theta_1\hbar_k + \theta_2), 0, \ldots, 0)$. If $\beta = 1$, the $\mathbb{G}_{p_2}$ component of $\boldsymbol{\delta}_{y_k}$ is expressed as $g_2^{\hat{\boldsymbol{v}}}$ where $\mathscr{B}$ implicitly sets $b :\equiv -t_2 \mod p_2$ and $\iota :\equiv t_2(\theta_1\hbar_k + \theta_2) \mod p_2$. Since $\theta_1\hbar_k + \theta_2 \mod p_1$ are independent from $\theta_1\hbar_k + \theta_2 \mod p_2$ by CRT, therefore $\boldsymbol{\delta}_{y_k}$ is perfectly distributed signature unless some correlation with vText is found later.

Forgery: $\mathscr{A}$ outputs a signature $\boldsymbol{\delta}_{y^*}$ for (m^*, y^*). Then, $\mathscr{B}$ prepares a vText for (m^*, y^*) as follows: It computes $\hbar^* := H(m^*, y^*)$. Runs $(\boldsymbol{c}_{y^*}, \omega_2) \longleftarrow \mathsf{Enc2}(y^*, N)$ with $|\boldsymbol{c}_{y^*}| = \omega_1$ and sets $\boldsymbol{c}_{y^*}^{\mathsf{M}} := (c_0, \boldsymbol{c}_{y^*})$. Then picks $\boldsymbol{s}' := (s'_0, \ldots, s'_{\omega_2}) \xleftarrow{\text{U}} \mathbb{Z}_N^{\omega_2+1}$. It computes a vText as $\mathcal{V} := (\mathcal{V}_{\mathsf{INT}} := e(g^{\alpha}, Z_1Z_2)^{s'_0}, \boldsymbol{\mathcal{V}}_{y^*} := (Z_1Z_2)^{\boldsymbol{c}_{y^*}^{\mathsf{M}}(\boldsymbol{s}', \boldsymbol{h}_{\mathsf{M}})})$. $\mathscr{B}$ returns 1 if $e(\boldsymbol{\delta}_{y^*}, \boldsymbol{\mathcal{V}}_{y^*}) = \mathcal{V}_{\mathsf{INT}}$ else 0.

Analysis: Now, we mainly concentrate on the joint distribution of k^{th} signature and vText as there may be a correlation between them. More precisely, we observe the distributional relation between $c_0^*(\hat{s}_0, \hat{\boldsymbol{\theta}}) := \hat{s}_0(\hat{\theta}_1\hbar^* + \hat{\theta}_2) :\equiv \tilde{s}_0(\theta_1\hbar^* + \theta_2) \mod p_2$ and $c_{y^*,1}(\hat{\boldsymbol{s}}, \hat{\boldsymbol{h}}) := \hat{s}_0 :\equiv \tilde{s}_0 \mod p_2$ with $\tilde{s}_0 :\equiv z_1 s'_0$ involved in $\boldsymbol{c}_{y^*}^{\mathsf{M}}(\hat{\boldsymbol{s}}, \hat{\boldsymbol{h}}_{\mathsf{M}})$ of vText. Unfortunately, a similar kind of relation is found in $\hat{\boldsymbol{v}}$, viz., between $b :\equiv -t_2 \mod p_2$ and $\iota :\equiv t_2(\theta_1\hbar_j + \theta_2) \mod p_2$. But that correlation does not hamper our life: since H has collision resistant property and $(m_j, y_j) \neq (m^*, y^*)$, we have $\hbar_j \neq \hbar^*$. By applying the argument of [32], we have $\theta_1\hbar_j + \theta_2$ and $\theta_1\hbar^* + \theta_2$ are independently and uniformly distributed[9] over $\mathbb{Z}_{p_2}$. Therefore, $(\tilde{s}_0, \tilde{s}_0(\theta_1\hbar^* + \theta_2)) \mod p_2$ is uncorrelated from (b, ι). Altogether, we have that the joint distribution of all the objects simulated by $\mathscr{B}$ is identical to that of $\text{Game}_{2-(k-1)-2}$ if $\beta = 0$ else Game_{2-k-1}. □

Lemma A.7. Game_{2-k-1} *and* Game_{2-k-2} *are indistinguishable under the DSG2 assumption. That is, for every adversary* $\mathscr{A}$*, there exists a PPT algorithm* $\mathscr{B}$ *such that*

$$|\mathsf{Adv}_{\mathscr{A},\text{PS}}^{2-k-1}(\kappa) - \mathsf{Adv}_{\mathscr{A},\text{PS}}^{2-k-2}(\kappa)| \le \mathsf{Adv}_{\mathscr{B}}^{\text{DSG2}}(\kappa) \quad \textit{for} \quad 1 \le k \le \nu_2.$$

Proof. We establish a PPT simulator $\mathscr{B}$ who receives an instance of DSG2, $(\mathcal{J}, g, Z_1Z_2, W_2W_3, Z_3, T_\beta)$ with $\beta \xleftarrow{\text{U}} \{0, 1\}$ and depending on the distribution of β, it simulates either Game_{2-k-1} or Game_{2-k-2}. The simulation is almost similar to Lemma A.6 except the answering k^{th} signature query. Note that in this case, we do not require the collision resistant property of H. We only illustrate here the k^{th} signature. The k^{th} signature is of either sf-type 1 or sf-type 2. $\mathscr{B}$ runs $(\boldsymbol{k}_{x_k}, m_2) \longleftarrow \mathsf{Enc1}(x_k, N)$ and $\mathsf{Pair}(x_k, y_k) \longrightarrow \boldsymbol{E} \in \mathbb{Z}_N^{m_1 \times \omega_1}$. Picks $\iota'_k \xleftarrow{\text{U}} \mathbb{Z}_N$, $\boldsymbol{v}_{\mathsf{sp}} \xleftarrow{\text{U}} (\mathbf{V}_{\mathsf{M}})^{\perp}$, $\boldsymbol{r} \xleftarrow{\text{U}} \mathbb{Z}_N^{m_2}$ and $\boldsymbol{R}_3 \xleftarrow{\text{U}} \mathbb{G}_{p_3}^{\omega_1+1}$. It computes $\hbar_k := H(m_k, y_k)$ and then returns the signature as given below:

$$\boldsymbol{\delta}_{y_k} := g^{(0,\ \boldsymbol{k}_{x_k}(\alpha, \boldsymbol{r}, \boldsymbol{h})\boldsymbol{E})} \cdot g^{\boldsymbol{v}_{\mathsf{sp}}} \cdot T_\beta^{(-1,\ 0,\ldots,0)} \cdot T_\beta^{(0,\ \theta_1\hbar_k+\theta_2,\ldots,0)} \cdot (W_2W_3)^{(0,\ \iota'_k,\ 0,\ldots,0)} \cdot \boldsymbol{R}_3.$$

Let $W_2W_3 = g_2^{w_2}g_3^{w_3}$. Let $g^{\tau} := T_\beta\big|_{\mathbb{G}_{p_1}}$ and for $\beta = 1$, $g_2^{t_2} := T_\beta\big|_{\mathbb{G}_{p_2}}$. Then, the $\mathbb{G}_{p_1}$ component of $\boldsymbol{\delta}_{y_k}$ can be written as $g^{\boldsymbol{v}+\boldsymbol{v}_{\mathsf{sp}}}$, where $\boldsymbol{v} := (-\tau, \boldsymbol{\psi} + \boldsymbol{k}_{x_k}(\alpha, \boldsymbol{r}, \boldsymbol{h})\boldsymbol{E})$ and $\boldsymbol{\psi} := (\tau(\theta_1\hbar_k + \theta_2), 0, \ldots, 0)$. If $\beta = 1$ (resp. $\beta = 0$), the $\mathbb{G}_{p_2}$ component of $\boldsymbol{\delta}_{y_k}$ is expressed as $g_2^{\hat{\boldsymbol{v}}}$, with $\hat{\boldsymbol{v}} := (b, \iota, 0, \ldots, 0) \in \mathbb{Z}_N^{\omega_1+1}$ where $\mathscr{B}$ implicitly sets $b :\equiv -t_2 \mod p_2$ (resp. $b :\equiv 0 \mod p_2$) and $\iota :\equiv t_2(\theta_1\hbar_k + \theta_2) + w_2\iota'_k \mod p_2$ (resp. $\iota :\equiv w_2\iota'_k \mod p_2$). Therefore, $\boldsymbol{\delta}_{y_k}$ is perfectly distributed sf-type 1 (resp. sf-type 2) signature if $\beta = 1$ (resp. $\beta = 0$). □

[9] To apply [32], we require that $\hbar_j - \hbar^* \not\equiv 0 \mod p_2$. From $\hbar_j - \hbar^* \not\equiv 0 \mod N$, we have $\hbar_j - \hbar^* \not\equiv 0 \mod p$ for at least one p such that $p \in \{p_1, p_2, p_3\}$. One can show that $\hbar_j - \hbar^* \not\equiv 0 \mod p$ for all p with $p \in \{p_1, p_2, p_3\}$ assuming factorization problem is hard. However, if $\hbar_j - \hbar^* \equiv 0 \mod p_2$ we can find a factor F of N with $p_2|F$ and which leads to break the DSG2 assumption, a contradiction.

Lemma A.8. Game$_{2-\nu_2-2}$ *and* Game$_{Final}$ *are indistinguishable under the DSG3 assumption. That is, for every adversary $\mathscr{A}$, there exists a PPT algorithm $\mathscr{B}$ such that*

$$|\mathsf{Adv}^{2-\nu_2-2}_{\mathscr{A},\mathrm{PS}}(\kappa) - \mathsf{Adv}^{Final}_{\mathscr{A},\mathrm{PS}}(\kappa)| \leq \mathsf{Adv}^{\mathrm{DSG3}}_{\mathscr{B}}(\kappa).$$

Proof. We establish a PPT simulator $\mathscr{B}$ who receives an instance of DSG1, $(\mathcal{J}, g, g^{\alpha}Y_2, g^{s_0}W_2, g_2, Z_3, T_\beta)$ with $\beta \xleftarrow{\mathrm{U}} \{0,1\}$ and depending on the distribution of β, it simulates either Game$_{2-\nu_2-2}$ or Game$_{Final}$.

Setup: $\mathscr{B}$ chooses $\theta_1, \theta_2 \xleftarrow{\mathrm{U}} \mathbb{Z}_N$, $\boldsymbol{h} \xleftarrow{\mathrm{U}} \mathbb{Z}_N^n$ and sets $\boldsymbol{h}_{\mathsf{M}} := (\theta_1, \theta_2, \boldsymbol{h})$. Let $H : \{0,1\}^* \longrightarrow \mathbb{Z}_N$ be a hash function. Then, it provides $\mathcal{PP} := (\mathcal{J}, g, g^{\boldsymbol{h}_{\mathsf{M}}}, g_T^{\alpha} := e(g, g^{\alpha}Y_2), Z_3, H)$ to $\mathscr{A}$. Implicitly sets $\hat{\boldsymbol{h}}_{\mathsf{M}} :\equiv \boldsymbol{h}_{\mathsf{M}}$ mod p_2. By Chinese Remainder Theorem (CRT), $\hat{\boldsymbol{h}}_{\mathsf{M}}$ is independent from $\boldsymbol{h}_{\mathsf{M}}$ mod p_1 and so $\hat{\boldsymbol{h}}_{\mathsf{M}}$ is perfectly distributed.

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

- $\mathsf{KeyGen}(x)$: It is sf-type 3 key. $\mathscr{B}$ runs $(\boldsymbol{k}_x, m_2) \longleftarrow \mathsf{Enc1}(x)$. Then picks $\boldsymbol{r} \xleftarrow{\mathrm{U}} \mathbb{Z}_N^{m_2}$, $\hat{\alpha}' \xleftarrow{\mathrm{U}} \mathbb{Z}_N$ and $\boldsymbol{R}_3 \xleftarrow{\mathrm{U}} \mathbb{G}_{p_3}^{m_1}$. Finally it returns

$$\mathcal{SK}_x := (g^{\alpha}Y_2)^{\boldsymbol{k}_x(1,\boldsymbol{0},\boldsymbol{0})} \cdot g^{\boldsymbol{k}_x(0,\boldsymbol{r},\boldsymbol{h})} \cdot g_2^{\boldsymbol{k}_x(\hat{\alpha}',\boldsymbol{0},\boldsymbol{0})} \cdot \boldsymbol{R}_3.$$

  If $Y_2 = g_2^{y_2}$, $\mathscr{B}$ implicitly sets $\hat{\alpha} := y_2 + \hat{\alpha}'$ mod p_2 and so, $\mathcal{SK}_x$ is a perfectly distributed sf-type 3 key.

- $\mathsf{Sign}(m, x, y)$: If $x \not\sim y$, it returns $\perp$. It is a query for sf-type 2 signature. $\mathscr{B}$ first creates sf-type 3 key $\mathcal{SK}_x$ and then using $\mathcal{SK}_x$, it can compute the sf-type 2 signature as described in **Remark 4.2**.

Forgery: $\mathscr{A}$ outputs a signature $\boldsymbol{\delta}_{y^*}$ for (m^*, y^*). Then, $\mathscr{B}$ prepares a vText for (m^*, y^*) as follows: It computes $\hbar^* := H(m^*, y^*)$. Runs $(\boldsymbol{c}_{y^*}, \omega_2) \longleftarrow \mathsf{Enc2}(y^*, N)$ with $|\boldsymbol{c}_{y^*}| = \omega_1$ and sets $\boldsymbol{c}^{\mathsf{M}}_{y^*} := (c_0, \boldsymbol{c}_{y^*})$. It picks $(s'_1, \ldots, s'_{\omega_2}) \xleftarrow{\mathrm{U}} \mathbb{Z}_N^{\omega_2}$ and sets $\boldsymbol{s}' := (1, s'_1, \ldots, s'_{\omega_2}) \in \mathbb{Z}_N^{\omega_2+1}$. Finally, computes a vText as $\mathcal{V} := (\mathcal{V}_{\mathsf{INT}} := T_\beta, \boldsymbol{\mathcal{V}}_{y^*} := (g^{s_0}W_2)^{\boldsymbol{c}^{\mathsf{M}}_{y^*}(\boldsymbol{s}', \boldsymbol{h}_{\mathsf{M}})})$. $\mathscr{B}$ returns 1 if $e(\boldsymbol{\delta}_{y^*}, \boldsymbol{\mathcal{V}}_{y^*}) = \mathcal{V}_{\mathsf{INT}}$ else 0.

$\mathscr{B}$ implicitly sets $\boldsymbol{s} :\equiv s_0\boldsymbol{s}'$ mod p_1 and $\hat{\boldsymbol{s}} :\equiv s_0\boldsymbol{s}'$ mod p_2. By CRT, $\boldsymbol{s}'$ mod p_1 is independent from $\boldsymbol{s}'$ mod p_2 and so, $\boldsymbol{s}$ and $\hat{\boldsymbol{s}}$ are perfectly distributed as required. Therefore, $\mathcal{V}$ is perfectly distributed sf-type 1 vText if $\beta = 0$ else sf-type 2 vText.

Analysis: All the components simulated above are perfectly distributed as required. Therefore, the joint distribution of all the objects simulated by $\mathscr{B}$ is identical to that of Game$_{2-\nu_2-2}$ if $\beta = 0$ else Game$_{Final}$. □

B Lemmas Used in the Proof of Theorem 6.1

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

$$|\mathsf{Adv}^{\mathrm{Real}}_{\mathscr{A},\mathrm{PE}}(\kappa) - \mathsf{Adv}^{\mathrm{Res}}_{\mathscr{A},\mathrm{PE}}(\kappa)| \leq \mathsf{Adv}^{\mathrm{DSG2}}_{\mathscr{B}}(\kappa).$$

Proof. The proof follows from Lemma 27 of [2] or Lemma A.1 in this paper. □

Lemma B.2. Game$_{Res}$ *and* Game$_0$ *are indistinguishable under the DSG1 assumption. That is, for every adversary* $\mathscr{A}$*, there exists a PPT algorithm* $\mathscr{B}$ *such that*

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

Proof. The proof is similar to Lemma 28 of [2] and Lemma A.2 in this paper. □

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

$$|\mathsf{Adv}^{1-(k-1)-3}_{\mathscr{A},\text{PE}}(\kappa) - \mathsf{Adv}^{1-k-1}_{\mathscr{A},\text{PE}}(\kappa)| \leq \mathsf{Adv}^{\text{DSG2}}_{\mathscr{B}}(\kappa)$$

for $1 \leq k \leq q_1$*, where* q_1 *is the number of pre-challenged key queries.*

Proof. For proof, refer to the proof of Lemma 29 of [2] and Lemma A.3 in this paper. □

Lemma B.4. 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* $\mathscr{A}$*, there exists a PPT algorithm* $\mathscr{B}$ *such that*

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

Proof. Following the proof of Lemma 30 of [2] and Lemma A.4 in this paper, it can be proven. Note that condition (1) of **Conditions 3.1** will be used here. □

Lemma B.5. Game$_{1-k-2}$ *and* Game$_{1-k-3}$ *are indistinguishable under the DSG2 assumption. That is, for every adversary* $\mathscr{A}$*, there exists a PPT algorithm* $\mathscr{B}$ *such that*

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

Proof. The proof is similar to that of Lemma 31 of [2] and Lemma A.5 in this paper. □

Lemma B.6. Game$_{1-q_1-3}$ *and* Game$_{1-(q_1+1)-1}$ *are indistinguishable under the DSG2 assumption. That is, for every adversary* $\mathscr{A}$*, there exists a PPT algorithm* $\mathscr{B}$ *such that*

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

Proof. For proof, we refer to Lemma 32 of [2]. □

Lemma B.7. 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* $\mathscr{A}$*, there exists a PPT algorithm* $\mathscr{B}$ *such that*

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

Proof. The proof is similar to Lemma 33 of [2]. Note that condition (1) of **Conditions 3.1** is applied here. □

Lemma B.8. Game$_{1-(q_1+1)-2}$ *and* Game$_{1-(q_1+1)-3}$ *are indistinguishable under the DSG2 assumption. That is, for every adversary* $\mathscr{A}$*, there exists a PPT algorithm* $\mathscr{B}$ *such that*

$$|\mathsf{Adv}^{1-(q_1+1)-2}_{\mathscr{A},\text{PE}}(\kappa) - \mathsf{Adv}^{1-(q_1+1)-3}_{\mathscr{A},\text{PE}}(\kappa)| \leq \mathsf{Adv}^{\text{DSG2}}_{\mathscr{B}}(\kappa).$$

Proof. The proof can be done in similar manner as in Lemma 34 of [2]. □

Lemma B.9. 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 $\mathscr{A}$, there exists a PPT algorithm $\mathscr{B}$ such that*

$$|\mathsf{Adv}_{\mathscr{A},\mathrm{PE}}^{2-(k-1)-2}(\kappa) - \mathsf{Adv}_{\mathscr{A},\mathrm{PE}}^{2-k-1}(\kappa)| \leq \mathsf{Adv}_{\mathscr{B}}^{\mathrm{DSG2}}(\kappa) + \mathsf{Adv}_{\mathscr{B}}^{\mathrm{CRH}}(\kappa) \ \ \textit{for} \ \ 1 \leq k \leq \nu.$$

Proof. We establish a PPT simulator $\mathscr{B}$ who receives an instance of DSG2, $(\mathcal{J},\ g,\ Z_1Z_2,\ W_2W_3,\ Z_3,\ T_\beta)$ with $\beta \xleftarrow{\mathrm{U}} \{0,1\}$ and depending on the distribution of β, it simulates either Game$_{2-(k-1)-2}$ or Game$_{2-k-1}$.

Setup: $\mathscr{B}$ chooses $\alpha, \theta_1, \theta_2 \xleftarrow{\mathrm{U}} \mathbb{Z}_N$, $\boldsymbol{h} \xleftarrow{\mathrm{U}} \mathbb{Z}_N^n$ and sets $\boldsymbol{h}_{\mathsf{M}} := (\theta_1, \theta_2, \boldsymbol{h})$. Let $H : \{0,1\}^* \longrightarrow \mathbb{Z}_N$ be a hash function. Then, it provides $\mathcal{PP} := [\mathcal{J}, g, g^{\boldsymbol{h}_{\mathsf{M}}}, g_T^{\alpha} := e(g,g)^{\alpha}, Z_3, H]$ to $\mathscr{A}$ and keeps $\mathcal{MSK} := (\alpha)$ to itself. Implicitly sets $\hat{\boldsymbol{h}}_{\mathsf{M}} :\equiv \boldsymbol{h}_{\mathsf{M}} \mod p_2$. By CRT, $\hat{\boldsymbol{h}}_{\mathsf{M}}$ is independent from $\boldsymbol{h}_{\mathsf{M}} \mod p_1$ and so $\hat{\boldsymbol{h}}_{\mathsf{M}}$ is perfectly distributed.

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

- $\mathsf{KeyGen}(x)$: Here all the keys are of sf-type 3 and simulation of the keys are same as the sf-type 3 keys of Lemma B.3.
- $\mathsf{Decrypt}(\mathsf{CT}, x)$: Let (CT_j, x_j) be the j^{th} decrypt query made $\mathscr{A}$. $\mathscr{B}$ first constructs the alt-key $\mathcal{SK}_{x_j}^{\mathsf{M}}$ as shown below and then answers to $\mathscr{A}$ by running $\mathsf{AltDecrypt}$ algorithm :
  - If $j > k$, it is normal alt-key. $\mathscr{B}$ can compute the key as the $\mathcal{MSK}$ is known to him.
  - If $j < k$, it is sf-type 2 alt-key. $\mathscr{B}$ first computes the normal alt-key $\mathcal{SK}_{x_j}^{\mathsf{M}}$, picks $\iota_j' \xleftarrow{\mathrm{U}} \mathbb{Z}_N$ and then creates the sf-type 2 alt-key as follows:

    $$\tilde{\mathcal{SK}}_{x_j}^{\mathsf{M}} := \mathcal{SK}_{x_j}^{\mathsf{M}} \cdot (W_2W_3)^{(0,\ \iota_j',\ 0,\ldots,0)}.$$

    If $W_2W_3 = g_2^{w_2} g_3^{w_3}$, then $\mathscr{B}$ implicitly sets $\iota_j := w_2 \iota_j'$. So, $\tilde{\mathcal{SK}}_{x_j}^{\mathsf{M}}$ is properly distributed sf-type 2 alt-key.
  - If $j = k$, it is either normal or sf-type 1 alt-key. $\mathscr{B}$ runs $(\boldsymbol{k}_{x_k}, m_2) \longleftarrow \mathsf{Enc1}(x_k, N)$ and $\mathsf{Pair}(x_k, y_k) \longrightarrow \boldsymbol{E} \in \mathbb{Z}_N^{m_1 \times \omega_1}$. Picks $\boldsymbol{r} \xleftarrow{\mathrm{U}} \mathbb{Z}_N^{m_2}$ and $\boldsymbol{R}_3 \xleftarrow{\mathrm{U}} \mathbb{G}_{p_3}^{\omega_1+1}$. It computes the alt-key as given below:

    $$\mathcal{SK}_{x_k}^{\mathsf{M}} := g^{(0,\ \boldsymbol{k}_{x_k}(\alpha,\boldsymbol{r},\boldsymbol{h})\boldsymbol{E})} \cdot T_\beta^{(-1,\ 0,\ldots,0)} \cdot T_\beta^{(0,\ \theta_1\hbar_k+\theta_2,\ldots,0)} \cdot \boldsymbol{R}_3.$$

    where $\hbar_k := H(C_{\mathsf{cpa}}^{(k)})$. Let $g^{\tau} := T_\beta\big|_{\mathbb{G}_{p_1}}$ and for $\beta = 1$, $g_2^{t_2} := T_\beta\big|_{\mathbb{G}_{p_1}}$. Then, $\mathscr{B}$ sets $g^{\boldsymbol{v}} := \mathcal{SK}_{x_k}^{\mathsf{M}}\big|_{\mathbb{G}_{p_1}}$, where $\boldsymbol{v} := (-\tau, \boldsymbol{\psi} + \boldsymbol{k}_{x_k}(\alpha, \boldsymbol{r}, \boldsymbol{h})\boldsymbol{E})$ and $\boldsymbol{\psi} := (\tau(\theta_1\hbar_k + \theta_2), 0, \ldots, 0)$. If $\beta = 1$, it sets $g_2^{\hat{\boldsymbol{v}}} := \mathcal{SK}_{x_k}^{\mathsf{M}}\big|_{\mathbb{G}_{p_2}}$ with $\hat{\boldsymbol{v}} := (b, \iota, 0, \ldots, 0) \in \mathbb{Z}_N^{\omega_1+1}$, where $\mathscr{B}$ implicitly sets $b :\equiv -t_2 \mod p_2$ and $\iota :\equiv t_2(\theta_1\hbar_k + \theta_2) \mod p_2$. Therefore, $\mathcal{SK}_{x_k}^{\mathsf{M}}$ is perfectly distributed normal (resp. sf-type 1) alt-key if $\beta = 0$ (resp. $\beta = 1$).

Challenge Phase: $\mathscr{A}$ provides two equal length messages m_0, m_1 and the challenge index y^* to $\mathscr{B}$. Then, $\mathscr{B}$ picks $b \xleftarrow{\mathrm{U}} \{0,1\}$. Runs $(\boldsymbol{c}_{y^*}, \omega_2) \longleftarrow \mathsf{Enc2}(y^*, N)$ with $|\boldsymbol{c}_{y^*}| = \omega_1$ and picks $\boldsymbol{s}' := (s_0', \ldots, s_{\omega_2}') \xleftarrow{\mathrm{U}} \mathbb{Z}_N^{\omega_2+1}$. It first computes $C_{\mathsf{cpa}}^* := (y^*, \boldsymbol{C}_{y^*} := (Z_1Z_2)^{\boldsymbol{c}_{y^*}(\boldsymbol{s}',\boldsymbol{h})}, C_{\mathsf{INT}} := m_b \cdot e(g^{\alpha}, Z_1Z_2)^{s_0'})$ and then computes $\hbar^* := H(C_{\mathsf{cpa}}^*)$. Finally, returns the challenge ciphertext $\mathsf{CT}^* := (y^*, \boldsymbol{C}_{y^*}^{\mathsf{M}} := (C_0^*, \boldsymbol{C}_{y^*}),\ C_{\mathsf{INT}})$, where $C_0^* := (Z_1Z_2)^{s_0'(\theta_1\hbar^*+\theta_2)}$. If $Z_1Z_2 = g^{z_1}g_2^{z_2}$, it implicitly sets $\boldsymbol{s} := z_1\boldsymbol{s}' \mod p_1$ and $\hat{\boldsymbol{s}} := z_2\boldsymbol{s}' \mod p_2$. Since, $\boldsymbol{s}' \mod p_1$ is independent from $\boldsymbol{s}' \mod p_2$, therefore CT^* is perfectly distributed sf-type 1 challenge ciphertext.

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

- $\mathsf{KeyGen}(x)$: Again note that for all the post key queries x, $\hat{\alpha}$'s appearing in $\mathbb{G}_{p_2}$ components of $\mathcal{SK}_x$ are identical. $\mathscr{B}$ picks $\alpha' \xleftarrow{\text{U}} \mathbb{Z}_N$. Let x_j be the j^{th} ($j > q_1$) query key-index. $\mathscr{B}$ runs $(\boldsymbol{k}_{x_j}, m_2) \longleftarrow \mathsf{Enc1}(x_j, N)$ with $|\boldsymbol{k}_{x_j}| = m_1$. It chooses $\boldsymbol{r}_j \xleftarrow{\text{U}} \mathbb{Z}_N^{m_2}$ and $\boldsymbol{R}_3 \xleftarrow{\text{U}} \mathbb{G}_{p_3}^{m_1}$. It computes $\mathcal{SK}_{x_j}$ as defined below:

$$\mathcal{SK}_{x_j} := g^{\boldsymbol{k}_{x_j}(\alpha, \boldsymbol{r}_j, \boldsymbol{h})} \cdot (W_2W_3)^{\boldsymbol{k}_{x_j}(\alpha', \boldsymbol{0}, \boldsymbol{0})} \cdot \boldsymbol{R}_3.$$

It implicitly sets $\hat{\alpha} := w_2\alpha'$, where $W_2W_3 = g_2^{w_2}g_3^{w_3}$. So, $\mathcal{SK}_{x_j}$ is properly distributed sf-type 3 key.

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

Guess: $\mathscr{A}$ sends a guess b' to $\mathscr{B}$. If $b' = b$ then $\mathscr{B}$ returns 1 else 0.

Analysis: By the natural restriction of the security game, $\mathscr{A}$ is allowed to decrypt query (CT_k, x_k) if $\mathsf{CT}^* \neq \mathsf{CT}_k$. The analysis part is similar to that of Theorem A.6. For this analysis, we have to make sure that $\hbar^* \neq \hbar_k$. In fact, if $\hbar^* = \hbar_k$, then $\mathscr{B}$ aborts the game in query-2 phase. Therefore, we only have to show that the probability of abort is negligible. By Fact B.1, 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 $\mathscr{B}$ is identical to that of $\text{Game}_{2-(k-1)-2}$ if $\beta = 0$ else Game_{2-k-1}.

Fact B.1. If for the k^{th} decrypt query (CT_k, x_k), $\mathsf{CT}^* \neq \mathsf{CT}_k$ and $\hbar^* = \hbar_k$, then $\mathscr{B}$ can solve the given instance of DSG2 assumption.

Proof of Fact B.1. We start with the following equalities:

$$\boxed{\mathsf{CT}^* \neq \mathsf{CT}_k \quad \text{and} \quad \hbar^* = \hbar_k} \tag{5}$$

Since, H is a collision resistant hash function, from the equation (5), we have

$$\boxed{C_0^* \neq C_0^{(k)} \quad \text{and} \quad C_{\mathsf{cpa}}^* = C_{\mathsf{cpa}}^{(k)}} \tag{6}$$

From the definition of $\mathsf{AltDecrypt}$, we have the following equations:

$$\boxed{C_0^{(k)}|_{\mathbb{G}_{p_3}} = 1_{\mathbb{G}} \quad \text{and } e(g, C_0^{(k)}) = e(g^{\theta_1\hbar_k + \theta_2}, C_1^{(k)})} \tag{7}$$

From the challenge ciphertext, we have

$$\boxed{C_0^*|_{\mathbb{G}_{p_3}} = 1_{\mathbb{G}} \quad \text{and } e(g, C_0^*) = e(g^{\theta_1\hbar^* + \theta_2}, C_1^*)} \tag{8}$$

Using the 2nd part of the equations (5), (6) (viz., $C_1^* = C_1^{(k)}$), (7) and (8), we have $e(g, C_0^*) = e(g, C_0^{(k)})$ which in turn implies that

$$\boxed{C_0^*|_{\mathbb{G}_{p_1}} = C_0^{(k)}|_{\mathbb{G}_{p_1}}} \tag{9}$$

Since $C_0^{(k)}|_{\mathbb{G}_{p_3}} = 1_{\mathbb{G}}$, $C_0^*|_{\mathbb{G}_{p_3}} = 1_{\mathbb{G}}$, using equation (9), 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 $Y_2 \neq 1_{\mathbb{G}}$. Therefore, $\mathscr{B}$ can break the given instance of DSG2 assumption using Y_2. □

Lemma B.10. Game$_{2-k-1}$ *and* Game$_{2-k-2}$ *are indistinguishable under the DSG2 assumption. That is, for every adversary $\mathscr{A}$, there exists a PPT algorithm $\mathscr{B}$ such that*

$$|\mathsf{Adv}^{2-k-1}_{\mathscr{A},\mathrm{PE}}(\kappa) - \mathsf{Adv}^{2-k-2}_{\mathscr{A},\mathrm{PE}}(\kappa)| \leq \mathsf{Adv}^{\mathrm{DSG2}}_{\mathscr{B}}(\kappa) \;\; \textit{for} \;\; 1 \leq k \leq \nu.$$

Proof. We establish a PPT simulator $\mathscr{B}$ who receives an instance of DSG2, $(\mathcal{J},\, g,\, Z_1Z_2,\, W_2W_3,\, Z_3,\, T_\beta)$ with $\beta \xleftarrow{\mathrm{U}} \{0,1\}$ and depending on the distribution of β, it simulates either Game$_{2-k-1}$ or Game$_{2-k-2}$. The simulation is almost similar to Lemma B.9 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. $\mathscr{B}$ runs $(\boldsymbol{k}_{x_k}, m_2) \longleftarrow \mathsf{Enc1}(x_k, N)$ and $\mathsf{Pair}(x_k, y_k) \longrightarrow \boldsymbol{E} \in \mathbb{Z}_N^{m_1 \times \omega_1}$. Picks $\iota'_k \xleftarrow{\mathrm{U}} \mathbb{Z}_N$ and $\boldsymbol{R}_3 \xleftarrow{\mathrm{U}} \mathbb{G}_{p_3}^{\omega_1+1}$. It computes the alt-key as given below:

$$\mathcal{SK}^{\mathsf{M}}_{x_k} := g^{(0,\ \boldsymbol{k}_{x_k}(\alpha,\boldsymbol{r},\boldsymbol{h})\boldsymbol{E})} \cdot T_\beta^{(-1,\ 0,\ldots,0)} \cdot T_\beta^{(0,\theta_1\hbar_k+\theta_2,\ \ldots,0)} \cdot (W_2W_3)^{(0,\ \iota'_k,\ 0,\ldots,0)} \cdot \boldsymbol{R}_3,$$

where $\hbar_k := H(C^{(k)}_{\mathsf{cpa}})$.

Let $W_2W_3 = g_2^{w_2} g_3^{w_3}$. Let $g^\tau := T_\beta|_{\mathbb{G}_{p_1}}$ and for $\beta = 1$, $g_2^{t_2} := T_\beta|_{\mathbb{G}_{p_1}}$. Then, $\mathscr{B}$ sets $g^{\boldsymbol{v}} = \mathcal{SK}^{\mathsf{M}}_{x_k}|_{\mathbb{G}_{p_1}}$, where $\boldsymbol{v} := (-\tau, \boldsymbol{\psi} + \boldsymbol{k}_{x_k}(\alpha, \boldsymbol{r}, \boldsymbol{h})\boldsymbol{E})$ and $\boldsymbol{\psi} := (\tau(\theta_1\hbar_k + \theta_2), 0, \ldots, 0)$. If $\beta = 1$ (resp. $\beta = 0$), it sets $g_2^{\hat{\boldsymbol{v}}} := \mathcal{SK}^{\mathsf{M}}_{x_k}|_{\mathbb{G}_{p_2}}$ with $\hat{\boldsymbol{v}} := (b, \iota, 0, \ldots, 0) \in \mathbb{Z}_N^{\omega_1+1}$, where $\mathscr{B}$ implicitly sets $b :\equiv -t_2 \mod p_2$ (resp. $b :\equiv 0 \mod p_2$) and $\iota :\equiv t_2(\theta_1\hbar_k + \theta_2) + w_2\iota'_k \mod p_2$ (resp. $\iota :\equiv w_2\iota'_k \mod p_2$). Therefore, $\mathcal{SK}^{\mathsf{M}}_{x_k}$ is perfectly distributed sf-type 1 (resp. sf-type 2) alt-key if $\beta = 1$ (resp. $\beta = 0$). □

Lemma B.11. Game$_{2-\nu-2}$ *and* Game$_{Final}$ *are indistinguishable under the DSG3 assumption. That is, for every adversary $\mathscr{A}$, there exists a PPT algorithm $\mathscr{B}$ such that*

$$|\mathsf{Adv}^{2-\nu}_{\mathscr{A},\mathrm{PE}}(\kappa) - \mathsf{Adv}^{Final}_{\mathscr{A},\mathrm{PE}}(\kappa)| \leq \mathsf{Adv}^{\mathrm{DSG3}}_{\mathscr{B}}(\kappa).$$

Proof. For proof, we refer to Lemma 35 of [2] and Lemma A.8 in this paper. Note that condition (3) of **Conditions 3.1** is applied here. □

C Semi-Functional Objects for Confidentiality and Unforgeability

To obtain confidentiality and unforgeability in adaptive-predicates models, we use the dual system proof technique [50], but its signcryption version as used in [43]. We consider two kinds of signcryptions, the replied signcryption (obtained from signcrypt oracle) and challenge signcryption. The former has three forms, N (stands for normal), sf-type I and sf-type II, whereas the later is of five forms, N, sf-type 1, sf-type 2, sf-type 3 and sf-type 4. For simplicity, we ignore the one-time signature and write signcryption := (signature $(\boldsymbol{\delta}_{y_s})$, ciphertext (CT)). We also consider a new object, called verification text key (in short $\mathsf{vTextKey}$) which is composed of alt-key and vText, i.e., better to write vTextKey := (alt-key, vText). This vTextKey will be used to unsigncrypt the signcryption (queried through the unsigncrypt oracle) using a new algorithm, $\mathsf{AltUnsigncrypt}$ described below. Similar to the forms of signcryption, we consider two kinds of vTextKeys, one is used to answer the unsigncrypt oracle queries and other to unsigncrypt the forgery signcryption. The former has three forms, N, sf-type I and sf-type II, whereas the later is of five forms, N, sf-type 1, sf-type 2, sf-type 3 and sf-type 4. The signcrypt (resp. unsigncrypt) is performed by running the two algorithms, Sign (resp. Ver)and $\mathsf{Encrypt}$ (resp. $\mathsf{Decrypt}$) almost in black-box manner. We describe the different forms of signcryptions (resp. vTextKeys) through already defined different forms of signatures (resp. alt-keys) and ciphertexts (resp. vTexts). For this purpose, we define a (type converter) function $f_{\mathsf{convrt}} : \{\mathsf{N}, \mathsf{I}, \mathsf{II}, 1, 2, 3, 4\} \to \{(i, j) \mid i, j \in \{\mathsf{N}, 1, 2\}\}$, which takes the type of a signcryption (resp. vTextKey) as an input and outputs a pair (i, j), where i is the form of signature (resp. alt-key) and j is the form of ciphertext (resp. vText). The function f_{convrt} is completely defined by the image as $f_{\mathsf{convrt}}(\mathsf{N}) := (\mathsf{N}, \mathsf{N})$, $f_{\mathsf{convrt}}(\mathsf{I}) := (1, \mathsf{N})$, $f_{\mathsf{convrt}}(\mathsf{II}) := (2, \mathsf{N})$, $f_{\mathsf{convrt}}(1) := (\mathsf{N}, 1)$, $f_{\mathsf{convrt}}(2) := (1, 1)$, $f_{\mathsf{convrt}}(3) := (2, 1)$ and $f_{\mathsf{convrt}}(4) := (2, 2)$. From the description of f_{convrt}, the form of ciphertexts (resp. vTexts) in the signcryptions (resp. vTextKeys) of sf-type I and sf-type II are always normal. In the following, we define various algorithms either to generate the semi-functional objects or answer the oracle queries.

- $\mathsf{SFSetup}(1^\kappa, \boldsymbol{j})$: Same as the $\mathsf{SFSetup}$ in Section 4.2.
- $\mathsf{SFKeyGen}(\mathcal{PP}, \mathcal{MSK}, x, g_2, \mathsf{type}, \hat{\boldsymbol{h}})$: Same as the $\mathsf{SFKeyGen}$ in Section 4.2.
- $\mathsf{SFEncrypt}(\mathcal{PP}, m, y, g_2, \mathsf{type}, \hat{\boldsymbol{h}}_{\mathsf{M}})$: Same as the $\mathsf{SFEncrypt}$ in Section 6.2.
- $\mathsf{SFSign}(\mathcal{PP}, m, \mathcal{SK}_x, y, g_2, \mathsf{type})$: Same as the SFSign in Section 4.2.
- $\mathsf{SFSigncrypt}(\mathcal{PP}, m, \mathcal{SK}_x, y_s, y_e, g_2, \mathsf{type}, \hat{\boldsymbol{h}}_{\mathsf{M}})$: If $x \not\sim y$, returns $\perp$. It first runs $(\mathsf{com}, \mathsf{decom}) \longleftarrow \mathsf{Commit}(m)$ and $(\mathsf{vk}, \mathsf{signk}) \longleftarrow \mathsf{OTS.Gen}(1^\kappa)$. Suppose $f_{\mathsf{convrt}}(\mathsf{type}) = (i, j)$. It then runs $\boldsymbol{\delta}_{y_s} \longleftarrow \mathsf{SFSign}(\mathcal{PP}, \mathsf{vk}, \mathcal{SK}_x, y_s, g_2, i)$ and $(C_{\mathsf{cpa}}, C_0) \longleftarrow \mathsf{SFEncrypt}(\mathcal{PP}, \mathsf{decom}, y, g_2, j, \hat{\boldsymbol{h}}_{\mathsf{M}})$, where $C_0 := g^{s_0(\theta_1 \hbar_e + \theta_2)}$ and $\hbar_e := H(0, \mathsf{com}, \boldsymbol{\delta}_{y_s}, \mathsf{vk}, C_{\mathsf{cpa}})$. It runs $\delta_o \longleftarrow \mathsf{OTS.Sign}(C_0 || y_s,\ \mathsf{signk})$ and returns the semi-functional signcryption $\mathsf{U} := (\mathsf{com}, \delta := (\boldsymbol{\delta}_{y_s}, \delta_o, \mathsf{vk}), \mathsf{CT} := (C_{\mathsf{cpa}}, C_0))$
- $\mathsf{SFAltKeyGen}(\mathcal{PP}, \mathcal{MSK}, \mathsf{CT}, x, g_2, \mathsf{type})$: Same as the $\mathsf{SFAltKeyGen}$ in Section 6.2.
- $\mathsf{SFVText}(\mathcal{PP}, m, y, g_2, \mathsf{type}, \hat{\boldsymbol{h}}_{\mathsf{M}})$: Same as the $\mathsf{SFVText}$ in Section 4.2.
- $\mathsf{SFVTextKey}(\mathcal{PP}, \mathcal{MSK}, \mathsf{U}, x, y_s, g_2, \mathsf{type}, \hat{\boldsymbol{h}}_{\mathsf{M}})$: Runs $\mathcal{SK}_x^{\mathsf{M}} \longleftarrow \mathsf{SFAltKeyGen}(\mathcal{PP},\ \mathcal{MSK},\ \mathsf{CT},\ x,\ g_2,\ i)$ and $\mathcal{V} \longleftarrow \mathsf{SFVText}(\mathcal{PP}, \mathsf{vk}, y_s, g_2, j, \hat{\boldsymbol{h}}_{\mathsf{M}})$, where $(i, j) = f_{\mathsf{convrt}}(\mathsf{type})$. It returns the semi-functional vTextKey, $\mathcal{VK} := (\mathcal{SK}_x^{\mathsf{M}}, \mathcal{V})$.
- $\mathsf{AltDecrypt}(\mathcal{PP}, \mathsf{CT}, \mathcal{SK}_x^{\mathsf{M}})$: Same as the $\mathsf{AltDecrypt}$ in Section 6.2.
- $\mathsf{AltVer}(\boldsymbol{\delta}_{y_s}, \boldsymbol{\mathcal{V}})$: This is same as Ver algorithm, but we do not require to compute the vText as it is supplied. Let $\boldsymbol{\mathcal{V}} = (\mathcal{V}_{\mathsf{INT}}, \boldsymbol{\mathcal{V}}_{y_s})$. It $e(\boldsymbol{\delta}_{y_s}, \boldsymbol{\mathcal{V}}_{y_s}) = \mathcal{V}_{\mathsf{INT}}$ returns 1, else 0.
- $\mathsf{AltUnsigncrypt}(\mathcal{PP}, \mathsf{U}, \mathcal{VK}, y_s)$: Let $\mathcal{VK} = (\mathcal{SK}_x^{\mathsf{M}}, \boldsymbol{\mathcal{V}})$. This is same as $\mathsf{Unsigncrypt}$ algorithm, but here we do not need to compute the alt-key $\mathcal{SK}_x^{\mathsf{M}}$ and vText $\boldsymbol{\mathcal{V}}$ respectively involved in the algorithms, $\mathsf{Decrypt}$ and Ver as they are supplied. In other words, it is same as $\mathsf{Unsigncrypt}$ algorithm, except the $\mathsf{Decrypt}$ and Ver are replaced by $\mathsf{AltDecrypt}$ and AltVer respectively.

We note that the objects of a particular form defined above may not be used in both, the proof confidentiality and unforgeability. For example, the signcryptions (resp. vTextKeys) of the forms, sf-type 1, sf-type 2, sf-type 3 and sf-type 4 are not used in the proof unforgeability (resp. confidentiality).

Remark C.1. By construction of signcryption, we have $\hbar_s \neq \hbar_e$. The function $f(X) := \theta_1 X + \theta_2$ is pairwise independent function. So, we do not need to pay attention on distributional relation between the objects involved in signature and alt-key (resp. ciphertext and vText) while simulating these objects in sf-type 1 form.

C.1 The Proof of Confidentiality

Theorem C.1. *Let* P *be a pair encoding scheme for a predicate* $\sim$ *which satisfies* **Conditions** 3.1 *and* $\sim$ *is domain-transferable. Suppose* P *has both the security,* SMH *and* CMH*, the assumptions, DSG1, DSG2 and DSG3 hold in* $\mathcal{J}$*, the one-time signature scheme* OTS *has strong unforgeability, the commitment scheme* $\mathcal{C}$ *has the hiding property and* H *is a collision resistant hash function, then the proposed predicate signcryption scheme* PSC *in Section 7 for the predicate* $\sim$ *is IND-CCA secure in adaptive-predicates model (Definition 2.18).*

Proof. Suppose there are at most q, ν_1 and ν_2 number of key, unsigncrypt and signcrypt oracle queries respectively made by an adversary $\mathscr{A}$. Then the security proof consists of hybrid argument over a sequence of $3q_1 + 2(\nu_1 + \nu_2) + 10$ games, where among the q key queries, q_1 and q_2 respectively be the number of pre-challenged and post-challenged key queries.

Let $\mathsf{U}^* = (\mathsf{com}^*, \delta^* = (\boldsymbol{\delta}_{y_s^*}, \delta_o^*, \mathsf{vk}^*), \mathsf{CT}^* = (C^*_{\mathsf{cpa}}, C^*_0))$ denote the challenge signcryption for the data-indices (y_s^*, y_e^*). Let (U, x, y_s) be any unsigncrypt oracle query, where $\mathsf{U} = (\mathsf{com}, \delta = (\boldsymbol{\delta}_{y_s}, \delta_o, \mathsf{vk}), \mathsf{CT} = (C_{\mathsf{cpa}}, C_0))$. We define an event E as

$$\mathsf{E} := [(\mathsf{vk}^* = \mathsf{vk}) \wedge (\delta_o^* || C_0^* || y_s^* \neq \delta_o || C_0 || y_s)].$$

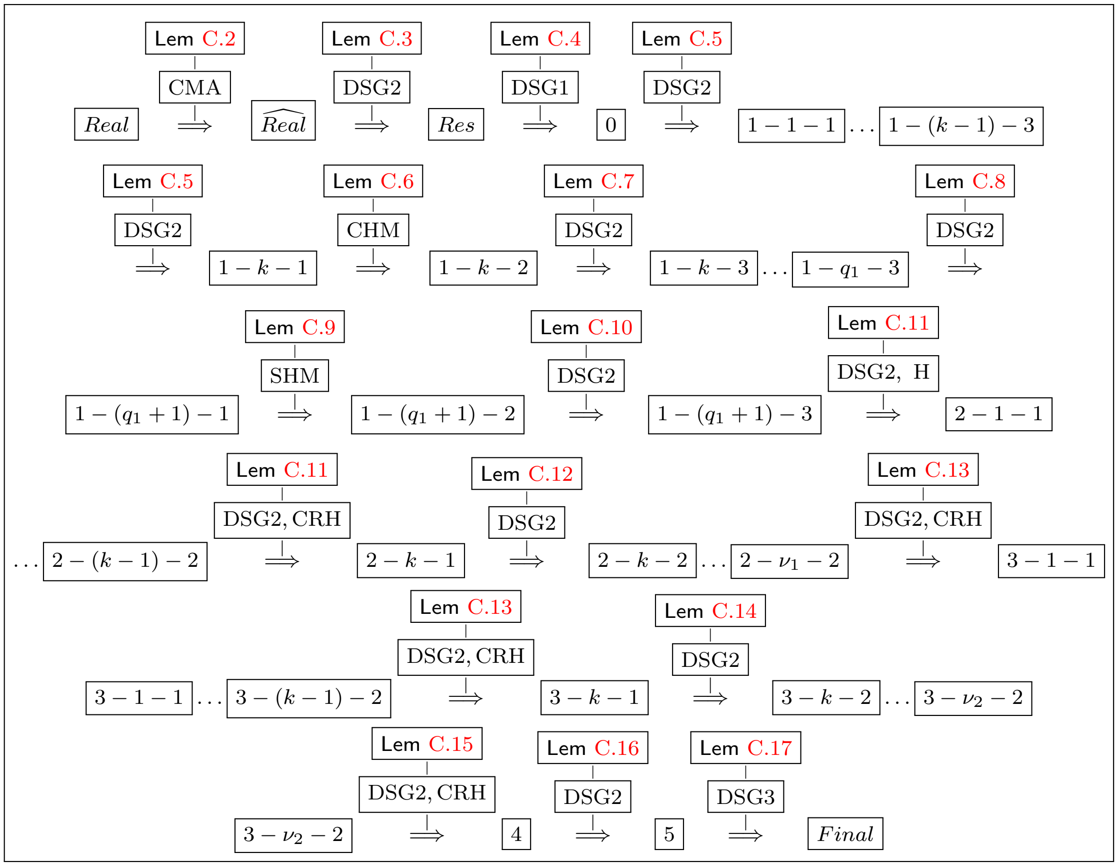
We will apply the hybrid arguments over the following games, where all the unsigncrypt queries are answered by the suitable forms of vTextKeys using the algorithm $\mathsf{AltUnsigncrypt}$.

- Game$_{Real}$:= Original APs-IND-CCA game of predicate signcryption scheme.
- Game$_{\widehat{Real}}$:= Same as Game$_{Real}$ except the challenger always returns $\perp$ on unsigncrypt oracle query if E occurs.
- Game$_{Res}$:= This is same as Game$_{\widehat{Real}}$ except $x \not\sim_N y^*$ is replaced by $x \not\sim_{p_2} y^*$ for each key query x made by $\mathscr{A}$.
- Game$_0$ (= Game$_{1-0-3}$) is just like Game$_{Res}$ except that the challenge signcryption is of sf-type 1.
- In Game$_{1-k-1}$ (for $1 \le k \le q_1$) is same as Game$_{1-(k-1)-3}$ except the k^{th} queried key is sf-type 1.
- Game$_{1-k-2}$ (for $1 \le k \le q_1$) is same as Game$_{1-k-1}$ except the k^{th} queried key is sf-type 2.
- Game$_{1-k-3}$ (for $1 \le k \le q_1$) is same as Game$_{1-k-2}$ except the k^{th} queried key is sf-type 3.
- Game$_{1-(q_1+1)-i}$ (for $1 \le i \le 3$) is same as Game$_{1-q_1-3}$ except the last q_2 queried keys[10] are of sf-type i.
- In Game$_{2-k-1}$ (for $1 \le k \le \nu_1$) is same as Game$_{2-(k-1)-2}$ except the k^{th} unsigncrypt oracle query is answered by vTextKey of the form, sf-type I. In this sequel, we define Game$_{2-0-2}$ = Game$_{1-(q_1+1)-3}$.
- Game$_{2-k-2}$ (for $1 \le k \le \nu_1$) is same as Game$_{2-k-1}$ except the k^{th} unsigncrypt query is answered by vTextKey of the form, sf-type II.
- In Game$_{3-k-1}$ (for $1 \le k \le \nu_2$) is same as Game$_{3-(k-1)-2}$ except the replied signcryption to k^{th} signcrypt oracle query is of sf-type I. In this sequel, we define Game$_{3-0-2}$ = Game$_{2-\nu_1-2}$.

[10] Similar to the proof of Theorem 6.1, we restrict the form of queried keys of type-2 and type-3 after the challenge phase. That is, for all the post key queries x_j ($q_1 + 1 \le j \le q = q_1 + q_2$), $\hat{\alpha}$'s appearing in $\mathbb{G}_{p_2}$ components of $\mathcal{SK}_{x_j}$ are identical.

- Game$_{3-k-2}$ (for $1 \leq k \leq \nu_2$) is same as Game$_{3-k-1}$ except the replied signcryption to k^{th} signcrypt oracle query is of II.
- Game$_4$ is similar to Game$_{3-\nu_2-2}$ except that the challenge signcryption is of sf-type 2.
- Game$_5$ is similar to Game$_4$ except that the challenge signcryption is of sf-type 3.
- Game$_{Final}$ is similar to Game$_5$ except that the challenge signcryption is of sf-type 4.

In Game$_{Final}$, the decommitment decom_b of the challenge message m_b is masked with an independently and uniformly chosen element from $\mathbb{G}_T$ implying the component C_{INT} does not leak any information about decom_b. Since, the primitive commitment schemes $\mathcal{C}$ has hiding property, com_b does not reveal any information about m_b from adversary point of view. Therefore, the adversary $\mathscr{A}$ has no advantage in Game$_{Final}$. The outline of the hybrid arguments over the games is given below, where Lem stands for Lemma.



Using the lemmas referred in the above box (for details of the lemmas, refer to Appendix C.2) and Lemma C.18, we have the following reduction:

$$\begin{aligned}\mathsf{Adv}_{\mathscr{A},\mathrm{PSC}}^{\mathrm{APs-IND-CCA}}(\kappa) \leq \mathsf{Adv}_{\mathscr{B}_0,\mathrm{OTS}}^{\mathrm{sUF-CMA}}(\kappa) + \mathsf{Adv}_{\mathscr{B}_1}^{\mathrm{DSG1}}(\kappa) +& \\ (2q_1 + 2\nu_1 + 2\nu_2 + 5)\mathsf{Adv}_{\mathscr{B}_2}^{\mathrm{DSG2}}(\kappa) + q_1\mathsf{Adv}_{\mathscr{B}_3,\mathrm{P}}^{\mathrm{CMH}}(\kappa) +& \\ \mathsf{Adv}_{\mathscr{B}_4,\mathrm{P}}^{\mathrm{SMH}}(\kappa) + (\nu_1 + \nu_2 + 1)\mathsf{Adv}_{\mathscr{B}_5}^{\mathrm{CRH}}(\kappa) +& \\ \mathsf{Adv}_{\mathscr{B}_6}^{\mathrm{DSG3}}(\kappa) + \mathsf{Adv}_{\mathscr{B}_7}^{\mathrm{Hiding}}(\kappa)&\end{aligned}$$

where $\mathscr{B}_0, \mathscr{B}_1, \mathscr{B}_2, \mathscr{B}_3, \mathscr{B}_4, \mathscr{B}_5, \mathscr{B}_6$ and $\mathscr{B}_7$ are PPT algorithms whose running times are same as that of $\mathscr{A}$. This completes the proof. □

Discussion C.2. By the definition of $\text{Game}_{\widehat{Real}}$, $\mathscr{B}$ returns $\perp$ to $\mathscr{A}$ if E occurs. When E does not occur, there are three possibilities, (a) $[(\mathsf{vk}^* = \mathsf{vk}) \wedge (\delta_o^* || C_0^* || y_s^* = \delta_o || C_0 || y_s)]$, (b) $[(\mathsf{vk}^* \neq \mathsf{vk}) \wedge (\delta_o^* || C_0^* || y_s^* = \delta_o || C_0 || y_s)]$ and (c) $[(\mathsf{vk}^* \neq \mathsf{vk}) \wedge (\delta_o^* || C_0^* || y_s^* \neq \delta_o || C_0 || y_s)]$. Since, for a valid unsigncrypt oracle query $\mathsf{U} = (\mathsf{com}, \delta, \mathsf{CT})$, the case (a) implies that $(\mathsf{U}^*, y_s^*) = (\mathsf{U}, y_s)$ which is forbidden by the natural restriction of the APs-IND-CCA game. The case (b) is impossible as $C_0^* = C_0$ implies $\mathsf{vk}^* = \mathsf{vk}$ which is absurd. Therefore, from $\text{Game}_{\widehat{Real}}$ onwards $\mathscr{B}$ answers the unsigncrypt queries of $\mathscr{A}$ by running $\mathsf{AltUnsigncrypt}$ algorithm if the case (c) only occurs else returns $\perp$.

C.2 Lemmas Used in the Proof of Theorem C.1

Lemma C.2. Game_{Real} *and* $\text{Game}_{\widehat{Real}}$ *are indistinguishable under the strong unforgeability of* OTS. *That is, for every adversary* $\mathscr{A}$*, there exists a PPT algorithm* $\mathscr{B}$ *such that* $|\mathsf{Adv}^{\text{Real}}_{\mathscr{A},\text{PSC}}(\kappa) - \mathsf{Adv}^{\widehat{Real}}_{\mathscr{A},\text{PSC}}(\kappa)| \leq \mathsf{Adv}^{\text{sUF-CMA}}_{\mathscr{B},\text{OTS}}(\kappa)$.

Proof. Suppose $\mathscr{A}$ can distinguish the games with a non-negligible probability, then we will program a PPT algorithm $\mathscr{B}$ for breaking the strong unforgeability of OTS with the same probability. Here $\mathscr{B}$ plays the role of an adversary in sUF-CMA game and the role of a challenger in APs-IND-CCA game. Let $\mathcal{CH}$ be the challenger for OTS. $\mathcal{CH}$ runs $(\mathsf{vk}^*, \mathsf{signk}^*) \longleftarrow \mathsf{OTS.Gen}$ and gives vk^* to $\mathscr{B}$. Then, $\mathscr{B}$ runs the Setup algorithm, keeps $\mathcal{MSK}$ to itself and gives the public parameters $\mathcal{PP}$ to $\mathscr{A}$.

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

- KeyGen: Let x be any key query made by $\mathscr{A}$. Since $\mathscr{B}$ knows $\mathcal{MSK}$, it replies $\mathcal{SK}_x$ to $\mathscr{A}$.
- $\mathsf{Signcrypt}$: Let (m, x, y_s, y_e) be any signcrypt query made by $\mathscr{A}$. Then, $\mathscr{B}$ constructs a key $\mathcal{SK}_x$ using $\mathcal{MSK}$. Then, using this key it runs $\mathsf{Signcrypt}$ algorithm (in Section 7) and answers the signcryption U to $\mathscr{A}$.
- $\mathsf{Unsigncrypt}$: Let (U, x, y_s) be any unsigncrypt query made by $\mathscr{A}$, where $\mathsf{U} = (\mathsf{com}, \delta, \mathsf{CT})$. If this query satisfies the event E, $\mathscr{B}$ returns δ_o and aborts. $\mathscr{B}$ first constructs the normal vTextKey $\mathcal{VK} := (\mathcal{SK}_x^{\mathsf{M}}, \mathcal{V})$, then using $\mathcal{VK}$ it runs $\mathsf{AltUnsigncrypt}$ and returns the output to $\mathscr{A}$.

Challenge Phase: $\mathscr{A}$ submits two equal length message m_0, m_1, a key-index x, a challenge associated data-index y_s^* of sender and a challenge associated data-index y_e^* of receiver to $\mathscr{B}$. Then, $\mathscr{B}$ computes the key $\mathcal{SK}_x$ as it knows $\mathcal{MSK}$. Picks $b \xleftarrow{\mathsf{U}} \{0, 1\}$ and runs $\mathsf{Signcrypt}(\mathcal{PP}, m_b, \mathcal{SK}_x, y_s^*, y_e^*)$, where it queries for one-time signature to $\mathcal{CH}$ for the message $C_0^* || y_s^*$ and gets the replied signature δ_o^*. It returns $\mathsf{U}^* := (\mathsf{com}^*, \delta^*, \mathsf{CT}^*)$, where $\delta^* := (\boldsymbol{\delta}_{y_s^*}, \delta_o^*, \mathsf{vk}^*)$ to $\mathscr{A}$.

Query Phase-2: Similar to phase-1.

Guess: $\mathscr{A}$ sends a guess b' to $\mathscr{B}$. ($\mathscr{B}$ does nothing with this b')

Analysis: Since both the games are identical except the event E with probability ε. By the event E, we have $\delta_o^* || C_0^* || y_s^* \neq \delta_o || C_0 || y_s$. Therefore, δ_o is a valid forgery for the message $C_0 || y_s$. □

Lemma C.3. $\mathrm{Game}_{\widehat{Real}}$ *and* Game_{Res} *are indistinguishable under the DSG2 assumption. That is, for every adversary* $\mathscr{A}$*, there exists a PPT algorithm* $\mathscr{B}$ *such that*

$$|\mathsf{Adv}_{\mathscr{A},\mathrm{PSC}}^{\widehat{Real}}(\kappa) - \mathsf{Adv}_{\mathscr{A},\mathrm{PSC}}^{\mathrm{Res}}(\kappa)| \le \mathsf{Adv}_{\mathscr{B}}^{\mathrm{DSG2}}(\kappa).$$

Proof. Similar to Lemma B.1. □

Lemma C.4. Game_{Res} *and* Game_0 *are indistinguishable under the DSG1 assumption. That is, for every adversary* $\mathscr{A}$*, there exists a PPT algorithm* $\mathscr{B}$ *such that*

$$|\mathsf{Adv}_{\mathscr{A},\mathrm{PSC}}^{\mathrm{Res}}(\kappa) - \mathsf{Adv}_{\mathscr{A},\mathrm{PSC}}^{0}(\kappa)| \le \mathsf{Adv}_{\mathscr{B}}^{\mathrm{DSG1}}(\kappa).$$

Proof. The only difference between the games is the form of the challenge signcryption, normal or sf-type 1. In both forms of signcryptions, the signature part is normal, but the ciphertext part is normal or sf-type 1 according to the challenge signcryption is normal or sf-type 1. Therefore, the proof could be done in similar way as in Lemma B.2. □

Lemma C.5. $\mathrm{Game}_{1-(k-1)-3}$ *and* Game_{1-k-1} *are indistinguishable under DSG2 assumption. That is, for every adversary* $\mathscr{A}$*, there exists a PPT algorithm* $\mathscr{B}$ *such that*

$$|\mathsf{Adv}_{\mathscr{A},\mathrm{PSC}}^{1-(k-1)-3}(\kappa) - \mathsf{Adv}_{\mathscr{A},\mathrm{PSC}}^{1-k-1}(\kappa)| \le \mathsf{Adv}_{\mathscr{B}}^{\mathrm{DSG2}}(\kappa) \quad \textit{for} \quad 1 \le k \le q_1.$$

Proof. For proof, refer to Lemma B.3. □

Lemma C.6. 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* $\mathscr{A}$*, there exists a PPT algorithm* $\mathscr{B}$ *such that*

$$|\mathsf{Adv}_{\mathscr{A},\mathrm{PSC}}^{1-k-1}(\kappa) - \mathsf{Adv}_{\mathscr{A},\mathrm{PSC}}^{1-k-2}(\kappa)| \le \mathsf{Adv}_{\mathscr{B},\mathrm{P}}^{\mathrm{CMH}}(\kappa) \quad \textit{for} \quad 1 \le k \le q_1.$$

Proof. Following the proof of Lemma B.4, it can be proven. □

Lemma C.7. Game_{1-k-2} *and* Game_{1-k-3} *are indistinguishable under the DSG2 assumption. That is, for every adversary* $\mathscr{A}$*, there exists a PPT algorithm* $\mathscr{B}$ *such that*

$$|\mathsf{Adv}_{\mathscr{A},\mathrm{PSC}}^{1-k-2}(\kappa) - \mathsf{Adv}_{\mathscr{A},\mathrm{PSC}}^{1-k-3}(\kappa)| \le \mathsf{Adv}_{\mathscr{B}}^{\mathrm{DSG2}}(\kappa) \quad \textit{for} \quad 1 \le k \le q_1.$$

Proof. The proof is similar to that of Lemma B.5. □

Lemma C.8. Game_{1-q_1-3} *and* $\mathrm{Game}_{1-(q_1+1)-1}$ *are indistinguishable under the DSG2 assumption. That is, for every adversary* $\mathscr{A}$*, there exists a PPT algorithm* $\mathscr{B}$ *such that*

$$|\mathsf{Adv}_{\mathscr{A},\mathrm{PSC}}^{1-q_1-3}(\kappa) - \mathsf{Adv}_{\mathscr{A},\mathrm{PSC}}^{1-(q_1+1)-1}(\kappa)| \le \mathsf{Adv}_{\mathscr{B}}^{\mathrm{DSG2}}(\kappa).$$

Proof. For proof, we refer to Lemma B.6 □

Lemma C.9. $\mathrm{Game}_{1-(q_1+1)-1}$ *and* $\mathrm{Game}_{1-(q_1+1)-2}$ *are indistinguishable under* SMH *security of the primitive pair encoding scheme,* P*. That is, for every adversary* $\mathscr{A}$*, there exists a PPT algorithm* $\mathscr{B}$ *such that*

$$|\mathsf{Adv}_{\mathscr{A},\mathrm{PSC}}^{1-(q_1+1)-1}(\kappa) - \mathsf{Adv}_{\mathscr{A},\mathrm{PSC}}^{1-(q_1+1)-2}(\kappa)| \le \mathsf{Adv}_{\mathscr{B},\mathrm{P}}^{\mathrm{SMH}}(\kappa).$$

Proof. The proof is similar to Lemma B.7. □

Lemma C.10. $\text{Game}_{1-(q_1+1)-2}$ *and* $\text{Game}_{1-(q_1+1)-3}$ *are indistinguishable under the DSG2 assumption. That is, for every adversary* $\mathscr{A}$*, there exists a PPT algorithm* $\mathscr{B}$ *such that*

$$|\mathsf{Adv}^{1-(q_1+1)-2}_{\mathscr{A},\text{PSC}}(\kappa) - \mathsf{Adv}^{1-(q_1+1)-3}_{\mathscr{A},\text{PSC}}(\kappa)| \leq \mathsf{Adv}^{\text{DSG2}}_{\mathscr{B}}(\kappa).$$

Proof. The proof can be done in similar manner as in Lemma B.8. □

Lemma C.11. $\text{Game}_{2-(k-1)-2}$ *and* Game_{2-k-1} *are indistinguishable under the DSG2 assumption and collision resistant property of* H*. That is, for every adversary* $\mathscr{A}$*, there exists a PPT algorithm* $\mathscr{B}$ *such that*

$$|\mathsf{Adv}^{2-(k-1)-2}_{\mathscr{A},\text{PSC}}(\kappa) - \mathsf{Adv}^{2-k-1}_{\mathscr{A},\text{PSC}}(\kappa)| \leq \mathsf{Adv}^{\text{DSG2}}_{\mathscr{B}}(\kappa) + \mathsf{Adv}^{\text{CRH}}_{\mathscr{B}}(\kappa) \quad \textit{for} \quad 1 \leq k \leq \nu_1.$$

Proof. The proof is similar to that of Lemma B.9. Following Discussion C.2 for a valid unsigncrypt query, we must have $\mathsf{vk}^* \neq \mathsf{vk}$, which in turn implies that $\hbar_e^* \neq \hbar_e$. Therefore, the proof will be simpler than than the proof of Lemma B.9. □

Lemma C.12. Game_{2-k-1} *and* Game_{2-k-2} *are indistinguishable under the DSG2 assumption. That is, for every adversary* $\mathscr{A}$*, there exists a PPT algorithm* $\mathscr{B}$ *such that*

$$|\mathsf{Adv}^{2-k-1}_{\mathscr{A},\text{PSC}}(\kappa) - \mathsf{Adv}^{2-k-2}_{\mathscr{A},\text{PSC}}(\kappa)| \leq \mathsf{Adv}^{\text{DSG2}}_{\mathscr{B}}(\kappa) \quad \textit{for } 1 \leq k \leq \nu_1.$$

Proof. The proof is similar to that of Lemma B.10. □

Lemma C.13. $\text{Game}_{3-(k-1)-2}$ *and* Game_{3-k-1} *are indistinguishable under the DSG2 assumption and collision resistant property of* H*. That is, for every adversary* $\mathscr{A}$*, there exists a PPT algorithm* $\mathscr{B}$ *such that*

$$|\mathsf{Adv}^{3-(k-1)-2}_{\mathscr{A},\text{PSC}}(\kappa) - \mathsf{Adv}^{3-k-1}_{\mathscr{A},\text{PSC}}(\kappa)| \leq \mathsf{Adv}^{\text{DSG2}}_{\mathscr{B}}(\kappa) + \mathsf{Adv}^{\text{CRH}}_{\mathscr{B}}(\kappa) \quad \textit{for } 1 \leq k \leq \nu_2.$$

Proof. In both the games, the queried key is sf-type 3, the unsigncrypt oracle queries are answered by vTextKeys of sf-type II, the challenge signcryption is of sf-type 1 (that means signature part is normal whereas ciphertext is sf-type 1) and the ciphertext part in all the replied signcryptions to signcrypt oracle queries are normal. The only difference between the games is the form of the replied signcryption to k^{th} signcrypt oracle query. More precisely, the signature part in k^{th} replied signcryption is normal (resp. sf-type 1) in $\text{Game}_{3-(k-1)-2}$ (resp. Game_{3-k-1}). Therefore, following the proof of Lemma A.6, it can be done. Since, ciphertext part in the challenge signcryption is sf-type 1, so the collision resistant property of H will be used only to guarantee $\hbar_e^* \neq \hbar_s^{(k)}$ in the proof, where $\hbar_s^{(k)} := H_s(1, \mathsf{vk}^{(k)}, y_s^{(k)})$. □

Lemma C.14. Game_{3-k-1} *and* Game_{3-k-2} *are indistinguishable under the DSG2 assumption. That is, for every adversary* $\mathscr{A}$*, there exists a PPT algorithm* $\mathscr{B}$ *such that*

$$|\mathsf{Adv}^{3-k-1}_{\mathscr{A},\text{PSC}}(\kappa) - \mathsf{Adv}^{3-k-2}_{\mathscr{A},\text{PSC}}(\kappa)| \leq \mathsf{Adv}^{\text{DSG2}}_{\mathscr{B}}(\kappa) \quad \textit{for} \quad 1 \leq k \leq \nu_2.$$

Proof. The only difference between the games is the form of replied signcryption to the k^{th} signcrypt oracle query. The form of signature part in the k^{th} replied signcryption is sf-type 1 (resp. sf-type 2) in Game_{3-k-1} (resp. Game_{3-k-2}). We refer to proof of Lemma A.7. □

Lemma C.15. $\text{Game}_{3-\nu_2-2}$ *and* Game_4 *are indistinguishable under the DSG2 assumption and collision resistant property of* H*. That is, for every adversary* $\mathscr{A}$*, there exists a PPT algorithm* $\mathscr{B}$ *such that*

$$|\mathsf{Adv}^{3-\nu_2-2}_{\mathscr{A},\text{PSC}}(\kappa) - \mathsf{Adv}^{4}_{\mathscr{A},\text{PSC}}(\kappa)| \leq \mathsf{Adv}^{\text{DSG2}}_{\mathscr{B}}(\kappa) + \mathsf{Adv}^{\text{CRH}}_{\mathscr{B}}(\kappa).$$

Proof. In both the games, the queried key is sf-type 3, the unsigncrypt queries are answered by vTextKeys of sf-type II, the replied signcryptions to signcrypt oracle queries are of sf-type II and the ciphertext part in the challenge signcryption is sf-type 1. The only difference between the games is the form of signature in the challenge signcryption, i.e., it is normal in $\text{Game}_{3-\nu_2-2}$ and sf-type 1 in Game_4. Therefore, the proof can be done following the proof of Lemma A.6. In the proof, the collision resistant property of H is used to guarantee $\hbar_s^* \neq \hbar_e^*$. □

Lemma C.16. *Game_4 and Game_5 are indistinguishable under the DSG3 assumption. That is, for every adversary $\mathscr{A}$, there exists a PPT algorithm $\mathscr{B}$ such that $|\mathsf{Adv}^{4}_{\mathscr{A},\text{PSC}}(\kappa) - \mathsf{Adv}^{5}_{\mathscr{A},\text{PSC}}(\kappa)| \leq \mathsf{Adv}^{\text{DSG2}}_{\mathscr{B}}(\kappa)$.*

Proof. The only difference between the games is the form of the signature in the challenge signcryption, i.e., it is either sf-type 1 or sf-type 2. We refer to proof of Lemma A.7. □

Lemma C.17. *Game_5 and Game_{Final} are indistinguishable under the DSG3 assumption. That is, for every adversary $\mathscr{A}$, there exists a PPT algorithm $\mathscr{B}$ such that $|\mathsf{Adv}^{5}_{\mathscr{A},\text{PSC}}(\kappa) - \mathsf{Adv}^{Final}_{\mathscr{A},\text{PSC}}(\kappa)| \leq \mathsf{Adv}^{\text{DSG3}}_{\mathscr{B}}(\kappa)$.*

Proof. The only difference between the games is the form of the ciphertext in the challenge signcryption, i.e., it is either sf-type 1 or sf-type 2. For proof, we refer to Lemma B.11. □

Lemma C.18. *For every adversary $\mathscr{A}$, there exists a PPT algorithm $\mathscr{B}$ such that $\mathsf{Adv}^{Final}_{\mathscr{A},\text{PSC}}(\kappa) \leq \mathsf{Adv}^{\text{Hiding}}_{\mathscr{B}}(\kappa)$.*

Proof. In the final game, Game_{Final} the decommitment part decom_b of the challenge message m_b is information theoretically hidden in the ciphertext part of the challenge signcryption. The only counter part of m_b remains in challenge signcryption is the commitment part com_b. Since, the commitment scheme $\mathcal{C}$ has hiding property, com_b does not leak any information about m_b from adversary point of view. We skip the details of the simulation as it is very straightforward. □

Theorem C.19. *Let P be a pair encoding scheme for a predicate $\sim$ which satisfies* **Conditions 3.1** *and $\sim$ is domain-transferable. Suppose P has the PMH security, the assumptions, DSG1, DSG2 and DSG3 hold in $\mathcal{J}$, the one-time signature scheme OTS has strong unforgeability, the commitment scheme $\mathcal{C}$ has the hiding property and H is a collision resistant hash function, then the proposed predicate signcryption scheme PSC in Section 7 for the predicate $\sim$ is IND-CCA secure in adaptive-predicates model.*

Proof. Similar to the proof of Theorem C.1. □

C.3 The Proof of Unforgeability

Theorem C.20. *Let P be a pair encoding scheme for a predicate $\sim$ which satisfies* **Conditions 3.1** *and $\sim$ is domain-transferable. Suppose P has the CMH security, the assumptions, DSG1, DSG2 and DSG3 hold in $\mathcal{J}$, the one-time signature scheme OTS has strong unforgeability and H is a collision resistant hash function, then the proposed predicate signcryption scheme PSC in Section 7 for the predicate $\sim$ is strongly existential unforgeable in adaptive-predicates model (Definition 2.22).*

Proof. Let $\mathscr{A}$ be an adversary in APs-sUF-CMA model who can break the strong unforgeability of the proposed predicate signcryption scheme with non-negligible advantage ε. Let ν_2 be the number of signcrypt oracle queries made by $\mathscr{A}$. Let $(m^{(i)}, x^{(i)}, y_s^{(i)}, y_e^{(i)})$ be the i^{th} query and $\mathsf{U}^{(i)} = (\mathsf{com}_i, \delta^{(i)} =$

$(\boldsymbol{\delta}_{y_s^{(i)}}, \delta_o^{(i)}, \mathsf{vk}^{(i)}), \mathsf{CT}^{(i)})$ be the corresponding replied signcryption. Let $\mathsf{U}^* = (\mathsf{com}^*, \delta^*, \mathsf{CT}^*)$ be the forgery made by $\mathscr{A}$ for the message (m^*, y_s^*, y_e^*). We define an event as

$$\mathsf{Forged} := \mathsf{vk}^* \notin \{\mathsf{vk}^{(i)} |\ i \in [\nu_2]\}.$$

Then, we have

$$\varepsilon \leq \Pr[\mathscr{A}\ \text{ Succeeds}] := \Pr[\mathscr{A}\ \text{ Succeeds} \wedge \mathsf{Forged}] + \Pr[\mathscr{A}\ \text{ Succeeds} \wedge \neg\mathsf{Forged}]$$

$$\implies \Pr[\mathscr{A}\ \text{ Succeeds} \wedge \mathsf{Forged}] \geq \varepsilon/2 \ \ \textit{or}\ \ \Pr[\mathscr{A}\ \text{ Succeeds} \wedge \neg\mathsf{Forged}] \geq \varepsilon/2$$

Case: $\neg$Forged**.** We will develop an algorithm $\mathscr{B}_{\mathsf{OTS}}$ for breaking the string unforgeability of the primitive one-time signature scheme OTS with advantage at least $\varepsilon/2\nu_2$. Let $\mathcal{CH}$ be the challenger for the primitive one-time signature scheme OTS. The challenger $\mathcal{CH}$ runs $(\mathsf{vk}^*, \mathsf{signk}^*) \longleftarrow \mathsf{OTS.Gen}(1^\kappa)$ and gives vk^* to $\mathscr{B}_{\mathsf{OTS}}$. $\mathscr{B}_{\mathsf{OTS}}$ runs the Setup algorithm (as described in Section 7), keeps $\mathcal{MSK}$ to itself and sends $\mathcal{PP}$ to $\mathscr{A}$. Then, picks $i \xleftarrow{\text{U}} [\nu_2]$ as a guess such that $\mathsf{vk}^* = \mathsf{vk}^{(i)}$. For notational simplicity, we ignore the superscript (i).

- KeyGen Query: $\mathscr{B}_{\mathsf{OTS}}$ answers this query using $\mathcal{MSK}$.
- Signcrypt Query: Let (m, x, y_s, y_e) be the j^{th} signcrypt oracle query to $\mathscr{B}_{\mathsf{OTS}}$ by $\mathscr{A}$.
  - $(j \neq i)$ **:** $\mathscr{B}_{\mathsf{OTS}}$ runs $(\mathsf{com}, \mathsf{decom}) \longleftarrow \mathsf{Commit}(m)$ and $(\mathsf{vk}, \mathsf{signk}) \longleftarrow \mathsf{OTS.Gen}(1^\kappa)$. It constructs a key $\mathcal{SK}_x$ using $\mathcal{MSK}$. Then, runs $\boldsymbol{\delta}_{y_s} \longleftarrow \mathsf{PS.Sign}(\mathsf{vk},\ \mathcal{SK}_x,\ y_s)$, $C_{\mathsf{cpa}} \longleftarrow \mathsf{PE.Encrypt}^*(\mathsf{decom}, y_e)$ and computes $\hbar_e := H(0,\ \mathsf{com},\ \boldsymbol{\delta}_{y_s}, \mathsf{vk}, C_{\mathsf{cpa}})$. Then, it computes $C_0 := g^{s_0(\theta_1\hbar_e + \theta_2)}$ and $\delta_o \longleftarrow \mathsf{OTS.Sign}(C_0||y_s, \mathsf{signk})$. Returns the signcryption $\mathsf{U} := (\mathsf{com}, \delta := (\boldsymbol{\delta}_{y_s}, \delta_o, \mathsf{vk}), \mathsf{CT} := (C_{\mathsf{cpa}}, C_0))$ to $\mathscr{A}$.
  - $(j = i)$ **:**
    Same as above except $\mathscr{B}_{\mathsf{OTS}}$ does not execute $\mathsf{OTS.Gen}(1^\kappa)$ but it sets $\mathsf{vk} := \mathsf{vk}^*$ and it makes an one-time signature query to $\mathcal{CH}$ for the message $C_0||y_s$ and gets the replied signature δ_o.
- Unsigncrypt Query: It can answer the query as it knows $\mathcal{MSK}$.
- Forgery: $\mathscr{A}$ outputs a tuple $(\mathsf{U}^*, y_s^*, y_e^*)$, where $\mathsf{U}^* := (\mathsf{com}^*, \delta^*, \mathsf{C}^*)$ and $\delta^* := (\boldsymbol{\delta}_{y_s^*}, \delta_o^*, \mathsf{vk}^*)$. Then, $\mathscr{B}_{\mathsf{OTS}}$ forges the signature δ_o^* for $C_0^*||y_s^*$ to the one-time signature scheme OTS.

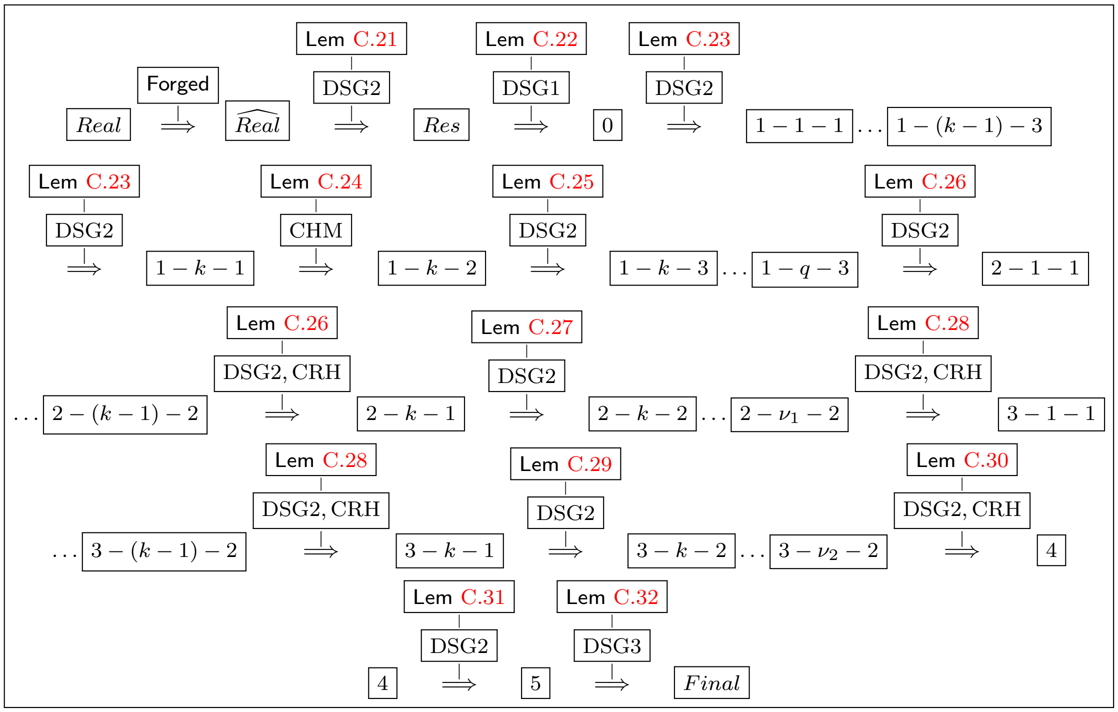
Analysis: With probability $1/\nu_2$, $\mathscr{B}_{\mathsf{OTS}}$ correctly guesses i such that the event Forged is happened. Now, we only have to show that $\delta_o^*||C_0^*||y_s^* \neq \delta_o||C_0||y_s$ (we ignore the superscript, (i)). Indeed, if $\delta_o^*||C_0||y_s^* = \delta_o||C_0^*||y_s$, we have $\delta_o^* = \delta_o$, $y_s^* = y_s$ and $C_0^* = C_0$. Now $C_0^* = C_0$ implies $\hbar_e^* = \hbar_e$, so we have $\mathsf{com}^* = \mathsf{com}$, $\boldsymbol{\delta}_{y_s^*} = \boldsymbol{\delta}_{y_s}$ and $C_{\mathsf{cpa}}^* = C_{\mathsf{cpa}}$. Overall, we have $(\mathsf{U}^*, m^*, y_s^*, y_e^*) = (\mathsf{U}, m, y_s, y_e)$ which leads a contradiction to *APs-sUF-CMA* security model.

Case: Forged**.** Suppose there are at most q key queries and ν_1 unsigncrypt queries, then the security proof consists of hybrid argument over a sequence of $3q + 2(\nu_1 + \nu_2) + 7$ games defined below:

- Game_{Real} := Original APs-sUF-CMA game of PSC scheme.
- $\text{Game}_{\widehat{Real}}$:= Same as Game_{Real} except the event Forged always happens.
- Game_{Res} := This is same as $\text{Game}_{\widehat{Real}}$ except $x \not\sim_N y^*$ is replaced by $x \not\sim_{p_2} y^*$ for each key query x made by $\mathscr{A}$.
- Game_0 $(= \text{Game}_{1-0-3})$ is just like Game_{Res} except that the vTextKey for verifying the forgery is of sf-type 1.

- In Game$_{1-k-1}$ (for $1 \leq k \leq q$) is same as Game$_{1-(k-1)-3}$ except the k^{th} queried key is sf-type 1.
- Game$_{1-k-2}$ (for $1 \leq k \leq q$) is same as Game$_{1-k-1}$ except the k^{th} queried key is sf-type 2.
- Game$_{1-k-3}$ (for $1 \leq k \leq q$) is same as Game$_{1-k-2}$ except the k^{th} queried key is sf-type 3.
- In Game$_{2-k-1}$ (for $1 \leq k \leq \nu_1$) is same as Game$_{2-(k-1)-2}$ except the k^{th} unsigncrypt oracle query is answered by the vTextKey of sf-type I. (In this sequel, we define Game$_{2-0-2}$ = Game$_{1-q-3}$)
- Game$_{2-k-2}$ (for $1 \leq k \leq \nu_1$) is same as Game$_{2-k-1}$ except the k^{th} unsigncrypt oracle query is answered by the vTextKey of sf-type II.
- In Game$_{3-k-1}$ (for $1 \leq k \leq \nu_2$) is same as Game$_{3-(k-1)-2}$ except the replied signcryption to the k^{th} signcrypt oracle query is of sf-type I. (So, in this sequel Game$_{3-0-2}$ = Game$_{2-\nu_1-2}$)
- Game$_{3-k-2}$ (for $1 \leq k \leq \nu_2$) is same as Game$_{3-k-1}$ except the replied signcryption to the k^{th} signcrypt oracle query is of sf-type II.
- Game$_4$ is similar to Game$_{3-\nu_2-2}$ except that the vTextKey for verifying the forgery is sf-type 2.
- Game$_5$ is similar to Game$_4$ except that the vTextKey for verifying the forgery is sf-type 3.
- Game$_{Final}$ is similar to Game$_5$ except that the vTextKey for verifying the forgery is sf-type 4

In Game$_{Final}$, the component $\mathcal{V}_{\mathsf{INT}}$ in the vText part of vTextKey is chosen independently and uniformly at random from $\mathbb{G}_T$. This implies that the forgery will be invalid with respect to the vTextKey. Therefore, the adversary $\mathscr{A}$ has no advantage in Game$_{Final}$. The outline of the hybrid arguments over the games is given below, where Lem stands for Lemma.



Using the lemmas referred in the above box (for details of the lemmas, refer to Appendix C.4), we have the following reduction:

$$\begin{aligned}\mathsf{Adv}_{\mathscr{A},\mathrm{PSC}}^{\mathrm{APs-sUF-CMA}}(\kappa) \leq \nu_2 \mathsf{Adv}_{\mathscr{B}_0,\mathrm{OTS}}^{\mathrm{sUF-CMA}}(\kappa) + \mathsf{Adv}_{\mathscr{B}_1}^{\mathrm{DSG1}}(\kappa) + \\ (2q + 2\nu_1 + 2\nu_2 + 3)\mathsf{Adv}_{\mathscr{B}_2}^{\mathrm{DSG2}}(\kappa) + q\mathsf{Adv}_{\mathscr{B}_3,\mathrm{P}}^{\mathrm{CMH}}(\kappa) + \\ (\nu_1 + \nu_2 + 1)\mathsf{Adv}_{\mathscr{B}_4}^{\mathrm{CRH}}(\kappa) + \mathsf{Adv}_{\mathscr{B}_5}^{\mathrm{DSG3}}(\kappa)\end{aligned}$$

where $\mathscr{B}_0, \mathscr{B}_1, \mathscr{B}_2, \mathscr{B}_3, \mathscr{B}_4$ and $\mathscr{B}_5$ are PPT algorithms whose running times are same as that of $\mathscr{A}$. This completes the proof. □

C.4 Lemmas Used in the Proof of Theorem C.20

Lemma C.21. $\mathrm{Game}_{\widehat{Real}}$ *and* Game_{Res} *are indistinguishable under the DSG2 assumption. That is, for every adversary* $\mathscr{A}$*, there exists a PPT algorithm* $\mathscr{B}$ *such that*

$$|\mathsf{Adv}^{\widehat{Real}}_{\mathscr{A},\mathrm{PSC}}(\kappa) - \mathsf{Adv}^{\mathrm{Res}}_{\mathscr{A},\mathrm{PSC}}(\kappa)| \leq \mathsf{Adv}^{\mathrm{DSG2}}_{\mathscr{B}}(\kappa).$$

Proof. Similar to Lemma A.1. □

Lemma C.22. Game_{Res} *and* Game_0 *are indistinguishable under the DSG1 assumption. That is, for every adversary* $\mathscr{A}$*, there exists a PPT algorithm* $\mathscr{B}$ *such that*

$$|\mathsf{Adv}^{\mathrm{Res}}_{\mathscr{A},\mathrm{PSC}}(\kappa) - \mathsf{Adv}^{0}_{\mathscr{A},\mathrm{PSC}}(\kappa)| \leq \mathsf{Adv}^{\mathrm{DSG1}}_{\mathscr{B}}(\kappa).$$

Proof. The proof could be done in similar way as in Lemma A.2. □

Lemma C.23. $\mathrm{Game}_{1-(k-1)-3}$ *and* Game_{1-k-1} *are indistinguishable under the DSG2 assumption. That is, for every adversary* $\mathscr{A}$*, there exists a PPT algorithm* $\mathscr{B}$ *such that*

$$|\mathsf{Adv}^{1-(k-1)-3}_{\mathscr{A},\mathrm{PSC}}(\kappa) - \mathsf{Adv}^{1-k-1}_{\mathscr{A},\mathrm{PSC}}(\kappa)| \leq \mathsf{Adv}^{\mathrm{DSG2}}_{\mathscr{B}}(\kappa) \ \ \textit{for} \ \ 1 \leq k \leq q.$$

Proof. For proof, refer to Lemma A.3. □

Lemma C.24. Game_{1-k-1} *and* Game_{1-k-2} *are indistinguishable under* CMH *security of the pair encoding scheme* P*. That is, for every adversary* $\mathscr{A}$*, there exists a PPT algorithm* $\mathscr{B}$ *such that*

$$|\mathsf{Adv}^{1-k-1}_{\mathscr{A},\mathrm{PSC}}(\kappa) - \mathsf{Adv}^{1-k-2}_{\mathscr{A},\mathrm{PSC}}(\kappa)| \leq \mathsf{Adv}^{\mathrm{CMH}}_{\mathscr{B},\mathrm{P}}(\kappa) \ \ \textit{for} \ \ 1 \leq k \leq q.$$

Proof. Following the proof of Lemma A.4, it can be proven. □

Lemma C.25. Game_{1-k-2} *and* Game_{1-k-3} *are indistinguishable under the DSG2 assumption. That is, for every adversary* $\mathscr{A}$*, there exists a PPT algorithm* $\mathscr{B}$ *such that*

$$|\mathsf{Adv}^{1-k-2}_{\mathscr{A},\mathrm{PSC}}(\kappa) - \mathsf{Adv}^{1-k-3}_{\mathscr{A},\mathrm{PSC}}(\kappa)| \leq \mathsf{Adv}^{\mathrm{DSG2}}_{\mathscr{B}}(\kappa) \ \ \textit{for} \ \ 1 \leq k \leq q.$$

Proof. The proof is similar to that of Lemma A.5. □

Lemma C.26. $\mathrm{Game}_{2-(k-1)-2}$ *and* Game_{2-k-1} *are indistinguishable under the DSG2 assumption and collision resistant property of* H*. That is, for every adversary* $\mathscr{A}$*, there exists a PPT algorithm* $\mathscr{B}$ *such that*

$$|\mathsf{Adv}^{2-(k-1)-2}_{\mathscr{A},\mathrm{PSC}}(\kappa) - \mathsf{Adv}^{2-k-1}_{\mathscr{A},\mathrm{PSC}}(\kappa)| \leq \mathsf{Adv}^{\mathrm{DSG2}}_{\mathscr{B}}(\kappa) + \mathsf{Adv}^{\mathrm{CRH}}_{\mathscr{B}}(\kappa) \ \ \textit{for} \ \ 1 \leq k \leq \nu_1.$$

Proof. The proof is similar to that of Lemma B.9. The collision resistant property of H will be used only to guarantee $\hbar^*_s \neq \hbar^{(k)}_e$ (following the Remark C.1). □

Lemma C.27. Game_{2-k-1} *and* Game_{2-k-2} *are indistinguishable under the DSG2 assumption. That is, for every adversary* $\mathscr{A}$*, there exists a PPT algorithm* $\mathscr{B}$ *such that*

$$|\mathsf{Adv}^{2-k-1}_{\mathscr{A},\mathrm{PSC}}(\kappa) - \mathsf{Adv}^{2-k-2}_{\mathscr{A},\mathrm{PSC}}(\kappa)| \leq \mathsf{Adv}^{\mathrm{DSG2}}_{\mathscr{B}}(\kappa) \text{ for } 1 \leq k \leq \nu_1.$$

Proof. The proof is similar to that of Lemma B.10 □

Lemma C.28. $\mathrm{Game}_{3-(k-1)-2}$ *and* Game_{3-k-1} *are indistinguishable under the DSG2 assumption and collision resistant property of* H*. That is, for every adversary* $\mathscr{A}$*, there exists a PPT algorithm* $\mathscr{B}$ *such that*

$$|\mathsf{Adv}^{3-(k-1)-2}_{\mathscr{A},\mathrm{PSC}}(\kappa) - \mathsf{Adv}^{3-k-1}_{\mathscr{A},\mathrm{PSC}}(\kappa)| \leq \mathsf{Adv}^{\mathrm{DSG2}}_{\mathscr{B}}(\kappa) + \mathsf{Adv}^{\mathrm{CRH}}_{\mathscr{B}}(\kappa) \text{ for } 1 \leq k \leq \nu_2.$$

Proof. In both the games, the vTextKey for verifying the forgery is of sf-type 1, the queried key is sf-type 3, the unsigncrypt oracle queries are answered by vTextKeys of sf-type II and the ciphertext part in all the replied signcryptions to the signcrypt oracle queries are normal. The only difference between the games is the form of the queried signcryption to the k^{th} signcrypt oracle query. Precisely, the signature part in the k^{th} replied signcryption is either normal or sf-type 1. By the event Forged, we have $\mathsf{vk}^* \neq \mathsf{vk}^{(k)}$, so $\hbar_s^* \neq \hbar_s^{(k)}$. Therefore, the proof can be done following the proof of Lemma A.6. □

Lemma C.29. Game_{3-k-1} *and* Game_{3-k-2} *are indistinguishable under the DSG2 assumption. That is, for every adversary* $\mathscr{A}$*, there exists a PPT algorithm* $\mathscr{B}$ *such that*

$$|\mathsf{Adv}^{3-k-1}_{\mathscr{A},\mathrm{PSC}}(\kappa) - \mathsf{Adv}^{3-k-2}_{\mathscr{A},\mathrm{PSC}}(\kappa)| \leq \mathsf{Adv}^{\mathrm{DSG2}}_{\mathscr{B}}(\kappa) \text{ for } 1 \leq k \leq \nu_2.$$

Proof. The only difference between the games is the form of the replied signcryption to the k^{th} signcrypt oracle query. Precisely, the signature part is either sf-type 1 or sf-type 2. We refer to proof of Lemma A.7. □

Lemma C.30. $\mathrm{Game}_{3-\nu_2-2}$ *and* Game_4 *are indistinguishable under the DSG2 assumption and collision resistant property of* H*. That is, for every adversary* $\mathscr{A}$*, there exists a PPT algorithm* $\mathscr{B}$ *such that*

$$|\mathsf{Adv}^{3-\nu_2-2}_{\mathscr{A},\mathrm{PSC}}(\kappa) - \mathsf{Adv}^{4}_{\mathscr{A},\mathrm{PSC}}(\kappa)| \leq \mathsf{Adv}^{\mathrm{DSG2}}_{\mathscr{B}}(\kappa) + \mathsf{Adv}^{\mathrm{CRH}}_{\mathscr{B}}(\kappa).$$

Proof. In both the games, the queried key is sf-type 3, the unsigncrypt queries are answered by vTextKeys of sf-type II, the replied signcryptions to the signcrypt oracle queries are of sf-type II and the vText part in the vTextKey for verifying the forgery is of sf-type 1. The only difference between the games is the form of alt-key in the vTextKey for verifying the forgery, i.e., it is either normal or sf-type 1. Therefore, the proof can be done following the proof of Lemma A.6. The collision resistant property of H will be used only to guarantee $\hbar_s^* \neq \hbar_e^*$ (following Remark C.1) in the construction of vTextKey for verifying the forgery. □

Lemma C.31. Game_4 *and* Game_5 *are indistinguishable under the DSG3 assumption. That is, for every adversary* $\mathscr{A}$*, there exists a PPT algorithm* $\mathscr{B}$ *such that*

$$|\mathsf{Adv}^{4}_{\mathscr{A},\mathrm{PSC}}(\kappa) - \mathsf{Adv}^{5}_{\mathscr{A},\mathrm{PSC}}(\kappa)| \leq \mathsf{Adv}^{\mathrm{DSG2}}_{\mathscr{B}}(\kappa).$$

Proof. The only difference between the games is the form of the alt-key in the vTextKey for verifying the forgery, i.e., the alt-key is either normal or sf-type 1. We refer to proof of Lemma A.7. □

Lemma C.32. Game$_5$ *and* Game$_{Final}$ *are indistinguishable under the DSG3 assumption. That is, for every adversary* $\mathscr{A}$*, there exists a PPT algorithm* $\mathscr{B}$ *such that*

$$|\mathsf{Adv}^{5}_{\mathscr{A},\mathrm{PSC}}(\kappa) - \mathsf{Adv}^{Final}_{\mathscr{A},\mathrm{PSC}}(\kappa)| \leq \mathsf{Adv}^{\mathrm{DSG3}}_{\mathscr{B}}(\kappa).$$

Proof. The only difference between the games is the form of the vText in the vTextKey for verifying the forgery, i.e., the vText is either sf-type 1 or sf-type 2. For proof, we refer to Lemma B.11. □

Theorem C.33. *Let* P *be a pair encoding scheme for a predicate* $\sim$ *which satisfies* **Conditions 3.1** *and* $\sim$ *is domain-transferable. Suppose* P *has the* PMH *security, the assumptions, DSG1, DSG2 and DSG3 hold in* $\mathcal{J}$*, the one-time signature scheme* OTS *has strong unforgeability and* H *is a collision resistant hash function, then the proposed predicate signcryption scheme* PSC *in Section 7 for the predicate* $\sim$ *is strongly existential unforgeable in adaptive-predicates model.*

Proof. Similar to the proof of Theorem C.20. □