Attribute-Based Signcryption: Signer Privacy, Strong Unforgeability and IND-CCA Security in Adaptive-Predicates Model (Extended Version)*

Tapas Pandit[1]†, Sumit Kumar Pandey[2], and Rana Barua[1]
[1]Stat-Math Unit, Indian Statistical Institute, Kolkata, India
{tapasgmmath, ranabarua.isi}@gmail.com
[2]School of Physical and Mathematical Sciences
Nanyang Technological University, Singapore
emailpandey@gmail.com

Abstract

Attribute-Based Signcryption (ABSC) is a natural extension of Attribute-Based Encryption (ABE) and Attribute-Based Signature (ABS), where one can have the message confidentiality and authenticity together. Since the signer privacy is captured in security of ABS, it is quite natural to expect that the signer privacy will also be preserved in ABSC. In this paper, first we propose an ABSC scheme which is *weak existential unforgeable* and *IND-CCA* secure in *adaptive-predicates* models and, achieves *signer privacy*. Then, by applying strongly unforgeable one-time signature (OTS), the above scheme is lifted to an ABSC scheme to attain *strong existential unforgeability* in *adaptive-predicates* model. Both the ABSC schemes are constructed on common setup, i.e the public parameters and key are same for both the encryption and signature modules. Our first construction is in the flavor of $\mathscr{C}t\mathscr{E}\&\mathscr{S}$ paradigm, except one extra component that will be computed using both signature components and ciphertext components. The second proposed construction follows a new paradigm (extension of $\mathscr{C}t\mathscr{E}\&\mathscr{S}$), we call it "Commit then Encrypt and Sign then Sign" ($\mathscr{C}t\mathscr{E}\&\mathscr{S}t\mathscr{S}$). The last signature is generated using a strong OTS scheme. Since, the non-repudiation is achieved by $\mathscr{C}t\mathscr{E}\&\mathscr{S}$ paradigm, our systems also achieve the same.

Keywords: Attribute-based encryption, Attribute-based signature, Attribute-based signcryption

1 Introduction

In the last couple of years, attribute-based encryption (ABE) has become a privilege way for encrypting a message for many users. In this encryption, a message is encoded with a policy and a key is labeled with a set of attributes. This form of ABE is known as ciphertext-policy attribute-based encryption (CP-ABE) and in its dual form, key-policy attribute-based encryption (KP-ABE), the role of policy and the set of attributes are interchanged. Since its introduction (Fuzzy Identity-Based Encryption) [43], many schemes have been studied, some of them are CP-ABE [5, 25, 35, 48, 24], some of them are KP-ABE [18, 38, 25, 35, 2], most of them are selectively secure under chosen plaintext attack (CPA) [18, 48, 38, 2], few of them are adaptively secure under CPA [35, 25, 37] and very few of them are secure under chosen ciphertext attack (CCA) [35] for general policies. But, there are techniques [11, 9, 49, 34] to convert a

Journal of Internet Services and Information Security (JISIS), volume: 6, number: 3 (August 2016), pp. 61-113

*This article is an extended version of the paper [39] that appeared in the Proceedings of 8th International Conference, ProvSec 2014, LNCS 8783, pp. 274-290, Springer. In the original version [39], the unforgeability of ABSC schemes was proven without giving the unsigncrypt oracle access to the adversary. This paper is extended by providing the proof of unforgeability in presence of unsigncrypt oracle access to the adversary.

†Corresponding author: Stat-Math Unit, Indian Statistical Institute, Kolkata, Pin - 700108, India, Tel: +91-33-2575-3461

CPA secure scheme to CCA secure one. However, the schemes that are adaptively secure under CCA in the standard model seem to be more powerful.

Side by side with ABE, attribute-based signature (ABS) also draws much attention due to its versatility. Unlike the traditional signature scheme, it captures unforgeability for a policy (group of users) and signer privacy. Similar to ABE, in attribute-based signature a message is signed under a policy and a key is associated with a set of attributes. We call this form of ABS as signature-policy(SP)-ABS [36, 29, 26, 30] and its dual form, where the role of the policy and the set of attributes are reversed, is called KP-ABS [44]. Similar to the traditional signature, ABS can be weak existential unforgeable[36, 29, 30, 26] or strong existential unforgeable under chosen message attack (CMA). Most of the ABS [44] proposed so far are weak existential unforgeable. But, by a simple technique [22] one can obtain strongly unforgeable signature scheme from weak unforgeable scheme. Since here the message is signed under a policy, similar to ABE there are two types of unforgeability, selective-predicate [44, 26] and adaptive-predicate [36, 29, 30].

Zheng [50] introduced the concept of *signcryption* that provides an efficient way of achieving message confidentiality and authenticity together. However, the author did not provide any formal security proof of the proposed signcryption as no security model was available. Later, Baek, Steinfeld and Zheng [3] formalized a security model and gave a formal security proof of the signcryption scheme [50] in this model. Then An, Dodis and Rabin [1] proposed generic constructions of signcryption from the primitives, encryption, signature and commitment schemes. Their constructions follow three paradigms, "Sign then Encrypt" ($\mathscr{S}t\mathscr{E}$), "Encrypt then Sign" ($\mathscr{E}t\mathscr{S}$) and "Commit then Encrypt and Sign" ($\mathscr{C}t\mathscr{E}\&\mathscr{S}$). As compared to $\mathscr{S}t\mathscr{E}$ and $\mathscr{E}t\mathscr{S}$ paradigms, $\mathscr{C}t\mathscr{E}\&\mathscr{S}$ has an advantage that in Signcrypt (resp. Unsigncrypt) both the routines, Encrypt and Sign (resp. Decrypt and Ver) can be executed in parallel, i.e., in $\mathscr{C}t\mathscr{E}\&\mathscr{S}$ paradigm both Signcrypt and Unsigncrypt run faster as compared to other two paradigms. The authors also discussed a natural paradigm of signcryption, called "Encrypt and Sign" ($\mathscr{E}\&\mathscr{S}$). In this paradigm, a signcryption consists of a ciphertext and a signature obtained by applying $\mathscr{E}$ and $\mathscr{S}$ on the message respectively. One major drawback of this paradigm is that it never provides IND-CPA security. For all these paradigms, KeyGen algorithms of the underlying encryption and signature schemes are run independently to generate receiver's key pair and sender's key pair respectively. The generic constructions in [1] were proven in two users model in PKI setting, but using some minor modification one can have the same security in multi user setting. Subsequently, several signcryption schemes [33, 32, 27, 28, 15, 13, 10] have been studied either in PKI setting or in IBE setting.

COMBINED-SETUP. It is a common practice to use the independent key distributions for different cryptographic primitives, e.g., encryption and signature schemes used in combination. To prove the security of one primitive, other primitives are assumed to be independent. One possible reason of such practice may be reducing the security threat because if the keys for the primitives are not independent, then their joint information may jeopardize the security goal of the target primitive. Haber and Pinkas [19] formally investigated that there are schemes, viz., encryption and signature schemes which can be used in combination. The security of each primitive in combination is same as their individual security even if keys for the primitives are dependent, in particular identical. The authors called the schemes in combination as *combined public-key scheme*. In this combined scheme, the Encrypt and Decrypt (resp. Sign and Ver) algorithms of the encryption (resp. signature) scheme are kept unchanged. This scheme is nothing but the $\mathscr{E}\&\mathscr{S}$ paradigm with keys for encryption and signature schemes being identical. The security model is called joint security of the combined public-key scheme. The joint security model is defined similarly as the individual security model of encryption and signature schemes, except the adversary gets access to both the oracles, sign oracle and decrypt oracle. However, in joint security model, ciphertext indistinguishability is not possible if a signature of the challenge message is additionally given with the challenge ciphertext.

Later, Vasco, Hess and Steinwandt [45] studied the combined public-key scheme in ID-based setting, where the combined scheme has a single setup algorithm and key-gen algorithm. In other words, primitive IBS and IBE scheme have identical setup and key-gen algorithms. They combined IBE [7] and IBS [21] in the joint security model, but they assumed random oracles. Then Paterson et al. [40] showed a construction of combined public-key scheme in the standard model using the primitives, an IBE, a short signature and a data encapsulation mechanism. Recently, Chen et al. [12] extended the concept of combined public-key scheme from ID-based setting to attribute-based setting. More specifically, the authors combined an ABE scheme and ABS scheme in the joint security model. Similar to [45], the primitives, ABS and ABE used in [12] have identical setup and key-gen algorithms. This setup algorithm is run only once to generate $\mathscr{PP}$ which will work for both, ABS and ABE. For each user, this key-gen algorithm is run only once to generate a key which will work for both, ABS and ABE. We call the above keys and $\mathscr{PP}$ generation process as *combined-setup*. 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 $\mathscr{PP}$ for the individual primitives, ABS and ABE. We note that the same keys and $\mathscr{PP}$ 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 $\mathscr{PP}$ than the schemes in independent-setup.

Signcryption in attribute-based setting is called attribute-based signcryption (ABSC) which captures the functionalities of both the primitives, ABS and ABE. Similar to ABS and ABE, there are two forms[1] of ABSC, signcryption-policy attribute-based signcryption (SCP-ABSC) and key-policy attribute-based signcryption (KP-ABSC). In SCP-ABSC, a signcryption is associated with two policies (for receiver and sender) and a key is labeled with two sets of attributes (for receiver and sender). In KP-ABSC, the role of policies and sets of attributes are interchanged. It was Gagné, Narayan and Safavi-Naini [17] who first proposed an ABSC scheme for threshold policies. Since then, many attribute-based signcryption schemes [46, 16, 12, 42] have been proposed in the literature. None of these schemes follows $\mathscr{CE}\&\mathscr{S}$ paradigm. Among them the only scheme that supports signer-privacy, non-repudiation and combined-setup is [12]. However, none of these schemes is secure in adaptive-predicates models. This leads us to ask the following question:

Can an attribute-based signcryption scheme be constructed which is secure in adaptive-predicates models and at the same time, it supports the aforementioned features?

Our Result. Affirmatively, we answer the above question. We propose two constructions of ABSC in signcryption-policy which are shown to be secure in adaptive-predicates models. Further, the schemes attain the above features, i.e., signer-privacy, non-repudiation, combined-setup and run in $\mathscr{CE}\&\mathscr{S}$ paradigm. Our first construction (given in Section 5) achieves weak existential unforgeability and IND-CCA security in adaptive-predicates models, and signer-privacy. The second construction (given in Section 7) is obtained by applying OTS to the first construction to guarantee strong unforgeability in adaptive-predicates model.

A detailed comparison of performance and security features between our second ABSC scheme and others is given in Table 1 and Table 2. The following are the specifications for Table 1. Let CS denote combined-setup. The schemes supporting combined-setup have a single key-extraction algorithm and in this case, we use A to indicate user's set of attributes. Otherwise two set of attributes, A_s and A_e are used respectively for signcrypt and unsigncrypt. In later case, the individual key sizes are separated by comma. Let ℓ_s and ℓ_e respectively denote the sizes of sender's policy Γ_s and receiver's policy Γ_e. The symbol φ stands for the maximum number of repetitions of an attribute in an access policy. Let ω_s, ω_e and t respectively represent sender's set of attributes, receiver's set of attributes and threshold value used in [17]. $\mathscr{U}_e$ and $|\mathsf{vk}|$ respectively denote the attribute universe involved in encryption and length of

[1] Other forms of ABSC may be possible, but we are not interested on them.

verification key for OTS. Let $\theta_e = 2|A_e| + 2|\mathsf{vk}| + 1$. The total size of commitment and OTS is denoted by $\wp$. Let $|\mathscr{I}_B|$ (resp. $|\mathscr{I}_A|$) be the number of rows in the policy Γ_e (resp. Γ_s) labeled by the attribute set B (resp. A). Let d be the length of digest of the hash function H_2 involved in the scheme [42]. The symbol $\mathscr{O}$ stands for asymptotic Big-O notation. The key size and signcryption size are measured by the number of group elements involved in the key and signcryption respectively. The time of signcrypt is measured by the number of exponentiations to construct a signcryption, whereas the time for unsigncrypt is both number of exponentiations and number of pairings.

The abbreviations, SAS, RAS, Unforge., Conf., NR, SP, APMs, NK, MAT, MSP, AGW, KP and SCP found in Table 2 stand for sender's access structure, receiver's access structure, unforgeability, confidentiality of message, non-repudiation, signer-privacy, adaptive-predicates models, not known, monotone access tree, monotone span program, AND-gate with wildcard respectively, key-policy and signcryption-policy.

ABSC	CS	Key size	Signcryption size	Signcrypt time	Unsigncrypt time
[17]	No	$2\|A_s\|, 3\|A_e\|$	$\mathscr{O}(\|\omega_s\| + \|\omega_e\|)$	$\mathscr{O}(\|\omega_s\| + \|\omega_e\|)$	$\mathscr{O}(\|\omega_s\| + t)$
[16]	No	$2\|A_s\|, \theta_e$	$\mathscr{O}(\ell_s + \|\mathscr{U}_e\| + \|\mathsf{vk}\|)$	$\mathscr{O}(\ell_s + \|\mathscr{U}_e\| + \|\mathsf{vk}\|)$	$\mathscr{O}(\ell_s + \|\mathscr{U}_e\| + \|\mathsf{vk}\|)$
[12]	Yes	$\varphi\|A\| + 2$	$2\ell_s + \ell_e + 4$	$\mathscr{O}(\ell_s) + \mathscr{O}(\ell_e)$	$\mathscr{O}(\ell_s) + \mathscr{O}(\|\mathscr{I}_B\|)$
[42]	No	$\|\mathscr{U}_s\|\ell_s, \|\mathscr{U}_e\|\ell_e$	6	$\mathscr{O}(\mathscr{I}_A) + \mathscr{O}(d)$	$\mathscr{O}(\mathscr{I}_B) + \mathscr{O}(d)$
Our	Yes	$\varphi\|A\| + 2$	$2\ell_s + 2\ell_e + 5 + \wp$	$Max\{\mathscr{O}(\ell_s), \mathscr{O}(\ell_e)\}$	$Max\{\mathscr{O}(\ell_s), \mathscr{O}(\|\mathscr{I}_B\|)\}$

Table 1: Performance of our second ABSC scheme

ABSC	Type	SAS	RAS	Unforge.	Conf.	NR	SP	APMs
[17]	KP	Threshold	Threshold	wUF-CMA	IND-CCA	No	NK	No
[16]	SCP	MAT	AGW	sUF-CMA	IND-CCA	Yes	No	No
[12]	SCP	MSP	MSP	sUF-CMA	IND-CCA	Yes	Yes	No
[42]	KP	MSP	MSP	sUF-CMA	IND-CCA	Yes	Yes	No
Our	SCP	MSP	MSP	sUF-CMA	IND-CCA	Yes	Yes	Yes

Table 2: Security features of our second ABSC scheme

Our Approach. The basic paradigm, we follow in the constructions is $\mathscr{C}t\mathscr{E}\&\mathscr{S}$. In this paradigm, a message m is first committed to $(\mathsf{com}, \mathsf{decom})$, then the commitment part com and decommitment part decom are respectively signed to δ and encrypted to CT in parallel to produce the signcryption $\mathsf{U} = (\mathsf{com}, \delta, \mathsf{CT})$. Similarly, the algorithms, Ver (to verify δ) and $\mathsf{Decrypt}$ (to get the decom) run in parallel in $\mathsf{Unsigncrypt}$ to extract the underlying message as $m \longleftarrow \mathsf{Open}(\mathsf{com}, \mathsf{decom})$.

For both the constructions, we use the CP-ABE scheme of [25] as a primitive encryption scheme. The scheme was constructed using composite order bilinear groups. Since, one of our focus is to construct SCP-ABSC scheme in combined-setup, we need an SP-ABS scheme which has the same $\mathscr{P}\mathscr{P}$ and key distribution as that of CP-ABE [25]. So, we first construct an SP-ABS scheme (refer to Section 3) which is suitable with the CP-ABE scheme to support combined-setup. The ABS scheme is shown (see Section 4) to be weakly unforgeable in adaptive-predicate model and attains signer-privacy. For proving unforgeability, we utilize the dual system methodology of Waters [47]. We do not use the dual system proof technique [47] directly as described in the context of ABE [25], but we utilize its signature variant [36]. In signature variant of dual system methodology, a signature is verified under a set of components, called verification text (in short vText). In brief, by applying hybrid arguments over a sequence of games, we reach to a final game. In final game, a particular component of vText for the forgery is changed to a uniformly and independently random element over some integer modulo group. This implies that the forgery will be invalid with respect to the changed vText.

The Signcrypt algorithm of the first scheme (given in Section 5) works in the following way. Suppose Alice wants to signcrypt a message m under a policy, Γ_s and Γ_e. She first runs $(\mathsf{com}, \mathsf{decom}) \longleftarrow \mathsf{Commit}(m)$. Then generates a signature δ for the message $\mathsf{com}||\Gamma_e$ under the sender's policy Γ_s using the proposed SP-ABS scheme. She also encrypts decom under a receiver's policy Γ_e using the primitive CP-ABE scheme and let C be the corresponding ciphertext. Actually, the signature δ and ciphertext C are generated in parallel. Now Alice binds com, C and δ through a collision resistant hash function H to $\hbar := H(\mathsf{com}, \mathsf{C}, \delta)$. Then she encodes $\hbar$ to an additional component C_{add} using a secret s involved in the encrypt algorithm of the primitive CP-ABE and Boneh-Boyen hash technique [6]. Finally, Alice outputs a signcryption $\mathsf{U} := (\mathsf{com}, \delta, \mathsf{CT} := (\mathsf{C}, C_{\mathsf{add}}))$. The component C_{add} prevents an adversary $\mathscr{A}$ from changing a given signcryption to another signcryption. Therefore, the component C_{add} is very crucial for achieving IND-CCA security in adaptive-predicates model. The Unsigncrypt algorithm works in similar fashion, where the algorithms, Ver of ABS and Decrypt of ABE run in parallel. Also a special checking for the component C_{add} is involved in Unsigncrypt algorithm. The proposed scheme is shown (see Section 6) to be weakly unforgeable and IND-CCA secure in adaptive-predicates models and perfectly-private. For showing weak unforgeability and IND-CCA security, we need relaxed-binding and hiding properties of the underlying commitment scheme respectively.

We note that in the first scheme, the adversary could modify a replied signcryption for a message (m, Γ_s, Γ_e) as follows. Since $\mathscr{A}$ has access to key $\mathscr{SK}_A$ with $\Gamma_e(A) = \mathsf{True}$, it can recover decom from CT and then re-encrypts it to get modified (new) signcryption for the same message (m, Γ_s, Γ_e). Therefore, the first scheme does not provide strong unforgeability. Our second scheme (see Section 7) is constructed by combining the first scheme and a strong OTS scheme to ensure strong unforgeability. Essentially, the components, $\hbar$, C_{add}, Γ_e and Γ_s are signed together using a strong OTS scheme to guarantee that a signcryption for a message can not be altered even if the adversary knows the unsigncrypt key. The scheme has (see Section 8) strong unforgeability and IND-CCA security in adaptive-predicates models and signer-privacy. The strong unforgeability of this SCP-ABSC scheme no more relies on relaxed-binding property of the underlying commitment scheme. But, for showing non-repudiation, we require the relaxed-binding property of the commitment scheme. There are many commitment schemes [14, 20, 41] suitable for our systems, but we use them as black box in our constructions.

Unforgeability of the proposed schemes is proven assuming the primitive ABS as black-box in the constructions. In this proof, signcrypt oracle queries are answered using the resource, sign oracle of ABS. But, the adversary is not allowed to ask the unsigncrypt oracle query as the simulator is incapable to answer. On the other hand, IND-CCA security of the constructions is proven using decisional subgroup assumptions, where neither ABS nor ABE is used as black-box. The adversary is free to query to the key oracle, signcrypt oracle and unsigncrypt oracle of its own choice. Again, we use the dual system methodology of Waters [47] in a novel way using composite order bilinear groups.

As we mentioned that the unforgeability of the proposed schemes (appeared in [39]) are proven (see Sections 6 and 8) without giving unsigncrypt oracle access to the adversary $\mathscr{A}$. However, in Section 9 we provide the proof of unforgeability of the second scheme (in Section 7), where $\mathscr{A}$ is given access to unsigncrypt oracle.

Related Works. Gagné, Narayan and Safavi-Naini [17] initiated a formal study of attribute-based signcryption. The receiver's policies and sender's policies they considered are the threshold policies. In their construction, Fuzzy IBE[43] and a new efficient threshold ABS were used as primitive encryption scheme and signature scheme respectively. Later, Emura, Miyaji, and Rahman [16] proposed a dynamic attribute-based signcryption, where access structures of the encryptor can be changed without re-issuing the secret keys of the users. The sender's access policy and receiver's policy in the above scheme are represented respectively by monotone access tree and AND-gate with wildcard. Both the schemes were shown to be secure in the selective-predicate models (Definitions 2.11 and 2.15). However, the signer-

privacy was not discussed in former scheme, whereas later ABSC scheme lacks this property.

Wang and Huang [46] proposed an ABSC scheme in signcryption-policy form. The receiver's policies and sender's policies, that their ABSC scheme supports, are monotone access trees. The confidentiality and unforgeability of the scheme were proven in the adaptive-predicates models. However, confidentiality of the scheme was proven in generic group model without giving access of unsigncrypt oracle to the adversary. The unforgeability of the scheme was proven in the random oracle model.

Chen et al. [12] proposed a combined public-key scheme in attribute-based setting. In this combined scheme, the underlying ABE and ABS schemes have identical public parameters and key distribution. Their scheme is based on the construction of Waters [48]. The scheme was shown to be selectively secure in the joint security model. Finally, the authors showed a generic extension from this combined scheme to attribute-based signcryption in $\mathscr{S}t\mathscr{E}$ paradigm. Both the policies considered in their scheme are monotone span programs. This proposed signcryption scheme entertains the signer-privacy. The confidentiality and unforgeability of the proposed ABSC were proven in the selective-predicate models.

Recently, Rao and Dutta [42] proposed a KP-ABSC scheme with the constant size signcryption. The number of pairings involved in unsigncrypt is 6, but the key size is comparatively large. The receiver's policy and sender's polices represented in their scheme are monotone span programs. Both the confidentiality and unforgeability of their proposed scheme were proven in the selective-predicate models. The signer privacy is maintained in the proposed signcryption.

Discussion 1.1. We note that our proposed solution is not generic. One may think that using the generic construction of An, Dodis and Rabin [1] such constructions are possible, but this is not the case for the following reasons. First, we emphasize that $\mathscr{C}t\mathscr{E}\&\mathscr{S}$ paradigm preserves only weak unforgeability and IND-gCCA (see footnote [2]). In contrast, our proposed scheme attains both strong unforgeability and IND-CCA security in adaptive-predicates models. Secondly, our solution supports the combined-setup, where the primitive ABS and ABE have the identical public parameters and keys distribution. The original construction [1] of signcryption in $\mathscr{C}t\mathscr{E}\&\mathscr{S}$ paradigm assumes the primitive encryption and signature schemes to be black-boxes, i.e., independent-setup. So, the security proof of [1] for $\mathscr{C}t\mathscr{E}\&\mathscr{S}$ paradigm can not carry through in combined-setup. Our first construction follows the $\mathscr{C}t\mathscr{E}\&\mathscr{S}$ paradigm, but in addition to $\mathscr{C}t\mathscr{E}\&\mathscr{S}$, an extra component is computed using signature and ciphertext components of signcryption. Our second construction follows a new paradigm what we call "Commit then Encrypt and Sign then Sign" ($\mathscr{C}t\mathscr{E}\&\mathscr{S}t\mathscr{S}$) paradigm, where the last sign is obtained using a strong OTS.

2 Preliminaries

Basic notations, composite order bilinear groups, hardness assumptions, definitions of access structure and linear secret sharing scheme, syntaxes and security definitions of commitment scheme, OTS, ABS and ABSC are provided in this section.

2.1 Notations

For a set X, $x \xleftarrow{R} X$ denotes that x is randomly picked from X according to the distribution R. Likewise, $x \xleftarrow{U} X$ indicates x is uniformly selected from X. For an algorithm A and variables x, y, the notation $x \longleftarrow A(y)$ (resp. $A(y) \longrightarrow x$) carries the meaning that when A is run on the input y, it outputs x. The symbol, 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

[2] IND-gCCA is a weaker security notion than IND-CCA. For details, reader may consult [1].

$x_1 \longleftarrow A_1(y_1) \parallel \ldots \parallel x_n \longleftarrow A_n(y_n)$ 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 a vector $\mathbf{x}$, the i^{th} component is denoted by x_i. For $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_N^n$, we define $< \mathbf{x}, \mathbf{y} >:= \sum_{i=1}^{n} x_i \cdot y_i$. For $S \subset \mathbb{Z}_N^n$ and $\alpha \in \mathbb{Z}_N^n$, we define $\alpha + S := \{\alpha + \beta \mid \beta \in S\}$. For a matrix $\mathbf{M}_e$ (resp. $\mathbf{M}_s$), the symbol $\mathbf{M}_e^{(i)}$ (resp. $\mathbf{M}_s^{(i)}$) represents the i^{th} row of the matrix $\mathbf{M}_e$ (resp. $\mathbf{M}_s$). The subscripts i and superscript (i) will be used for indexing. The notations $\mathbf{1}$ and $\mathbf{0}$ stand for $(1, 0, \ldots, 0)$ and $(0, \ldots, 0)$ respectively. For better understanding the schemes, we use two subscripts, s and e respectively for encryption and signature.

2.2 Composite Order Bilinear Groups

Composite order bilinear groups [8, 23] are defined to be a tuple $\mathscr{J} := (N := p_1 p_2 p_3, \mathbb{G}, \mathbb{G}_T, e)$, 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 $\mathscr{G}_{\mathsf{cbg}}$ denote an algorithm which takes 1^κ as a security parameter and returns a description of composite order bilinear groups $\mathscr{J} = (N = p_1 p_2 p_3, \mathbb{G}, \mathbb{G}_T, e)$. 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 three distinct primes, p_1, p_2 and p_3, a cyclic group $\mathbb{G}$ of order $N = p_1 p_2 p_3$, 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 $X = X_1 X_2 X_3$, 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., $X_i = X\big|_{\mathbb{G}_{p_i}}$. 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 [25] for 3 primes, DSG1, DSG2 and DSS3 in composite order bilinear groups. Let $\mathscr{J} := (N = p_1 p_2 p_3, \mathbb{G}, \mathbb{G}_T, e) \xleftarrow{\mathrm{U}} \mathscr{G}_{\mathsf{cbg}}(1^\kappa)$ be the common parameters for each assumptions. In the following, we define instance for each assumption.

- DSG1. Let $g \xleftarrow{\mathrm{U}} \mathbb{G}_{p_1}; Z_3 \xleftarrow{\mathrm{U}} \mathbb{G}_{p_3}; T_0 \xleftarrow{\mathrm{U}} \mathbb{G}_{p_1}; T_1 \xleftarrow{\mathrm{U}} \mathbb{G}_{p_1 p_2}$. Define $\mathscr{D} := (\mathscr{J}, g, Z_3)$.
- 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 $\mathscr{D} := (\mathscr{J}, g, Z_1 Z_2, W_2 W_3, Z_3)$.
- 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 $\mathscr{D} := (\mathscr{J}, g, g^\alpha Y_2, g^s W_2, g_2, Z_3)$.

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

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

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

Below we define a factorization problem for the composite numbers of the form, $N = p_1p_2p_3$ and p_1, p_2 and p_3 are three distinct primes. Let CNG be an algorithm for generating the composite number of the above form. The algorithm CNG takes as input 1^κ and returns a composite number of length polynomial to κ.

Definition 2.2 (Factorization Problem)**.** A factorization problem for product of three primes is said to be hard if for all PPT algorithms $\mathscr{A}$, the advantage

$$\mathsf{Adv}_{\mathscr{A}}^{\mathrm{Factor}}(\kappa) := \Pr\left[(q|N) \wedge (q \neq 1) \wedge (q \neq N) \;\middle|\; \begin{matrix} N := p_1p_2p_3 \longleftarrow \mathsf{CNG}(1^\kappa); \\ q \longleftarrow \mathscr{A}(1^\kappa, N) \end{matrix}\right]$$

is negligible function in κ.

Proposition 2.1. *The hardness of DSG2 assumption implies the hardness of factorization problem.*

Proof. We establish a PPT algorithm $\mathscr{B}$ for breaking the DSG2 assumption using an adversary $\mathscr{A}$ for breaking the factorization problem. $\mathscr{B}$ is given an instance of DSG2, $(\mathscr{J}, g, Z_1Z_2, W_2W_3, Z_3, T_\beta)$, where $\mathscr{J} = (N = p_1p_2p_3, \mathbb{G}, \mathbb{G}_T, e)$ with $\beta \xleftarrow{\mathrm{U}} \{0,1\}$. Then N is given to $\mathscr{A}$. Let F be the non-trivial factor returned by $\mathscr{A}$. Then $F = p_1$ or p_2 or p_3 or p_1p_2 or p_1p_3 or p_2p_3, i.e., F is either a prime or the product of two distinct primes. Using the parameters of the given instance of DSG2, $\mathscr{B}$ recognizes the exact form of F as follows. If $g^F = 1_\mathbb{G}$, then $F = p_1$ or p_1p_2 or p_1p_3 else $F = p_2$ or p_3 or p_2p_3.

– **(Case: $g^F = 1_\mathbb{G}$).** If $(Y_1Y_2)^F = 1_\mathbb{G}$, then $F = p_1p_2$ else if $(Z_3)^F = 1_\mathbb{G}$ then $F = p_1p_3$ else $F = p_1$.

– **(Case: $g^F \neq 1_\mathbb{G}$).** If $(W_2W_3)^F = 1_\mathbb{G}$, then $F = p_2p_3$, else if $(Z_3)^F = 1_\mathbb{G}$, then $F = p_3$ else $F = p_2$.

How $\mathscr{B}$ breaks the instance of DSG2 is shown only for three forms of F, viz., p_1, p_3 and p_1p_3 as the rest of the cases are handled by assigning $F := \frac{N}{F}$.

– $F = p_1$. It checks $e((Y_1Y_2)^F, T_\beta) \stackrel{?}{=} 1$. If equality holds, it returns 0 else 1.

– $F = p_3$. It checks $e((W_2W_3)^F, T_\beta) \stackrel{?}{=} 1$. If equality holds, it returns 0 else 1.

– $F = p_1p_3$. It checks $(T_\beta)^F \stackrel{?}{=} 1_\mathbb{G}$. If equality holds, it returns 0 else 1.

□

Proposition 2.2. *Suppose the DSG2 assumption holds in $\mathscr{J} = (N = p_1p_2p_3, \mathbb{G}, \mathbb{G}_T, e)$. Let $a, b \in \mathbb{Z}_N$. Then for $X_1, X_2 \in \mathbb{Z}_N$ with $X_1 \neq X_2$, $aX_1 + b$ and $aX_2 + b$ are uniformly and independently distributed over $\mathbb{Z}_{p_2}$.*

Proof. Using the fact of [23], we have $aX_1 + b$ and $aX_2 + b$ are uniformly and independently distributed over $\mathbb{Z}_{p_2}$ if $X_1 \not\equiv X_2 \mod p_2$. Therefore, it is sufficient to show that $X_1 \not\equiv X_2 \mod p_2$. Indeed, if $X_1 - X_2 \equiv 0 \mod p_2$, we can find a non-trivial factor $F := \gcd(X_1 - X_2, N)$ of N with $p_2|F$. Then, using Proposition 2.1, we can break the hardness of DSG2 assumption, which is a contradiction. □

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

For correctness, it is required that[3] $\mathsf{Open}(\mathsf{Commit}(m)) = m$ for all message $m \in \mathscr{M}$, where $\mathscr{M}$ is the message space.

2.5 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:

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

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

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

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

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

Remark 2.1. It is immediate that the relaxed-binding property is weaker than the binding property.

There are many commitment schemes [14, 20, 41] available in the literature. However, in this paper, we use the commitment scheme as a black-box in the proposed constructions.

2.6 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.

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

2.7 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.

2.8 Access Structure and Linear Secret Sharing Scheme

Definition 2.3 (Access Structure)**.** Let $\mathscr{P} = \{P_1,\ldots,P_n\}$ be a set of parties. A collection $\Gamma \subset 2^{\mathscr{P}}$ is said to be monotone if Γ is closed under superset, i.e, if $\forall\ B,C$: if $B \in \Gamma$ and $B \subset C$, then $C \in \Gamma$. An access structure (resp. monotone access structure) is a collection (respectively, monotone collection) Γ of non-empty subsets of $\mathscr{P}$, i.e., $\Gamma \subset 2^{\mathscr{P}} \setminus \{\emptyset\}$. The members of Γ are called authorized sets, and the sets not in Γ are called unauthorized sets.

The monotone access structures are represented by access trees and linear secret sharing schemes. We define linear secret sharing scheme (LSSS) of [48, 4] as follows.

Definition 2.4 (Linear Secret Sharing Scheme)**.** A secret sharing scheme Π over a set of parties $\mathscr{P}$ is called linear (over $\mathbb{Z}_p$) if

1. The shares for each party form a vector over $\mathbb{Z}_p$
2. There exists a matrix $\mathbf{M}$ of order $\ell \times r$, called share generating matrix. For all $i = 1,2,..,\ell$, the i^{th} row of $\mathbf{M}$ is labeled by a party $\rho(i)$ (ρ is a function from $\{1,\ldots,\ell\}$ to $\mathscr{P}$). For $\mathbf{v} = (s,t_2,\ldots,t_r)$, where $s \in \mathbb{Z}_p$ is the secret to be shared and $t_2,\ldots,t_r \xleftarrow{\mathrm{U}} \mathbb{Z}_p$, $\mathbf{Mv}$ is the vector of ℓ shares of the secret s according to Π. The share $s_i := (\mathbf{Mv})_i$ belongs to party $\rho(i)$.

Property of LSSS. Every LSSS according to the above definition enjoys linear reconstruction property defined as follows. Suppose Π is an LSSS for an access structure Γ. Let $A \in \Gamma$ be an authorized set and $\mathscr{I} := \{i \in [\ell] \mid \rho(i) \in A\}$. Then there exist constants $\{\alpha_i \in \mathbb{Z}_p\}_{i\in\mathscr{I}}$ such that

$$\sum_{i\in\mathscr{I}} \alpha_i \mathbf{M}_i = (1,0,\ldots,0) \tag{1}$$

where $\mathbf{M}_i$ is the i^{th} row of $\mathbf{M}$. Hence, if $\{s_i\}$ are valid shares of any secret s according to Π, then by taking inner product of the equation 1 with $\mathbf{v}$, we have $\sum_{i\in\mathscr{I}} \alpha_i s_i = s$. These constants $\{\alpha_i\}$ are computed in time polynomial in the size of share-generating matrix $\mathbf{M}$.

Span program. Given any monotone access structure over a set of parties $\mathscr{P}$, one can obtain the corresponding LSSS representation (denoted by $(\mathbf{M},\rho)$) by applying the technique of [4]. The function $\rho : [\ell] \to \mathscr{P}$ is called the row labeling function. For a monotone access structure Γ, the corresponding LSSS representation, $(\mathbf{M},\rho)$ is called monotone span program (MSP).

2.9 Signature-Policy Attribute-Based Signature

A signature-policy attribute-based signature (SP-ABS) scheme consists of four PPT algorithms - Setup, KeyGen, Sign and Ver.

- Setup: It takes a security parameter κ and a universe of attributes $\mathscr{U}$ as input, outputs public parameters $\mathscr{PP}$ and master secret key $\mathscr{MSK}$.
- KeyGen: It takes as input $\mathscr{PP}$, $\mathscr{MSK}$ and a set of attributes A and outputs a secret key $\mathscr{SK}_A$ corresponding to A.
- Sign: It takes $\mathscr{PP}$, a message $m \in \mathscr{M}$, a secret key $\mathscr{SK}_A$ and an associated policy Γ over $\mathscr{U}$ with $\Gamma(A) = \mathsf{True}$ and returns a signature δ.
- Ver: It receives $\mathscr{PP}$, a message $m \in \mathscr{M}$, a signature δ and a claimed associated policy Γ as input. It returns a boolean value 1 for acceptance or 0 for rejection.

Correctness. For all $(\mathscr{PP}, \mathscr{MSK}) \longleftarrow \mathsf{Setup}(1^\kappa, \mathscr{U})$, all $m \in \mathscr{M}$, all sets of attributes A, $\mathscr{SK}_A \longleftarrow \mathsf{KeyGen}(\mathscr{PP}, \mathscr{MSK}, A)$ and all policies Γ with $\Gamma(A) = \mathsf{True}$, it is required that

$$\mathsf{Ver}(\mathscr{PP}, m, \mathsf{Sign}(\mathscr{PP}, m, \mathscr{SK}_A, \Gamma), \Gamma) = 1.$$

Remark 2.2. As in ABS of [31], we assume that signer sends both signature and policy Γ to receiver.

2.10 Security of SP-ABS

Definition 2.5 (Signer Privacy)**.** An SP-ABS scheme is said to be perfectly private if for all $(\mathscr{PP}, \mathscr{MSK}) \longleftarrow \mathsf{Setup}(1^\kappa, \mathscr{U})$, all $A_1, A_2 \subset \mathscr{U}$, $\mathscr{SK}_{A_1} \longleftarrow \mathsf{KeyGen}(\mathscr{PP}, \mathscr{MSK}, A_1)$, $\mathscr{SK}_{A_2} \longleftarrow \mathsf{KeyGen}(\mathscr{PP}, \mathscr{MSK}, A_2)$, all $m \in \mathscr{M}$, and all policies Γ with $\Gamma(A_1) = \mathsf{True}$ and $\Gamma(A_2) = \mathsf{True}$, the distributions of $\mathsf{Sign}(\mathscr{PP}, m, \mathscr{SK}_{A_1}, \Gamma)$ and $\mathsf{Sign}(\mathscr{PP}, m, \mathscr{SK}_{A_2}, \Gamma)$ 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 2.6 (Adaptive-Predicate Unforgeability)**.** An SP-ABS scheme is said to be existential unforgeable in adaptive-predicate model (or AP-UF-CMA) if for all PPT algorithms $\mathscr{A}$, the advantage

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

in $\mathsf{Exp}^{\mathrm{AP-UF-CMA}}_{\mathscr{A},\mathsf{ABS}}(\kappa)$ defined in Figure 1 is negligible function in κ, where $\mathscr{A}$ is provided access to keygen oracle $\mathscr{O}_K$ and sign oracle $\mathscr{O}_{Sg}$ (described below) and NRn is a natural restriction that (m^*, A, Γ^*) with $\Gamma^*(A) = \mathsf{True}$ was never queried to $\mathscr{O}_{Sg}$ oracle and for each set of attributes A queried to $\mathscr{O}_K$, it holds that $\Gamma^*(A) = \mathsf{False}$.

- KeyGen oracle ($\mathscr{O}_K$): Given a set of attributes A, oracle returns $\mathscr{SK}_A \longleftarrow \mathsf{KeyGen}(\mathscr{PP}, \mathscr{MSK}, A)$.
- Sign oracle ($\mathscr{O}_{Sg}$): Given (m, A, Γ), oracle returns $\delta \longleftarrow \mathsf{Sign}(\mathscr{PP}, m, \mathscr{SK}_A, \Gamma)$.

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

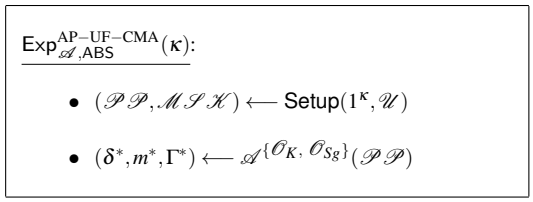


Figure 1: Experiments for unforgeability

Definition 2.7 (Selective-Predicate Unforgeability)**.** There is another variant of unforgeability, called selective-predicate existential unforgeability (SP-UF-CMA), where $\mathscr{A}$ submits a challenge policy Γ^* (later on which $\mathscr{A}$ will forge) before obtaining the $\mathscr{PP}$ of ABS.

Definition 2.8 (Strong Unforgeability)**.** The unforgeability defined in Definition 2.6 and 2.7 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^*,Γ^*) but then $\delta^* \neq \delta$ must hold, where δ is a signature obtained from sign oracle on the query message (m^*,Γ^*).

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

2.11 Signcryption-Policy Attribute-Based Signcryption

A signcryption-policy attribute-based signcryption (SCP-ABSC) scheme consists of four PPT algorithms - Setup, KeyGen, Signcrypt and Unsigncrypt.

- Setup: It takes a security parameter κ and a universe of attributes $\mathscr{U}$ as input, outputs public parameters $\mathscr{PP}$ and master secret key $\mathscr{MSK}$.
- KeyGen: It takes as input $\mathscr{PP}$, $\mathscr{MSK}$ and a set of attributes A and outputs a secret key $\mathscr{SK}_A$ corresponding to A.
- Signcrypt: It takes $\mathscr{PP}$, a message $m \in \mathscr{M}$, a signing key $\mathscr{SK}_A$, an associated policy Γ_s for sender with $\Gamma_s(A) = \mathsf{True}$ and an associated policy Γ_e for receiver as input and returns a signcryption U for (Γ_s,Γ_e) (we assume that U implicitly contains Γ_e).
- Unsigncrypt: It takes as input $\mathscr{PP}$, a signcryption U, a secret key $\mathscr{SK}_A$ and an associated policy Γ_s for sender. It returns a value from $\mathscr{M} \cup \{\perp\}$.

Correctness. For all $(\mathscr{PP},\mathscr{MSK}) \longleftarrow \mathsf{Setup}(1^\kappa,\mathscr{U})$, all $m \in \mathscr{M}$, all sets of attributes A, $\mathscr{SK}_A \longleftarrow \mathsf{KeyGen}(\mathscr{PP},\mathscr{MSK},A)$, all sender's associated policies Γ_s with $\Gamma_s(A) = \mathsf{True}$, all receiver's associated policies Γ_e, all signcryptions $\mathsf{U} \longleftarrow \mathsf{Signcrypt}(\mathscr{PP},m,\mathscr{SK}_A,\Gamma_s,\Gamma_e)$ and all sets of attributes $\tilde{A}$, $\mathscr{SK}_{\tilde{A}} \longleftarrow \mathsf{KeyGen}(\mathscr{PP},\mathscr{MSK},\tilde{A})$, it is required that $\mathsf{Unsigncrypt}(\mathscr{PP},\mathsf{U},\mathscr{SK}_{\tilde{A}},\Gamma_s) = m$ (resp. $\perp$) if $\Gamma_e(\tilde{A}) = \mathsf{True}$ (resp. $\Gamma_e(\tilde{A}) = \mathsf{False}$).

Remark 2.3. Similar to predicate signature, we assume that signer sends both signcryption and sender's policy Γ_s to receiver in predicate signcryption.

2.12 Security of SCP-ABSC

Definition 2.9 (Signer Privacy)**.** An SCP-ABSC scheme is said to be perfectly private if for all $(\mathscr{PP}, \mathscr{MSK}) \longleftarrow \mathsf{Setup}(1^\kappa, \mathscr{U})$, all sets of attributes $A_1, A_2 \subset \mathscr{U}$, all keys $\mathscr{SK}_{A_1} \longleftarrow \mathsf{KeyGen}(\mathscr{PP}, \mathscr{MSK}, A_1)$, $\mathscr{SK}_{A_2} \longleftarrow \mathsf{KeyGen}(\mathscr{PP}, \mathscr{MSK}, A_2)$, all messages $m \in \mathscr{M}$, all sender's associated policies Γ_s such that $\Gamma_s(A_1) = \mathsf{True}$ and $\Gamma_s(A_2) = \mathsf{True}$, and all receiver's associated policies Γ_e, the distributions of $\mathsf{Signcrypt}(\mathscr{PP}, m, \mathscr{SK}_{A_1}, \Gamma_s, \Gamma_e)$ and $\mathsf{Signcrypt}(\mathscr{PP}, m, \mathscr{SK}_{A_2}, \Gamma_s, \Gamma_e)$ 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. An SCP-ABSC scheme with signer-privacy is called perfectly private.

Definition 2.10 (Adaptive-Predicates IND-CCA Security)**.** An SCP-ABSC 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

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

in $\mathsf{Exp}^{\text{APs-IND-CCA}}_{\mathscr{A},\text{ABSC}}(\kappa)$ defined in Figure 2 is negligible function in security parameter κ, where $\mathscr{A}$ is provided access to key-gen oracle $\mathscr{O}_K$, signcrypt oracle $\mathscr{O}_S$ and unsigncrypt oracle $\mathscr{O}_U$ (described below), and NRn is a natural restriction that $(\mathsf{U}^*, A, \Gamma_s^*)$ with $\Gamma_e^*(A) = \mathsf{True}$ was never queried to $\mathscr{O}_U$ and for each set of attributes A queried to $\mathscr{O}_K$, it holds that $\Gamma_e^*(A) = \mathsf{False}$.

- KeyGen oracle ($\mathscr{O}_K$): Given a set of attributes A, oracle returns $\mathscr{SK}_A \longleftarrow \mathsf{KeyGen}(\mathscr{PP}, \mathscr{MSK}, A)$.
- $\mathsf{Signcrypt}$ oracle ($\mathscr{O}_S$): Given $(m, A, \Gamma_s, \Gamma_e)$, oracle returns $\mathsf{U} \longleftarrow \mathsf{Signcrypt}(\mathscr{PP}, m, \mathscr{SK}_A, \Gamma_s, \Gamma_e)$.
- $\mathsf{Unsigncrypt}$ oracle ($\mathscr{O}_U$): Given $(\mathsf{U}, A, \Gamma_s)$, oracle returns $\mathsf{Unsigncrypt}(\mathscr{PP}, \mathsf{U}, \mathscr{SK}_A, \Gamma_s)$.

$\underline{\mathsf{Exp}^{\text{APs-IND-CCA}}_{\mathscr{A},\text{ABSC}}(\kappa)}$:

- $(\mathscr{PP}, \mathscr{MSK}) \longleftarrow \mathsf{Setup}(1^\kappa, \mathscr{U})$
- $(m_0, m_1, A, \Gamma_s^*, \Gamma_e^*, st) \longleftarrow \mathscr{A}_1^{\{\mathscr{O}_K, \mathscr{O}_S, \mathscr{O}_U\}}(\mathscr{PP})$ with $|m_0| = |m_1|$
- $b \xleftarrow{\text{U}} \{0,1\}$
- $\mathsf{U}^* \longleftarrow \mathsf{Signcrypt}(\mathscr{PP}, m_b, \mathscr{SK}_A, \Gamma_s^*, \Gamma_e^*)$
- $b' \longleftarrow \mathscr{A}_2^{\{\mathscr{O}_K, \mathscr{O}_S, \mathscr{O}_U\}}(\mathscr{PP}, \mathsf{U}^*, st)$

Figure 2: 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 2.11 (Selective-Predicate IND-CCA Security)**.** Similarly to Definition 2.10, except in this model, $\mathscr{A}$ has to submit the challenge receiver's policy Γ_e^* before receiving $\mathscr{PP}$ of SCP-ABSC and the challenge sender's policy Γ_s^* in the challenge phase.

Remark 2.4. The selective-predicate IND-CCA (SP-IND-CCA) security model (Definition 2.11) is weaker than APs-IND-CCA security model (Definition 2.10).

Definition 2.12 (Selective-Predicates IND-CCA Security)**.** Similarly to Definition 2.10, except in this model, $\mathscr{A}$ has to submit the challenge receiver's policy Γ_e^* and challenge sender's policy Γ_s^* before receiving $\mathscr{PP}$ of SCP-ABSC.

Remark 2.5. Selective-predicates IND-CCA (SPs-IND-CCA) security model (Definition 2.12) is weaker than SP-IND-CCA security model (Definition 2.11).

Definition 2.13 (Adaptive-Predicates Unforgeability)**.** An SCP-ABSC scheme is said to be existential unforgeable in adaptive-predicates model (or APs-UF-CMA secure) if for all PPT $\mathscr{A}$, the advantage

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

in $\mathsf{Exp}^{\text{APs}-\text{UF}-\text{CMA}}_{\mathscr{A},\text{ABSC}}(\kappa)$ defined in Figure 3 is negligible function in κ, where $\mathscr{A}$ is provided access to key-gen oracle $\mathscr{O}_K$, signcrypt oracle $\mathscr{O}_S$ and unsigncrypt oracle $\mathscr{O}_U$ (described below), and NRn is a natural restriction that for each tuple (m,A,Γ_s,Γ_e) queried to $\mathscr{O}_S$ oracle, $(m,\Gamma_s,\Gamma_e) \neq (m^*,\Gamma_s^*,\Gamma_e^*)$ and for each set of attributes A queried to $\mathscr{O}_K$ oracle, it holds that $\Gamma_s^*(A) = \mathsf{False}$.

- KeyGen oracle ($\mathscr{O}_K$): Given a set of attributes A, oracle returns $\mathscr{SK}_A \longleftarrow \mathsf{KeyGen}(\mathscr{PP}, \mathscr{MSK}, A)$.
- $\mathsf{Signcrypt}$ oracle ($\mathscr{O}_S$): Given (m,A,Γ_s,Γ_e), oracle returns $\mathsf{U} \longleftarrow \mathsf{Signcrypt}(\mathscr{PP}, m, \mathscr{SK}_A, \Gamma_s, \Gamma_e)$.
- $\mathsf{Unsigncrypt}$ oracle ($\mathscr{O}_U$): Given (U,A,Γ_s), oracle returns $\mathsf{Unsigncrypt}(\mathscr{PP}, \mathsf{U}, \mathscr{SK}_A, \Gamma_s)$.

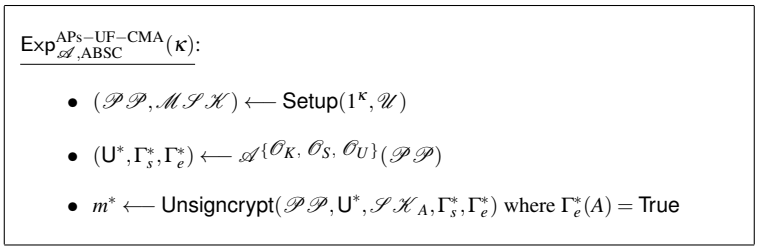


Figure 3: Experiment for unforgeability (adaptive-predicates UF-CMA security)

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

Definition 2.14 (Adaptive-Predicates Strong Unforgeability)**.** The above unforgeability (Definition 2.13) 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 $(m,\Gamma_s,\Gamma_e) \neq (m^*,\Gamma_s^*,\Gamma_e^*)$ is replaced by $(\mathsf{U},m,\Gamma_s,\Gamma_e) \neq (\mathsf{U}^*,m^*,\Gamma_s^*,\Gamma_e^*)$, where U is the reply for the query (m,A,Γ_s,Γ_e) to $\mathscr{O}_S$ oracle.

Definition 2.15 (Selective-Predicate Strong Unforgeability)**.** Similarly to Definition 2.14, except in this model, $\mathscr{A}$ has to submit the challenge sender's polycy Γ_s^* before receiving $\mathscr{PP}$ of SCP-ABSC and the challenge receiver's policy Γ_e^* at the time of forgery.

Remark 2.6. The selective-predicate sUF-CMA (SP-sUF-CMA) security model (Definition 2.15) is weaker than APs-sUF-CMA security model (Definition 2.14).

Definition 2.16 (Selective-Predicates Strong Unforgeability)**.** Similarly to Definition 2.14, except in this model, $\mathscr{A}$ has to submit the challenge sender's policy Γ_s^* and challenge receiver's policy Γ_e^* before receiving $\mathscr{PP}$ of PSC.

Remark 2.7. Selective-predicates sUF-CMA (SPs-sUF-CMA) security model (Definition 2.16) is weaker than SP-sUF-CMA security model (Definition 2.15).

3 Signature-Policy Attribute-Based Signature

We propose a basic signature-policy attribute-based signature (SP-ABS) scheme for monotone span programs. The construction is based on composite order bilinear pairing groups $(N := p_1p_2p_3, \mathbb{G}, \mathbb{G}_T, e)$ for 3 distinct primes p_1, p_2 and p_3. The subgroup $\mathbb{G}_{p_2}$ has no role in this scheme but it will be used to prove the security. The proposed SP-ABS scheme has the similar structure to that of [12] except for some modifications, viz., the encoding function from hash of messages to group elements and pairing groups. To guarantee unforgeability of the ABS scheme in adaptive-predicate model, we allow such modifications. In this basic SP-ABS construction, the policies, i.e., MSPs are restricted to have each entry of row labeling function ρ_s to be distinct. In other words, the row labeling functions ρ_s of the monotone span programs $\Gamma_s := (\mathbf{M}_s, \rho_s)$ are injective. From this basic ABS scheme, a complete ABS scheme can be constructed using the mechanism of Section 10, where ρ_s is no more assumed to be injective.

$\mathsf{Setup}(1^\kappa, \mathscr{U})$: It executes $\mathscr{J} := (N := p_1p_2p_3, \mathbb{G}, \mathbb{G}_T, e) \longleftarrow \mathscr{G}_{\mathsf{cbg}}(1^\kappa)$. It chooses $g \xleftarrow{\mathrm{U}} \mathbb{G}_{p_1}, Z_3 \xleftarrow{\mathrm{U}} \mathbb{G}_{p_3}, a, a_s, b_s, \alpha \xleftarrow{\mathrm{U}} \mathbb{Z}_N$ and $t_i \xleftarrow{\mathrm{U}} \mathbb{Z}_N$ for each attribute $i \in \mathscr{U}$. It then sets $u_s := g^{a_s}, v_s := g^{b_s}, T_i := g^{t_i}$ for $i \in \mathscr{U}$. Let $H_s : \{0,1\}^* \longrightarrow \mathbb{Z}_N$ be a hash function. The public parameters and master secret are given by

$\mathscr{PP} := (\mathscr{J}, g, g^a, u_s, v_s, g_T^\alpha, \{T_i\}_{i\in\mathscr{U}}, Z_3, H_s)$,
$\mathscr{MSK} := (\alpha)$.

$\mathsf{KeyGen}(\mathscr{PP}, \mathscr{MSK}, A)$: It picks $t \xleftarrow{\mathrm{U}} \mathbb{Z}_N, R, R_0' \xleftarrow{\mathrm{U}} \mathbb{G}_{p_3}$. For each attribute $i \in A$, the algorithm chooses $R_i \xleftarrow{\mathrm{U}} \mathbb{G}_{p_3}$ and outputs the secret key

$\mathscr{SK}_A := [A,\ K := g^{\alpha+at}R,\ L := g^tR_0',\ K_i := T_i^tR_i\ \forall i \in A]$.

$\mathsf{Sign}(\mathscr{PP}, m, \mathscr{SK}_A, \Gamma_s := (\mathbf{M}_s, \rho_s))$: Let $\mathbf{M}_s$ be an $\ell_s \times n_s$ matrix. If $\Gamma_s(A) = \mathsf{False}$, returns $\perp$. Otherwise computes the sets $\mathscr{I}_A \subset [\ell_s]$ and $\{\alpha_s^{(i)}\}_{i\in\mathscr{I}_A}$ such that $\sum_{i\in\mathscr{I}_A}\alpha_s^{(i)}\mathbf{M}_s^{(i)} = \mathbf{1}$. It selects $\beta \xleftarrow{\mathrm{U}} \{\beta = (\beta_1, \ldots, \beta_{\ell_s}) \in \mathbb{Z}_N^{\ell_s} \mid \sum_{i\in[\ell_s]}\beta_i\mathbf{M}_s^{(i)} = \mathbf{0}\}$. Suppose $\mathscr{SK}_A := [A,\ K := g^{\alpha+at}R,\ L := g^tR_0',\ K_i := T_i^tR_i\ \forall i \in A]$, then it re-randomizes the $\mathbb{G}_{p_1}$ part of the key $\mathscr{SK}_A$ as follows.

$\tilde{\mathscr{SK}}_A := [A,\ \tilde{K} := Kg^{a\hat{t}},\ \tilde{L} := Lg^{\hat{t}},\ \tilde{K}_i := K_iT_i^{\hat{t}}\ \forall i \in A]$, where $\hat{t} \xleftarrow{\mathrm{U}} \mathbb{Z}_N$
$:= [A,\ \tilde{K} := g^{\alpha+a\tilde{t}}R,\ \tilde{L} := g^{\tilde{t}}R_0',\ \tilde{K}_i := T_i^{\tilde{t}}R_i\ \forall i \in A]$, where $\tilde{t} := t + \hat{t}$.

It picks $r_s, \tau \xleftarrow{\mathrm{U}} \mathbb{Z}_N, \bar{R}, \bar{R}_0 \xleftarrow{\mathrm{U}} \mathbb{G}_{p_3}$ and for each $i \in [\ell_s]$, it chooses $\bar{R}_i \xleftarrow{\mathrm{U}} \mathbb{G}_{p_3}$. Then it computes $\hbar_s := H_s(m, \Gamma_s)$. The components of signature are given by

$\mathbf{S}_0 := \big(\ \tilde{K}(u_s^{\hbar_s}v_s)^{r_s}\bar{R},\ g^{r_s}\bar{R}_0\ \big)$,
$\mathbf{S}_i := \big(\ \tilde{L}^{\alpha_s^{(i)}}(g^\tau)^{\beta_i}\bar{R}_i,\ (\tilde{K}_{\rho_s(i)})^{\alpha_s^{(i)}}(T_{\rho_s(i)})^{\tau\beta_i}\bar{R'}_i\ \big)$ for $i \in [\ell_s]$.

In the above expression, it sets $\alpha_s^{(i)} := 0$ for $i \notin \mathscr{I}_A$.
After simplification, it gives

$\mathbf{S}_0 := \big(\ g^{\alpha+a\tilde{t}}(u_s^{\hbar_s}v_s)^{r_s}\tilde{R},\ g^{r_s}\tilde{R}_0\ \big)$, where $\tilde{R} := R\bar{R}, \tilde{R}_0 := \bar{R}_0$,
$\mathbf{S}_i := \big(\ (g^{\tilde{t}})^{\alpha_s^{(i)}}(g^\tau)^{\beta_i}\tilde{R}_i,\ (T_{\rho_s(i)}^{\tilde{t}})^{\alpha_s^{(i)}}(T_{\rho_s(i)}^{\tau})^{\beta_i}\tilde{R'}_i\ \big)$ for $i \in [\ell_s]$.

In the above expression, it sets $\tilde{R}_i := (R'_0)^{\alpha_s^{(i)}} \bar{R}_i$, $\tilde{R'}_i := R_{\rho_s(i)}^{\alpha_s^{(i)}+\tau\beta_i} \bar{R}'_i$.
It returns the signature $\delta := (\mathbf{S}_0, \{\mathbf{S}_i\}_{i\in[\ell_s]})$.

$\mathsf{Ver}(\mathscr{PP}, m, \delta, \Gamma_s)$: It first computes a verification text, then using this verification text it will verify the signature. The following is the construction of verification text. It picks $\mathbf{u}_s := (s, u_2, \ldots, u_{n_s}) \xleftarrow{\mathrm{U}} \mathbb{Z}_N^{n_s}$ and $r_s^{(i)} \xleftarrow{\mathrm{U}} \mathbb{Z}_N$ for $i \in [\ell_s]$. It computes $\hbar_s := H_s(m, \Gamma_s)$. Let $\mathbf{M}_s^{(i)}$ denote the i^{th} row of the matrix, $\mathbf{M}_s$ and let $\lambda_s^{(i)} :=< \mathbf{M}_s^{(i)}, \mathbf{u}_s >$. The components of verification text are given by

$\mathbf{V}_0 := \left(g^s, (u_s^{\hbar_s} v_s)^s, g_T^{\alpha s} \right)$,

$\mathbf{V}_i := \left(g^{a\lambda_s^{(i)}} T_{\rho_s(i)}^{-r_s^{(i)}}, \quad g^{r_s^{(i)}} \right)$ for $i \in [\ell_s]$.

The final verification text is $\mathscr{V} := (\mathbf{V}_0, \{\mathbf{V}_i\}_{i\in[\ell_s]})$.

Now, it computes $\Delta_s := \frac{e(S_{01}, V_{01})}{e(S_{02}, V_{02}) \cdot \prod_{i=1}^{\ell_s} (e(S_{i1}, V_{i1}) \cdot e(S_{i2}, V_{i2}))}$ and checks $\Delta_s \stackrel{?}{=} V_{03}$. It returns 1 if $\Delta_s = V_{03}$, else returns 0.

Correctness.

$$\begin{aligned}
\Delta_s &= \frac{e(S_{01}, V_{01})}{e(S_{02}, V_{02}) \cdot \prod_{i=1}^{\ell_s} (e(S_{i1}, V_{i1}) \cdot e(S_{i2}, V_{i2}))} \\
&= \frac{g_T^{\alpha s + a\tilde{t} s} \cdot e(u_s^{\hbar_s} v_s, g)^{s r_s}}{e(u_s^{\hbar_s} v_s, g)^{s r_s} \cdot \prod_{i=1}^{\ell_s} \left(e\left(g^{\tilde{t}\alpha_s^{(i)} + \tau\beta_i}, g^{a\lambda_s^{(i)} - r_s^{(i)} t_{\rho_s(i)}}\right) \cdot e\left(g^{\tilde{t}\alpha_i t_{\rho_s(i)} + \tau\beta_i t_{\rho_s(i)}}, g^{r_s^{(i)}}\right) \right)} \\
&= \frac{g_T^{\alpha s + a\tilde{t} s}}{\prod_{i=1}^{\ell_s} g_T^{a\tilde{t}\lambda_s^{(i)}\alpha_s^{(i)} + a\tau\lambda_s^{(i)}\beta_i}} = \frac{g_T^{\alpha s + a\tilde{t} s}}{g_T^{a\tilde{t}\sum_{i=1}^{\ell_s}\lambda_s^{(i)}\alpha_s^{(i)} + a\tau\sum_{i=1}^{\ell_s}\lambda_s^{(i)}\beta_i}} = g_T^{\alpha s}
\end{aligned}$$

4 Security Proof of SP-ABS

Theorem 4.1. *The proposed attribute-based signature scheme in Section 3 is perfectly private (Definition 2.5).*

Proof. Let A_1 and A_2 be two sets of attributes and $\Gamma_s := (\mathbf{M}_s, \rho_s)$ be an access policy such that $\Gamma_s(A_1) = \Gamma_s(A_2) = \mathsf{True}$. Then, there exist sets $\mathscr{I}_{A_1} \subset [\ell_s]$, $\mathscr{I}_{A_2} \subset [\ell_s]$ and $\{\alpha_{s1}^{(i)}\}_{i\in[\ell_s]}$, $\{\alpha_{s2}^{(i)}\}_{i\in[\ell_s]}$ such that $\sum_{i\in\mathscr{I}_{A_1}} \alpha_{s1}^{(i)} \mathbf{M}_s^{(i)} = \mathbf{1}$ and $\sum_{i\in\mathscr{I}_{A_2}} \alpha_{s2}^{(i)} \mathbf{M}_s^{(i)} = \mathbf{1}$. In other words, there exist vectors α_{s1} and α_{s2} such that $\sum_{i=1}^{\ell_s} \alpha_{s1}^{(i)} \mathbf{M}_s^{(i)} = \sum_{i=1}^{\ell_s} \alpha_{s2}^{(i)} \mathbf{M}_s^{(i)} = \mathbf{1}$, where $\alpha_{s1}^{(i)} = 0$ if $i \notin \mathscr{I}_{A1}$ and $\alpha_{s2}^{(i)} = 0$ if $i \notin \mathscr{I}_{A2}$. We will show that the signatures δ_1 and δ_2 generated respectively by the keys $\mathscr{SK}_{A_1}$ and $\mathscr{SK}_{A_2}$ on behalf of access policy Γ_s are identical. First of all note that the $\mathbb{G}_{p_3}$ components in the signature do not carry any information about the attributes used to sign a message. An unbounded adversary can compute the values $r_s, \tilde{t}$ using the public parameters and $\mathbf{S}_0$, but since those values are chosen uniformly and independently random from $\mathbb{Z}_N$, those values also do not carry any information regarding the attribute set used to sign the message. So, $\mathbf{S}_0$ of any two signatures are identical. The only parts of the signature carrying the information of attributes are $\mathbf{S}_i$ for $i \in [\ell_s]$. Now consider two system of equations, homogeneous and non-homogeneous given below:

$$\sum_{i=1}^{\ell_s} \beta_i \mathbf{M}_s^{(i)} = \mathbf{0} \tag{2}$$

$$\sum_{i=1}^{\ell_s} \alpha_s^{(i)} \mathbf{M}_s^{(i)} = \mathbf{1} \tag{3}$$

Let $\mathscr{Y}_0 := \{\beta \mid \sum_{i=1}^{\ell_s} \beta_i \mathbf{M}_s^{(i)} = \mathbf{0}\}$ and $\mathscr{Y}_1 := \{\alpha_s \mid \sum_{i=1}^{\ell_s} \alpha_s^{(i)} \mathbf{M}_s^{(i)} = \mathbf{1}\}$ respectively denote the solution set of equation 2 and 3. Let α_{s1} and α_{s2} be any two solutions of system 3, then we can write $\mathscr{Y}_1 = \alpha_{s1} + \mathscr{Y}_0 = \alpha_{s2} + \mathscr{Y}_0$. So, the distributions of $\alpha_{s1} + \beta$ and $\alpha_{s2} + \beta$ are identical for $\beta \xleftarrow{\text{U}} \mathscr{Y}_0$. Therefore, the distribution of $(\tilde{t}_1 \alpha_{s1}^{(i)} + \tau_1 \beta_i)_{i \in [\ell_s]}$ and $(\tilde{t}_2 \alpha_{s2}^{(i)} + \tau_2 \beta_i)_{i \in [\ell_s]}$ are identical. Since the distribution of $\mathbf{S}_i$ is $(g^{\tilde{t}\alpha_s^{(i)} + \tau\beta_i},\ g^{(\tilde{t}\alpha_s^{(i)} + \tau\beta_i) t_{\rho_s(i)}})$, the distributions of the signatures δ_1 and δ_2 are identical. □

4.1 Adaptive-Predicate Existential Unforgeability of SP-ABS

We prove the adaptive-predicate unforgeability of our basic ABS scheme by the proof technique of Okamoto and Takashima [36] and the dual system methodology of Brent Waters [47]. This methodology requires to define the semi-functional verification texts, signatures and keys. Here, we define two types of semi-functional verification texts, viz., type 1 and type 2. Two forms of semi-functional keys are considered here – type 1 and type 2. Our semi-functional signatures consist of two forms, viz., type 1 and type 2. In the sequence of games, the verification text is first changed from normal to semi-functional type 1. Then, each queried key is changed from normal to semi-functional type 1, then semi-functional type 1 to type 2. Consecutively, each queried signature is changed from normal to semi-functional type 2 through semi-functional type 1 signature. In the final game, the semi-functional type 1 verification text is changed to semi-functional type 2 verification text.

In the following material, the part framed by a box indicates that either it will be changed in next description or it has been changed from previous description. We use the abbreviation 'sf' and 'vText' for 'semi-functional' and 'verification text' respectively.

Semi-functional type 1 verification text. Pick $c, \iota \xleftarrow{\text{U}} \mathbb{Z}_N$, $\mathbf{v}_s \xleftarrow{\text{U}} \mathbb{Z}_N^{n_s}$. For each $i \in [\ell_s]$, pick $\gamma_s^{(i)} \xleftarrow{\text{U}} \mathbb{Z}_N$. For each $i \in \mathscr{U}$, choose $z_i \xleftarrow{\text{U}} \mathbb{Z}_N$. The sf-type 1 vText is obtained by modifying normal vText $\mathscr{V} = (\mathbf{V}_0, \{\mathbf{V}_i\}_{i \in [\ell_s]})$ as given below:

$\mathbf{V}_0 := \big(\ g^s \boxed{g_2^c}, (u_s^{\hbar_s} v_s)^s \boxed{g_2^{\iota}},\ g_T^{\alpha s}\ \big)$,

$\mathbf{V}_i := \big(\ g^{a\lambda_s^{(i)}} T_{\rho_s(i)}^{-r_s^{(i)}} \boxed{g_2^{<\mathbf{M}_s^{(i)}, \mathbf{v}_s> + \gamma_s^{(i)} z_{\rho_s(i)}}},\quad g^{r_s^{(i)}} \boxed{g_2^{-\gamma_s^{(i)}}}\ \big)$, for $i \in [\ell_s]$.

Semi-functional type 2 verification text. This is same as sf-type 1 vText except the following:

$\mathbf{V}_0 := \big(\ g^s g_2^c, (u_s^{\hbar_s} v_s)^s g_2^{\iota}, \boxed{\hat{g}_t}\ \big)$, where $\hat{g}_t \xleftarrow{\text{U}} \mathbb{G}_T$.

Semi-functional type 1 key. Choose $b, d \xleftarrow{\text{U}} \mathbb{Z}_N$. First create a normal key

$\mathscr{SK}_A := [A,\ K := g^{\alpha + at} R,\ L := g^t R_0',\ K_i := T_i^t R_i\ \forall i \in A]$

and then modify it to sf-type 1 key as shown below:

$\mathscr{SK}_A := [A,\ K := g^{\alpha + at} R \boxed{g_2^d},\ L := g^t R_0' \boxed{g_2^b},\ K_i := T_i^t R_i \boxed{g_2^{bz_i}}\ \forall i \in A]$.

Semi-functional type 2 key. This is same as sf-type 1 key except $b = 0$, i.e.,

$\mathscr{SK}_A := [A,\ K := g^{\alpha + at} R g_2^d,\ L := g^t R_0',\ K_i := T_i^t R_i\ \forall i \in A]$.

Semi-functional type 1 signature. Choose $\tilde{b}, \tilde{d} \xleftarrow{\text{U}} \mathbb{Z}_N$. First, a normal signature is created, then this is changed to sf-type 1 signature by adding $\mathbb{G}_{p_2}$ part as given below:

$$\mathbf{S}_0 := \Big(g^{\alpha+a\tilde{t}}(u_s^{\hbar_s}v_s)^{r_s}\tilde{R}\boxed{g_2^{\tilde{d}}},\ g^{r_s}\tilde{R}_0\boxed{g_2^{\tilde{b}}} \Big),$$
$$\mathbf{S}_i := \Big((g^{\tilde{t}})^{\alpha_s^{(i)}}(g^{\tau})^{\beta_i}\tilde{R}_i,\qquad (T^{\tilde{t}}_{\rho_s(i)})^{\alpha_s^{(i)}}(T^{\tau}_{\rho_s(i)})^{\beta_i}\tilde{R'}_i \Big).$$

Semi-functional type 2 signature. This is same as sf-type 1 signature except $\tilde{b} = 0$, i.e.,
$\mathbf{S}_0 := \Big(g^{\alpha+a\tilde{t}}(u_s^{\hbar_s}v_s)^{r_s}\tilde{R}g_2^{\tilde{d}},\ g^{r_s}\tilde{R}_0 \Big)$,
$\mathbf{S}_i := \Big((g^{\tilde{t}})^{\alpha_s^{(i)}}(g^{\tau})^{\beta_i}\tilde{R}_i,\quad (T^{\tilde{t}}_{\rho_s(i)})^{\alpha_s^{(i)}}(T^{\tau}_{\rho_s(i)})^{\beta_i}\tilde{R'}_i \Big)$.
A normal signature can be verified by a normal vText as well as an sf-type 1 vText. But, if a valid sf-type 1 (resp. sf-type 2) signature is verified by an sf-type 1 vText, we will have an additional term $e(g_2,g_2)^{\tilde{d}c-\tilde{b}\iota}$ (resp. $e(g_2,g_2)^{\tilde{d}c}$) in Δ_s, i.e., the verification process fails. In sf-type 2 vText, the V_{03} part are made completely random by setting $V_{03} := \hat{g}_t$, where $\hat{g}_t \xleftarrow{\text{U}} \mathbb{G}_T$. Therefore, any form of signatures even including normal will be invalid with respect to sf-type 2 vText.

Theorem 4.2. *The proposed basic SP-ABS scheme in Section 3 is existential unforgeable in adaptive-predicate model (Definition 2.6) if DSG1, DSG2 and DSG3 assumptions hold in $\mathscr{J}$ and H_s is a collision resistant hash function.*

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 $2(\nu_1+\nu_2)+3$ games which are defined below:

– Game_{Real} := Original AP-UF-CMA game of ABS scheme.

– Game_0 (:= Game_{1-0-2}) is just like Game_{Real} except that the vText is of sf-type 1.

– Game_{1-k-1} (for $1 \le k \le \nu_1$) ia same as $\text{Game}_{1-(k-1)-2}$ except the k^{th} key is of sf-type 1.

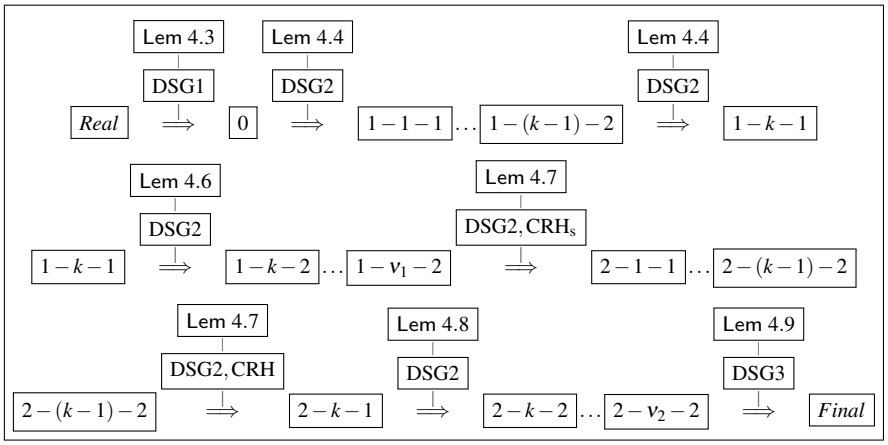
– Game_{1-k-2} (for $1 \le k \le \nu_1$) is same as Game_{1-k-1} except the k^{th} key is sf-type 2.

– Game_{2-k-1} (for $1 \le k \le \nu_2$) is same as $\text{Game}_{2-(k-1)-2}$ except the k^{th} signature is sf-type 1. So, in this sequel, we define $\text{Game}_{2-0-2} := \text{Game}_{1-\nu_1-2}$.

– 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 $\text{Game}_{2-\nu_2-2}$ except that the vText is of sf-type 2.

In Game_{Final}, the part V_{03} in $\mathbf{V}_0$ is chosen independently and uniformly 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 Section 4.2), we have the following reduction:

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

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

4.2 Lemmas Used in the Proof of Theorem 4.2

Lemma 4.3. Game_{Real} *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* $|\mathsf{Adv}^{\mathrm{Real}}_{\mathscr{A},\mathrm{ABS}}(\kappa) - \mathsf{Adv}^{0}_{\mathscr{A},\mathrm{ABS}}(\kappa)| \leq \mathsf{Adv}^{\mathrm{DSG1}}_{\mathscr{B}}(\kappa)$.

Proof. We establish a PPT simulator $\mathscr{B}$ who receives an instance of DSG1, $(\mathscr{J}, g, Z_3, T_\beta)$ with $\beta \xleftarrow{\mathrm{U}} \{0,1\}$ and depending on the distribution of β, it simulates either Game_{Real} or Game_0.

Setup: $\mathscr{B}$ chooses $\alpha, a, a_s, b_s \xleftarrow{\mathrm{U}} \mathbb{Z}_N$ and $t_i \xleftarrow{\mathrm{U}} \mathbb{Z}_N$ for $i \in \mathscr{U}$. Then, it sets $u_s := g^{a_s}, v_s := g^{b_s}$ and $T_i := g^{t_i}$ for $i \in \mathscr{U}$. $\mathscr{B}$ selects a hash function $H_s : \{0,1\}^* \longrightarrow \mathbb{Z}_N$. It provides $\mathscr{PP} := (\mathscr{J}, g, g^a, u_s, v_s, g_T^\alpha, \{T_i\}_{i\in\mathscr{U}}, Z_3, H_s)$ to $\mathscr{A}$ and keeps $\mathscr{MSK} := (\alpha)$ to itself.

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

- KeyGen: It is normal key. $\mathscr{B}$ can handle the key queries of $\mathscr{A}$, since the $\mathscr{MSK}$ is known to him.
- Sign: Let (m, A, Γ_s) be a signature query made by $\mathscr{A}$. If $\Gamma_s(A) = \mathsf{False}$, it returns $\perp$. It is normal signature. $\mathscr{B}$ can answer the queries of $\mathscr{A}$, since he can construct $\mathscr{SK}_A$ using the $\mathscr{MSK}$ known to him.

Forgery: $\mathscr{A}$ outputs a signature δ^* for $(m^*, \Gamma_s^* := (\mathbf{M}_s^*, \rho_s^*))$, where $\mathbf{M}_s^*$ is a matrix of order $\ell_s^* \times n_s^*$. Then, $\mathscr{B}$ prepares a vText for (m^*, Γ_s^*) as follows. It computes $\hbar_s^* := H_s(m^*, \Gamma_s^*)$. It selects $\mathbf{v}_\mathbf{s}' := (1, v_2', \ldots, v_{n_s^*}')$, where $v_2', \ldots, v_{n_s^*}' \xleftarrow{\mathrm{U}} \mathbb{Z}_N$. For $i \in [\ell_s^*]$, it sets $\lambda_s^{*(i)} := < \mathbf{M}_s^{*(i)}, \mathbf{v}_s' >$. It chooses $r_i' \xleftarrow{\mathrm{U}} \mathbb{Z}_N$ for $i \in [\ell_s^*]$. $\mathscr{B}$ implicitly sets $g^s := T_\beta\big|_{\mathbb{G}_{p_1}}$. Then it computes the following components:

$$\mathbf{V}_0 := \left(T_\beta, T_\beta^{\hbar_s^* a_s + b_s}, e(g^\alpha, T_\beta) \right),$$
$$\mathbf{V}_i := \left(T_\beta^{a\lambda_s^{*(i)}} T_\beta^{-r_i' t_{\rho_s^*(i)}}, \quad T_\beta^{r_i'} \right) \text{ for } i \in [\ell_s^*].$$

The final vText is $\mathscr{V} := (\mathbf{V}_0, \{\mathbf{V}_i\}_{i\in[\ell_s^*]})$. $\mathscr{B}$ verifies the signature δ^* using the vText $\mathscr{V}$ and returns 1 if it passes verification else returns 0.

Analysis: $\mathscr{B}$ implicitly sets $\mathbf{u}_s := s\mathbf{v}_s' = (s, sv_2', \ldots, sv_{n_s^*}')$ and $r_s^{(i)} := sr_i'$ for $i \in [\ell_s^*]$. Since, $v_2', \ldots, v_{n_s^*}'$ are chosen uniformly and independently from $\mathbb{Z}_N$, so the vector $\mathbf{u}_s$ is a random vector over $\mathbb{Z}_{p_1}$. Similarly, $r_s^{(1)}, \ldots, r_s^{(\ell_s^*)}$ are uniformly and independently distributed over $\mathbb{Z}_{p_1}$ as $r_1', \ldots, r_{\ell_s^*}'$ are so over $\mathbb{Z}_N$. Therefore, $\mathscr{V}$ is a properly distributed normal verification text if $\beta = 0$ (i.e., $T_\beta = g^s$).

Suppose $\beta = 1$, i.e., $T_\beta = g^s g_2^c$ for some $c \in \mathbb{Z}_N$. It implicitly sets $\mathbf{v}_s := ca\mathbf{v}_s' = (ca, cav_2', \ldots, cav_{n_s^*}')$, $\iota := c(\hbar_s^* a_s + b_s)$, $\gamma_s^{(i)} := -cr_i'$ and $z_{\rho_s^*(i)} := t_{\rho_s^*(i)}$ for $i \in [\ell_s^*]$. By Chinese Remainder Theorem (CRT), the values $v_2', \ldots, v_{n_s^*}'$ and $r_i', t_{\rho_s^*(i)}$ for $i \in [\ell_s^*]$ over $\mathbb{Z}_{p_1}$ are uncorrelated from those values over $\mathbb{Z}_{p_2}$. Therefore, $\mathscr{V}$ is a properly distributed sf-type 1 verification text if $\beta = 1$. □

Lemma 4.4. $\text{Game}_{1-(k-1)-2}$ *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* $|\mathsf{Adv}^{1-(k-1)-2}_{\mathscr{A},\text{ABS}}(\kappa) - \mathsf{Adv}^{1-k-1}_{\mathscr{A},\text{ABS}}(\kappa)| \leq \mathsf{Adv}^{\text{DSG2}}_{\mathscr{B}}(\kappa)$ *for* $1 \leq k \leq \nu_1$.

Proof. $\mathscr{B}$ is given an instance of DSG2, $(\mathscr{J}, g, Z_1Z_2, W_2W_3, Z_3, T_\beta)$ with $\beta \xleftarrow{\text{U}} \{0,1\}$ and depending on the distribution of β, $\mathscr{B}$ simulates either $\text{Game}_{1-(k-1)-2}$ or Game_{1-k-1}.

Setup: Similar to Lemma 4.3.

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

– KeyGen: The first $(k-1)$ keys are of sf-type 2 and the last $(\nu_1 - k)$ are normal keys. The k^{th} key is normal in $\text{Game}_{1-(k-1)-2}$ and sf-type 1 in Game_{1-k-1}. Let A_j be the j^{th} query set of attributes. $\mathscr{B}$ answers the key $\mathscr{SK}_{A_j}$ 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 sf-type 2 key. It picks $t \xleftarrow{\text{U}} \mathbb{Z}_N$, $R'_0, R_i \xleftarrow{\text{U}} \mathbb{G}_{p_3}$ for $i \in A_j$ and returns the following key to $\mathscr{A}$:
  $\mathscr{SK}_{A_j} := [A_j,\ K := g^{\alpha+at}(W_2W_3)^t,\ L := g^t R'_0,\ K_i := T_i^t R_i,\ \forall i \in A_j]$.
  Since, $t \mod p_2$ and $t \mod p_3$ are uncorrelated, so the key $\mathscr{SK}_{A_j}$ is properly distributed sf-type 2 key.
- If $j = k$ then it is either normal or sf-type 1 key. $\mathscr{B}$ generates $\mathscr{SK}_{A_k}$ using T_β of the instance of DSG2. $\mathscr{B}$ implicitly sets $g^t := T_\beta\big|_{\mathbb{G}_{p_1}}$. It chooses $R, R'_0, R_i \xleftarrow{\text{U}} \mathbb{G}_{p_3}$ for $i \in A_k$ and returns the following key to $\mathscr{A}$:
  $\mathscr{SK}_{A_k} := [A_k,\ K := g^\alpha T_\beta^a R,\ L := T_\beta R'_0,\ K_i := T_\beta^{t_i} R_i,\ \forall i \in A_k]$.

– Sign: Let (m, A, Γ_s) be a signature query made by $\mathscr{A}$. If $\Gamma_s(A) = \mathsf{False}$, it returns $\perp$. It is normal signature. $\mathscr{B}$ can answer the queries of $\mathscr{A}$, since he can construct $\mathscr{SK}_A$ using the $\mathscr{MSK}$ known to him.

Forgery: $\mathscr{A}$ outputs a signature δ^* for $(m^*, \Gamma^*_s := (\mathbf{M}^*_s, \rho^*_s))$, where $\mathbf{M}^*_s$ is a matrix of order $\ell^*_s \times n^*_s$. Then, $\mathscr{B}$ prepares a vText for (m^*, Γ^*_s) as follows. It computes $\hbar^*_s := H_s(m^*, \Gamma^*_s)$. It selects $\mathbf{v}'_s := (1, v'_2, \ldots, v'_{n^*_s})$, where $v'_2, \ldots, v'_{n^*_s} \xleftarrow{\text{U}} \mathbb{Z}_N$. For $i \in [\ell^*_s]$, it sets $\lambda^{*(i)}_s :=< \mathbf{M}^{*(i)}_s, \mathbf{v}'_s >$. It chooses $r'_i \xleftarrow{\text{U}} \mathbb{Z}_N$ for $i \in [\ell^*_s]$ and computes the following components:

$\mathbf{V}_0 := \big(\ Z_1Z_2,\ (Z_1Z_2)^{\hbar^*_s a_s + b_s},\ e(g^\alpha, Z_1Z_2)\ \big)$,

$\mathbf{V}_i := \big(\ (Z_1Z_2)^{a\lambda^{*(i)}_s}(Z_1Z_2)^{-r'_i t_{\rho^*_s(i)}},\quad (Z_1Z_2)^{r'_i}\ \big)$, for $i \in [\ell^*_s]$.

The final vText is $\mathscr{V} := (\mathbf{V}_0, \{\mathbf{V}_i\}_{i \in [\ell^*_s]})$. $\mathscr{B}$ verifies the signature δ^* using the vText $\mathscr{V}$ and returns 1 if it is valid else 0.

Analysis: Let $Z_1Z_2 = g^s g_2^c$. $\mathscr{B}$ implicitly sets $\mathbf{u}_s := s\mathbf{v}'_s = (s, sv'_2, \ldots, sv'_{n^*_s})$ and $r^{(i)}_s := sr'_i$ for $i \in [\ell^*_s]$. Since, $v'_2, \ldots, v'_{n^*_s}$ are chosen uniformly and independently from $\mathbb{Z}_N$, so the vector $\mathbf{u}_s$ is a random vector over $\mathbb{Z}_{p_1}$. Similarly, $r'_1, \ldots, r'_{\ell^*_s}$ are uniformly and independently distributed over $\mathbb{Z}_{p_1}$ as $r^{(1)}_s, \ldots, r^{(\ell^*_s)}_s$ are so over $\mathbb{Z}_N$. It implicitly sets $\mathbf{v}_s := ca\mathbf{v}'_s = (ca, cav'_2, \ldots, cav'_{n^*_s})$, $\iota := c(\hbar^*_s a_s + b_s)$, $\gamma^{(i)}_s := -cr'_i$ and $z_{\rho^*_s(i)} := t_{\rho^*_s(i)}$ for $i \in [\ell^*_s]$. By CRT, the values $v'_2, \ldots, v'_{n^*_s}$ and $r'_i, t_{\rho^*_s(i)}$ for $i \in [\ell^*_s]$ over $\mathbb{Z}_{p_1}$ are uncorrelated from those values over $\mathbb{Z}_{p_2}$. Hence, $\mathscr{V}$ is a properly distributed sf-type 1 verification text. Therefore, the joint distribution of keys, signatures and vText is identical to that of $\text{Game}_{1-(k-1)-2}$ if $\beta = 0$ (i.e., $T_\beta = g^t g_3^\varsigma$). Now, suppose $\beta = 1$, i.e., $T_\beta = g^t g_2^b g_3^\varsigma$. $\mathscr{B}$ implicitly sets $d := ba$, $z_i := t_i$ for $i \in A_k$. Since, $a \mod p_1$ and $t_i \mod p_1$ are uncorrelated respectively from $a \mod p_2$ and $t_i \mod p_2$, $\mathscr{SK}_{A_k}$ is almost

properly distributed sf-type 1 key except, the correlation between b and $d = ba$ (the exponents of g_2 in L and K resp.) also appears between c (the exponent of g_2 in V_{01}) and ac (first component of $\mathbf{v}_s$). If we show either $d = ba$ or ac is independent from the adversary's point of view, then we are done.

Claim 4.5. *The shared value ac is information-theoretically hidden from the adversary $\mathscr{A}$.*

Proof of the claim requires the injective restriction on row labeling function ρ_s^* and the restriction on key queries $\mathscr{SK}_A$ such that $\Gamma_s^*(A) = \mathsf{False}$. Proof of the claim is found in proof of the Lemma 8 in [25].

Therefore, the joint distribution of keys, signatures and vText is identical to that of Game_{1-k-1} if $\beta = 1$ (i.e., $T_\beta = g^t g_2^b g_3^\varsigma$). □

Lemma 4.6. Game_{1-k-1} *and* Game_{1-k-2} *are indistinguishable under the DSG2 assumption. That is, for every adversary $\mathscr{A}$, there exists a PPT algorithm $\mathscr{B}$ such that* $|\mathsf{Adv}^{1-k-1}_{\mathscr{A},\text{ABS}}(\kappa) - \mathsf{Adv}^{1-k-2}_{\mathscr{A},\text{ABS}}(\kappa)| \leq \mathsf{Adv}^{\text{DSG2}}_{\mathscr{B}}(\kappa)$ *for* $1 \leq k \leq \nu_1$.

Proof. It is similar to that of Lemma 4.4 except, the k^{th} key query answering. An instance of DSG2, $(\mathscr{J}, g, Z_1Z_2, W_2W_3, Z_3, T_\beta)$ with $\beta \xleftarrow{\text{U}} \{0,1\}$ is given to the simulator $\mathscr{B}$ and depending on the distribution of β, it simulates either Game_{1-k-1} or Game_{1-k-2}. Below, we only describe the construction of k^{th} key.

- It is either sf-type 1 or sf-type 2 key. $\mathscr{B}$ generates $\mathscr{SK}_{A_k}$ using T_β of the instance of DSG2. $\mathscr{B}$ implicitly sets $g^t := T_\beta\big|_{\mathbb{G}_{p_1}}$. It chooses $\zeta \xleftarrow{\text{U}} \mathbb{Z}_N$, $R'_0, R_i \xleftarrow{\text{U}} \mathbb{G}_{p_3}$ for $i \in A_k$ and returns the following key to $\mathscr{A}$:

  $$\mathscr{SK}_{A_k} := [A_k,\ K := g^\alpha T_\beta^a (W_2W_3)^\zeta,\ L := T_\beta R'_0,\ K_i := T_\beta^{t_i} R_i,\ \forall i \in A_k].$$

In k^{th} key construction an additional term, $(W_2W_3)^\zeta$ is added to K which says that perfectness of the simulation does not require Claim 4.5.

It is easy to check that if $\beta = 0$ (i.e., $T_\beta = g^t g_3^\varsigma$), then the joint distribution of keys, signatures and vText is identical to that of Game_{1-k-1}. Now, suppose $\beta = 1$, i.e., $T_\beta = g^t g_2^b g_3^\varsigma$. $\mathscr{B}$ implicitly sets $d := ba + y\zeta$, where $W_2 = g_2^w$ $z_i := t_i$ for $i \in A_k$. Due to the distribution of $d = ba + w\zeta$, the above correlation (found in Lemma 4.4) between key and ciphertext is not possible.

Therefore, the joint distribution of keys, signatures and vText is identical to that of Game_{1-k-2} if $\beta = 1$, i.e., $T_\beta = g^t g_2^b g_3^\varsigma$. □

Lemma 4.7. $\text{Game}_{2-(k-1)-2}$ *and* Game_{2-k-1} *are indistinguishable under DSG2 assumption and collision resistant property of H_s. That is, for every adversary $\mathscr{A}$, there exists PPT algorithm $\mathscr{B}$ such that* $|\mathsf{Adv}^{2-(k-1)-2}_{\mathscr{A},\text{ABS}}(\kappa) - \mathsf{Adv}^{2-k-1}_{\mathscr{A},\text{ABS}}(\kappa)| \leq \mathsf{Adv}^{\text{DSG2}}_{\mathscr{B}}(\kappa) + \mathsf{Adv}^{\text{CRH}_s}_{\mathscr{B}}(\kappa)$ *for* $1 \leq k \leq \nu_2$.

Proof. Similar to previous lemma, $\mathscr{B}$ receives an instance of DSG2, $(\mathscr{J}, g, Z_1Z_2, W_2W_3, Z_3, T_\beta)$ with $\beta \xleftarrow{\text{U}} \{0,1\}$ and depending on the distribution of β, $\mathscr{B}$ simulates either $\text{Game}_{2-(k-1)-2}$ or Game_{2-k-1}.

Setup: Similar to Lemma 4.3.

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

– KeyGen: Let A be a key query. The answer will be the sf-type 2 key. It picks $t \xleftarrow{\text{U}} \mathbb{Z}_N$, $R'_0, R_i \xleftarrow{\text{U}} \mathbb{G}_{p_3}$ for $i \in A$ and returns the following key to $\mathscr{A}$:

  $$\mathscr{SK}_A := [A,\ K := g^{\alpha+at}(W_2W_3)^t,\ L := g^t R'_0,\ K_i := T_i^t R_i,\ \forall i \in A].$$

  Since, $t \mod p_2$ and $t \mod p_3$ are uncorrelated, so the key $\mathscr{SK}_A$ is properly distributed sf-type 2 key.

– Sign: Let $(m^{(j)}, A_j, \Gamma_s^{(j)} := (\mathbf{M}_s, \rho_s))$ be the j^{th} query tuple, where $\mathbf{M}_s$ is an $\ell_s \times n_s$ matrix. If $\Gamma_s^{(j)}(A_j) =$ False, returns $\perp$. The first $(k-1)$ signatures are of sf-type 2 and the last $(\nu_2 - k)$ are normal signatures. The k^{th} signature is normal in $\text{Game}_{2-(k-1)-2}$ and sf-type 1 in Game_{2-k-1}. It chooses $r_s, \tilde{t}, \tau \xleftarrow{U} \mathbb{Z}_N$, $\tilde{R}_0, \tilde{R}_i, \tilde{R'}_i \xleftarrow{U} \mathbb{G}_{p_3}$ for $i \in [\ell_s]$. It then, chooses sets $\mathscr{I}_{A_j} \subset [\ell_s]$ and $\{\alpha_s^{(i)}\}_{i \in \mathscr{I}_{A_j}}$ such that $\sum_{i \in \mathscr{I}_{A_j}} \alpha_s^{(i)} \mathbf{M}_s^{(i)} = \mathbf{1}$. $\mathscr{B}$ sets $\alpha_s^{(i)} := 0$ for $i \notin \mathscr{I}_{A_j}$. It selects $\beta \xleftarrow{U} \{\beta = (\beta_1, \ldots, \beta_{\ell_s}) \in \mathbb{Z}_N^{\ell_s} \mid \sum_{i \in [\ell_s]} \beta_i \mathbf{M}_s^{(i)} = \mathbf{0}\}$. It sets $\hbar_s^{(j)} := H_s(m^{(j)}, \Gamma_s^{(j)})$. Then, $\mathscr{B}$ answers the signature δ_j for $(m^{(j)}, A_j, \Gamma_s^{(j)})$ as given below:

- If $j > k$, it is normal. $\mathscr{B}$ can handle it using $\mathscr{MSK}$.
- If $j < k$, it is sf-type 2. $\mathscr{B}$ computes the components of δ_j as follows:
  $\mathbf{S}_0 := \big(g^{\alpha + a\tilde{t}} (u_s^{\hbar_s^{(j)}} v_s)^{r_s} (W_2 W_3)^{\tilde{t}},\ g^{r_s} \tilde{R}_0 \big)$,
  $\mathbf{S}_i := \big((g^{\tilde{t}})^{\alpha_s^{(i)}} (g^{\tau})^{\beta_i} \tilde{R}_i,\ (T^{\tilde{t}}_{\rho_s(i)})^{\alpha_s^{(i)}} (T^{\tau}_{\rho_s(i)})^{\beta_i} \tilde{R'}_i \big)$ for $i \in [\ell_s]$.
  Since, $\tilde{t} \mod p_2$ and $\tilde{t} \mod p_3$ are uncorrelated, so δ_j is properly distributed sf-type 2 signature.
- If $j = k$ then it is either normal or sf-type 1 signature. $\mathscr{B}$ generates δ_k using T_β of the instance of DSG2, where it implicitly sets $g^{r_s} := T_\beta\big|_{\mathbb{G}_{p_1}}$. The components of δ_j are given as follows:
  $\mathbf{S}_0 := \big(g^{\alpha + a\tilde{t}} (T_\beta)^{\hbar_s^{(k)} a_s + b_s},\ T_\beta \big)$,
  $\mathbf{S}_i := \big((g^{\tilde{t}})^{\alpha_s^{(i)}} (g^{\tau})^{\beta_i} \tilde{R}_i,\quad (T^{\tilde{t}}_{\rho_s(i)})^{\alpha_s^{(i)}} (T^{\tau}_{\rho_s(i)})^{\beta_i} \tilde{R'}_i \big)$ for $i \in [\ell_s]$.

Forgery: $\mathscr{A}$ outputs a signature δ^* for $(m^*, \Gamma_s^* := (\mathbf{M}_s^*, \rho_s^*))$, where $\mathbf{M}_s^*$ is a matrix of order $\ell_s^* \times n_s^*$. Then, $\mathscr{B}$ prepares a vText $\mathscr{V}$ for (m^*, Γ_s^*) as follows. It computes $\hbar_s^* := H_s(m^*, \Gamma_s^*)$. It selects $\mathbf{v}_s' := (1, v_2', \ldots, v_{n_s^*}')$, where $v_2', \ldots, v_{n_s^*}' \xleftarrow{U} \mathbb{Z}_N$. For $i \in [\ell_s^*]$, it sets $\lambda_s^{*(i)} :=< \mathbf{M}_s^{*(i)}, \mathbf{v}_s' >$. It chooses $r_i' \xleftarrow{U} \mathbb{Z}_N$ for $i \in [\ell_s^*]$ and computes the following components:

$\mathbf{V}_0 := \big(Z_1 Z_2,\ (Z_1 Z_2)^{\hbar_s^* a_s + b_s},\ e(g^\alpha, Z_1 Z_2) \big)$,
$\mathbf{V}_i := \big((Z_1 Z_2)^{a \lambda_s^{*(i)}} (Z_1 Z_2)^{-r_i' t_{\rho^*(i)}},\quad (Z_1 Z_2)^{r_i'} \big)$, for $i \in [\ell_s^*]$.

The final vText is $\mathscr{V} := (\mathbf{V}_0, \{\mathbf{V}_i\}_{i \in [\ell_s^*]})$. $\mathscr{B}$ verifies the signature δ^* using the vText $\mathscr{V}$ and returns 1 if it is valid else 0.

Analysis: It is easily verified that if $\beta = 0$, i.e., $T_\beta = g^{r_s} g_3^{\varsigma}$, then the joint distribution of keys, signatures and vText is identical to that of $\text{Game}_{2-(k-1)-2}$. Now, suppose $\beta = 1$, i.e., $T_\beta = g^{r_s} g_2^b g_3^{\varsigma}$. Let $Z_1 Z_2 = g^s g_2^c$. $\mathscr{B}$ implicitly sets $\tilde{d} := b(\hbar_s^{(k)} a_s + b_s), \tilde{b} := b$ and $\iota := c(\hbar_s^* a_s + b_s)$ in k^{th} signature and vText respectively. Since, in the security definition 2.6, the adversary must not forge δ^* for a pair (m^*, Γ_s^*) for which $\mathscr{A}$ has made a signature query. This implies that $(m^*, \Gamma_s^*) \neq (m^{(k)}, \Gamma_s^{(k)})$. Since, H_s is a collision resistant hash function, we have $\hbar_s^* \neq \hbar_s^{(k)}$. By the Proposition 2.2, we have $\hbar_s^* a_s + b_s$ and $\hbar_s^{(k)} a_s + b_s$ are uniformly and independently distributed over $\mathbb{Z}_{p_2}$. Therefore, the joint distribution of keys, signatures and vText is identical to that of Game_{2-k-1} if $\beta = 1$, i.e., $T_\beta = g^{r_s} g_2^b g_3^{\varsigma}$. □

Lemma 4.8. 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* $|\text{Adv}^{2-k-1}_{\mathscr{A},\text{ABS}}(\kappa) - \text{Adv}^{2-k-2}_{\mathscr{A},\text{ABS}}(\kappa)| \leq \text{Adv}^{\text{DSG2}}_{\mathscr{B}}(\kappa)$ *for* $1 \leq k \leq \nu_2$.

Proof. It is similar to the proof of Lemma 4.7, except, answering the k^{th} signature query. $\mathscr{B}$ is given an instance of DSG2, $(\mathscr{J}, g, Z_1 Z_2, W_2 W_3, Z_3, T_\beta)$ with $\beta \xleftarrow{U} \{0,1\}$ and depending on the distribution of β, $\mathscr{B}$ simulates either Game_{2-k-1} or Game_{2-k-2}. Described here only is the construction of k^{th} signature

- The k^{th} signature is either sf-type 1 or sf-type 2. $\mathscr{B}$ generates δ_k using T_β of the instance of DSG2, where it implicitly sets $g^{r_s} := T_\beta\big|_{\mathbb{G}_{p_1}}$. The components of δ_k are given as follows:

$$\mathbf{S}_0 := \big(g^{\alpha+a\tilde{t}}(T_\beta)^{h_s^{(k)}a_s+b_s}(W_2W_3)^{\tilde{t}}, T_\beta \big),$$
$$\mathbf{S}_i := \big((g^{\tilde{t}})^{\alpha_s^{(i)}}(g^{\tau})^{\beta_i}\tilde{R}_i,\ (T_{\rho_s(i)}^{\tilde{t}})^{\alpha_s^{(i)}}(T_{\rho_s(i)}^{\tau})^{\beta_i}\tilde{R'}_i \big) \text{ for } i \in [\ell_s].$$

Note that an extra term, $(W_2W_3)^{\tilde{t}}$ is added to the first component of $\mathbf{S}_0$. Unlike to Lemma 4.7, we do not require the restriction argument between the replied signatures and vText. It is straightforward that if $\beta = 0$ (resp. $\beta = 1$), the joint distribution of keys, signatures and vText is identical to that of Game_{2-k-1} (resp. Game_{2-k-2}). □

Lemma 4.9. Game_{2-v_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* $|\mathsf{Adv}^{\text{Final}}_{\mathscr{A},\text{ABS}}(\kappa) - \mathsf{Adv}^{2-v_2-2}_{\mathscr{A},\text{ABS}}(\kappa)| \leq \mathsf{Adv}^{\text{DSG3}}_{\mathscr{B}}(\kappa)$

Proof. The simulator $\mathscr{B}$ receives an instance of DSG3, $(\mathscr{J}, g, g^{\alpha}X_2, g^{s}Y_2, Z_2, Z_3, T_\beta)$ with $\beta \xleftarrow{\text{U}} \{0,1\}$ and depending on the distribution of β, it simulates either Game_{2-v_2-2} or Game_{Final}.

Setup: $\mathscr{B}$ chooses $a, a_s, b_s \xleftarrow{\text{U}} \mathbb{Z}_N$ and $t_i \xleftarrow{\text{U}} \mathbb{Z}_N$ for $i \in \mathscr{U}$. Then, it sets $u_s := g^{a_s}, v := g^{b_s}$ and $T_i := g^{t_i}$ for $i \in \mathscr{U}$. $\mathscr{B}$ picks a hash function $H_s : \{0,1\}^* \longrightarrow \mathbb{Z}_N$. It provides $\mathscr{PP} := (\mathscr{J}, g, g^a, u_s, v_s, g_T^{\alpha} := e(g, g^{\alpha}X_2), \{T_i\}_{i\in\mathscr{U}}, Z_3, H_s)$ to $\mathscr{A}$. In this case, $\mathscr{B}$ does not know the master secret $\mathscr{MSK}$.

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

– KeyGen: Let A be a key query. The answer will be the sf-type 2 key. It picks $t \xleftarrow{\text{U}} \mathbb{Z}_N,\ R_0, R_0', R_i \xleftarrow{\text{U}} \mathbb{G}_{p_3}$ for $i \in A$ and returns the following key to $\mathscr{A}$:

$$\mathscr{SK}_A := [A,\ K := (g^{\alpha}X_2)(g^aZ_2)^tR_0,\ L := g^tR_0',\ K_i := T_i^tR_i,\ \forall i \in A].$$

Since, $t \mod p_1$ and $t \mod p_2$ are uncorrelated, so the key $\mathscr{SK}_A$ is properly distributed sf-type 2 key.

– Sign: Let (m, A, Γ_s) be a signature query made by $\mathscr{A}$. If $\Gamma_s(A) = \mathsf{False}$, returns $\perp$. It is sf-type 2. Let $\Gamma_s := (\mathbf{M}_s, \rho_s)$, where $\mathbf{M}_s$ is an $\ell_s \times n_s$ matrix. It chooses $r_s, \tilde{t}, \tau \xleftarrow{\text{U}} \mathbb{Z}_N,\ \tilde{R}, \tilde{R}_0, \tilde{R}_i, \tilde{R'}_i \xleftarrow{\text{U}} \mathbb{G}_{p_3}$ for $i \in [\ell_s]$. It then, chooses sets $\mathscr{I}_A \subset [\ell_s]$ and $\{\alpha_s^{(i)}\}_{i\in\mathscr{I}_A}$ such that $\sum_{i\in\mathscr{I}_A}\alpha_s^{(i)}\mathbf{M}_s^{(s)} = \mathbf{1}$. $\mathscr{B}$ sets $\alpha_s^{(i)} := 0$ for $i \notin \mathscr{I}_A$. It selects $\beta \xleftarrow{\text{U}} \{\beta = (\beta_1, \ldots, \beta_{\ell_s}) \in \mathbb{Z}_N^{\ell_s} \mid \sum_{i\in[\ell_s]}\beta_i\mathbf{M}_s^{(i)} = \mathbf{0}\}$. It sets $\hbar_s := H_s(m, \Gamma_s)$. Then, $\mathscr{B}$ answers the signature δ for (m, A, Γ_s) as follows:

$$\mathbf{S}_0 := \big((g^{\alpha}X_2)(g^aZ_2)^{\tilde{t}}(u_s^{\hbar_s}v_s)^{r_s}\tilde{R}, g^{r_s}\tilde{R}_0 \big),$$
$$\mathbf{S}_i := \big((g^{\tilde{t}})^{\alpha_s^{(i)}}(g^{\tau})^{\beta_i}\tilde{R}_i, \qquad (T_{\rho_s(i)}^{\tilde{t}})^{\alpha_s^{(i)}}(T_{\rho_s(i)}^{\tau})^{\beta_i}\tilde{R'}_i \big) \text{ for } i \in [\ell_s].$$

It is easy to check that δ_j is properly distributed sf-type 2 signature.

Forgery: $\mathscr{A}$ outputs a signature δ^* for $(m^*, \Gamma_s^* := (\mathbf{M}_s^*, \rho_s^*))$, where $\mathbf{M}_s^*$ is a matrix of order $\ell_s^* \times n_s^*$. Then $\mathscr{B}$ prepares a vText $\mathscr{V}$ for (m^*, Γ_s^*) as follows. It computes $\hbar_s^* := H_s(m^*, \Gamma_s^*)$. It selects $\mathbf{v}_s' := (1, v_2', \ldots, v_{n_s^*}')$, where $v_2', \ldots, v_{n_s^*}' \xleftarrow{\text{U}} \mathbb{Z}_N$. For $i \in [\ell_s^*]$, it sets $\lambda_s^{*(i)} := < \mathbf{M}_s^{*(i)}, \mathbf{v}_s' >$. It chooses $r_i' \xleftarrow{\text{U}} \mathbb{Z}_N$ for $i \in [\ell_s^*]$ and computes the vText $\mathscr{V}$ as given below:

$$\mathbf{V}_0 := \big(g^sY_2,\ (g^sY_2)^{\hbar_s^*a_s+b_s},\ T_\beta \big),$$
$$\mathbf{V}_i := \big((g^sY_2)^{a\lambda_s^{*(i)}}(g^sY_2)^{-r_i't_{\rho_s^*(i)}},\quad (g^sY_2)^{r_i'} \big), \text{ for } i \in [\ell_s^*].$$

$\mathscr{B}$ verifies the signature δ^* using the vText $\mathscr{V}$ and returns 1 if it is valid else returns 0.

Analysis. It is easy to check that if $\beta = 0$, i.e., $T_\beta = g_T^{\alpha s}$ (resp. $\beta = 1$, i.e., T_β is uniformly and independently distributed over $\mathbb{G}_T$), then the joint distribution of keys, signatures and vText is identical to that of Game_{2-v_2-2} (resp. Game_{Final}). □

5 Proposed ABSC: Construction 1

In this section, we present our basic signcryption-policy attribute-based signcryption (SCP-ABSC) for monotone span programs. The scheme is based on composite order bilinear pairing groups. Here we consider two policies, sender's policy $\Gamma_s := (\mathbf{M}_s, \rho_s)$ and receiver's policy $\Gamma_e := (\mathbf{M}_e, \rho_e)$. Similar to Section 3, the row labeling functions ρ_s and ρ_e are assumed to be injective in our basic SCP-ABSC scheme. From this basic ABSC scheme, a complete ABSC scheme can be constructed using the mechanism of Section 10, where ρ_s and ρ_e are no more assumed to be injective.

This construction works in the flavor of $\mathscr{C}t\mathscr{E}\&\mathscr{S}$ paradigm. To construct our scheme, we use a commitment scheme with hiding and relaxed-binding properties, CP-ABE scheme [25] and the ABS scheme described in Section 3. Let $\mathsf{ABS} := (\mathsf{ABS.Setup}, \mathsf{ABS.KeyGen}, \mathsf{ABS.Sign}, \mathsf{ABS.Ver})$ and $\mathscr{C} := (\mathsf{CSetup}, \mathsf{Commit}, \mathsf{Open})$ be respectively the ABS scheme described in Section 3 and commitment scheme.

– $\mathsf{Setup}(1^\kappa, \mathscr{U})$: It runs $\mathsf{CSetup}(1^\kappa) \longrightarrow \mathscr{CK}$ and $\mathsf{ABS.Setup}(1^\kappa, \mathscr{U}) \longrightarrow (\mathscr{ABS}.\mathscr{PP}, \mathscr{ABS}.\mathscr{MSK})$. It chooses $a_e, b_e \xleftarrow{\mathrm{U}} \mathbb{Z}_N$ and sets $u_e := g^{a_e}, v_e := g^{b_e}$. Let $H_e : \{0,1\}^* \longrightarrow \mathbb{Z}_N$ be a hash functions. The public parameters (combining $\mathscr{ABS}.\mathscr{PP}, \mathscr{CK}$ and u_e, v_e, H_e) and master secret are given by

$$\mathscr{PP} := (\mathscr{I}, g, g^a, u_s, u_e, v_s, v_e, g_T^\alpha, \{T_i\}_{i\in\mathscr{U}}, Z_3, H_s, H_e, \mathscr{CK}),$$
$$\mathscr{MSK} := \mathscr{ABS}.\mathscr{MSK} = (\alpha).$$

– $\mathsf{KeyGen}(\mathscr{PP}, \mathscr{MSK}, A)$: It runs $\mathscr{SK}_A \longleftarrow \mathsf{ABS.KeyGen}(\mathscr{ABS}.\mathscr{PP}, \mathscr{MSK}, A)$.

– $\mathsf{Signcrypt}(\mathscr{PP}, m, \mathscr{SK}_A, \Gamma_s := (\mathbf{M}_s, \rho_s), \Gamma_e := (\mathbf{M}_e, \rho_e))$: Let $\mathbf{M}_s$ (resp. $\mathbf{M}_e$) be an $\ell_s \times n_s$ (resp. $\ell_e \times n_e$) matrix. It runs $(\mathsf{com}, \mathsf{decom}) \longleftarrow \mathsf{Commit}(m)$. The $\mathsf{Signcrypt}$ algorithm has two part, Sign and $\mathsf{Encrypt}$, both run in parallel and then it is followed by the computation of an extra component C_{ℓ_e+1}.

Sign: $\delta := (\mathbf{S}_0, \{\mathbf{S}_i\}_{i\in[\ell_s]}) \longleftarrow \mathsf{ABS.Sign}(\mathscr{ABS}.\mathscr{PP}, \mathsf{com}||\Gamma_e, \mathscr{SK}_A, \Gamma_s := (\mathbf{M}_s, \rho_s))$, where the components are given by

$$\mathbf{S}_0 := \left(g^{\alpha+a\tilde{t}}(u_s^{\hbar_s} v_s)^{r_s} \tilde{R},\ g^{r_s}\tilde{R}_0 \right), \text{ where } \hbar_s := H_s(\mathsf{com}||\Gamma_e, \Gamma_s),$$
$$\mathbf{S}_i := \left((g^{\tilde{t}})^{\alpha_s^{(i)}} (g^\tau)^{\beta_i} \tilde{R}_i,\ (T_{\rho_s(i)}^{\tilde{t}})^{\alpha_s^{(i)}} (T_{\rho_s(i)}^\tau)^{\beta_i} \tilde{R'}_i \right).$$

$\mathsf{Encrypt}$: It picks $\mathbf{u}_e := (s_e, u_2, \ldots, u_{n_e}) \xleftarrow{\mathrm{U}} \mathbb{Z}_N^{n_e}$ and $r_e^{(i)} \xleftarrow{\mathrm{U}} \mathbb{Z}_N$ for $i \in [\ell_e]$. Let $\mathbf{M}_e^{(i)}$ denote the i^{th} row of the matrix, $\mathbf{M}_e$ and let $\lambda_e^{(i)} :=< \mathbf{M}_e^{(i)}, \mathbf{u}_e >$. The ciphertext components of the signcryption are given by

$$\mathbf{C}_0 := \left(g^{s_e}, \mathsf{decom} \cdot g_T^{\alpha s_e} \right),$$
$$\mathbf{C}_i := \left(g^{a\lambda_e^{(i)}} T_{\rho_e(i)}^{-r_e^{(i)}},\quad g^{r_e^{(i)}} \right) \text{ for } i \in [\ell_e].$$

Now, it sets $\mathsf{C} := (\mathbf{C}_0, \{\mathbf{C}_i\}_{i\in[\ell_e]})$ and computes $\hbar_e := H_e(\mathsf{com}, \mathsf{C}, \delta)$. Then, it computes the last component

$$C_{\ell_e+1} := (u_e^{\hbar_e} v_e)^{s_e}.$$

So, the ciphertext part of the signcryption is $\mathsf{CT} := (\mathsf{C}, C_{\ell_e+1})$.

It outputs the signcryption $\mathsf{U} := (\mathsf{com}, \delta, \mathsf{CT})$.

– $\mathsf{Unsigncrypt}(\mathscr{PP}, \mathsf{U}, \mathscr{SK}_B, \Gamma_s := (\mathbf{M}_s, \rho_s))$: Let $\Gamma_e := (\mathbf{M}_e, \rho_e)$ be the policy for receiver implicitly contained in U. $\mathbf{M}_s$ (resp. $\mathbf{M}_e$) be an $\ell_s \times n_s$ (resp. $\ell_e \times n_e$) matrix. This algorithm consists of two algorithms, Ver and $\mathsf{Decrypt}$, both run in parallel.

Ver: $\mathsf{flag} \longleftarrow \mathsf{ABS.Ver}(\mathscr{PP}, \mathsf{com}||\Gamma_e, \delta, \Gamma_s)$. If $\mathsf{flag} = 0$, it returns $\perp$.

Decrypt: It computes $\hbar_e := H_e(\mathsf{com}, \mathsf{C}, \delta)$ and picks $R' \in \mathbb{G}_{p_3}$. Then it checks $e(gR', C_{\ell_e+1}) \stackrel{?}{=} e(u_e^{\hbar_e} v_e, C_{01})$ and if the equality does not hold, it returns $\perp$. If $\Gamma_e(B) = \mathsf{False}$, it returns $\perp$, else computes the sets $\mathscr{I}_B \subset [\ell_e]$ and $\{\alpha_e^{(i)}\}_{i \in \mathscr{I}_B}$ such that $\sum_{i \in \mathscr{I}_B} \alpha_e^{(i)} \mathbf{M}_e^{(i)} = \mathbf{1}$. Then, it picks $r \xleftarrow{\mathrm{U}} \mathbb{Z}_N$, $R_0 \xleftarrow{\mathrm{U}} \mathbb{G}_{p_3}$ and computes the following:

$$\Delta_e := \frac{e(K(u_e^{\hbar_e} v_e)^r, C_{01})}{e(g^r R_0, C_{\ell_e+1}) \cdot \prod_{i \in \mathscr{I}_B} (e(L, C_{i1}) \cdot e(K_{\rho_e(i)}, C_{i2}))^{\alpha_e^{(i)}}}$$
$$\text{and } m := \mathsf{Open}(\mathsf{com}, C_{02}/\Delta_e).$$

Finally, it returns the message m.

Correctness. It follows from the correctness of Ver and Decrypt algorithms. Since, the correctness of Ver is immediate from that of ABS in Section 3, we illustrate here only the correctness of Decrypt.

$$\begin{aligned}
\Delta_e &= \frac{e(K(u_e^{\hbar_e} v_e)^r, C_{01})}{e(g^r R_0, C_{\ell_e+1}) \cdot \prod_{i \in \mathscr{I}_B} (e(L, C_{i1}) \cdot e(K_{\rho_e(i)}, C_{i2}))^{\alpha_e^{(i)}}} \\
&= \frac{g_T^{\alpha s_e + a t s_e} \cdot e(u_e^{\hbar_e} v_e, g)^{r s_e}}{e(u_e^{\hbar_e} v_e, g)^{r s_e} \cdot \prod_{i \in \mathscr{I}_B} (g_T^{a t \lambda_e^{(i)} - t \rho_e(i) r_e^{(i)}} \cdot g_T^{t \rho_e(i) r_e^{(i)}})^{\alpha_e^{(i)}}} \\
&= \frac{g_T^{\alpha s_e + a t s_e}}{\prod_{i \in \mathscr{I}_B} g_T^{a t \alpha_e^{(i)} \lambda_e^{(i)}}} = \frac{g_T^{\alpha s_e + a t s_e}}{g_T^{a t \sum_{i \in \mathscr{I}_B} \alpha^{(i)}{}_e \lambda_e^{(i)}}} = g_T^{\alpha s_e}
\end{aligned}$$

$$\mathsf{Open}(\mathsf{com}, C_{02}/\Delta_e) = \mathsf{Open}(\mathsf{com}, \mathsf{decom}) = m.$$

A high level description. In the construction above, we apparently use an IND-CCA version of the CP-ABE scheme of [25] which has the same public parameters and keys as the primitive attribute-based signature scheme ABS. Let $\mathsf{ABE} := (\mathsf{ABE.Setup}, \mathsf{ABE.KeyGen}, \mathsf{ABE.Encrypt}, \mathsf{ABE.Decrypt})$ be the CP-ABE scheme involved in above construction. The descriptions of Signcrypt and Unsigncrypt are given below.

- $\mathsf{Signcrypt}(\mathscr{PP}, m, \mathscr{SK}_A, \Gamma_s, \Gamma_e)$: It runs $(\mathsf{com}, \mathsf{decom}) \longleftarrow \mathsf{Commit}(m)$. Then, executes in parallel $\mathsf{C} \longleftarrow \mathsf{ABE.Encrypt}(\mathscr{PP}, \mathsf{decom}, \Gamma_e)$ and $\delta \longleftarrow \mathsf{ABS.Sign}(\mathscr{PP}, \mathsf{com}||\Gamma_e, \Gamma_s)$. Then, it computes $\hbar_e := H_e(\mathsf{com}, \mathsf{C}, \delta)$ and $C_{\ell_e+1} \longleftarrow \mathsf{fun}(\mathscr{PP}, \hbar_e, s_e)$, for some function fun (see footnote [4]). It sets $\mathsf{CT} := (\mathsf{C}, C_{\ell_e+1})$ and outputs $\mathsf{U} := (\mathsf{com}, \delta, \mathsf{CT})$.
- $\mathsf{Unsigncrypt}(\mathscr{PP}, \mathsf{U}, \mathscr{SK}_B, \Gamma_s, \Gamma_e)$: It runs $\mathsf{flag} \longleftarrow \mathsf{ABS.Ver}(\mathscr{PP}, \mathsf{com}||\Gamma_e, \delta, \Gamma_s)$ and $\mathsf{decom} \longleftarrow \mathsf{ABE.Decrypt}(\mathscr{PP}, \mathsf{CT}, \mathscr{SK}_B, \Gamma_e)$ in parallel. If $\mathsf{flag} = 1$, it returns $\mathsf{Open}(\mathsf{com}, \mathsf{decom})$ else $\perp$.

Remark 5.1. We note that the computation of C_{ℓ_e+1} is a part of the algorithm ABE.Encrypt. Since, it depends upon δ_w, its computation is delayed. To avoid the conflict between the output of ABE.Encrypt and input of ABE.Decrypt, we assume the understanding that $\mathsf{CT} := (\mathsf{C}, C_{\ell_e+1})$ is produced by ABE.Encrypt.

[4] $\mathsf{fun}(\mathscr{PP}, \hbar_e, s_e) = (u_e^{\hbar_e} v_e)^{s_e}$, where s_e is the secret involved in ABE.Encrypt.

Non-repudiation. Let $\mathsf{U} := (\mathsf{com}, \delta, \mathsf{CT})$ be a signcryption generated for a message (m, Γ_s, Γ_e) by a sender whose set of attributes satisfies a policy Γ_s. Let Bob be a receiver for the above signcryption. Bob extracts out decom from the ciphertext component CT and gives $(\mathsf{com}, \delta, \mathsf{decom}, m, \Gamma_s)$ to a third party. Since, the primitive commitment scheme $\mathscr{C}$ has relaxed-binding property, given a valid pair $(\mathsf{com}, \mathsf{decom})$ for m, Bob can not fool the third party by giving decom' such that $(\mathsf{com}, \mathsf{decom}')$ is also a valid pair for m' with $m' \neq m$. The objective of Bob is to convince the third party that the received message (m, Γ_s) was sent by a user whose set of attributes satisfies Γ_s. Now, the third party can verify the signature part δ of U against com by the verification algorithm of ABS. If it fails, it means that Bob fails to convince the third party. Otherwise, the third part checks $m \stackrel{?}{=} \mathsf{Open}(\mathsf{com}, \mathsf{decom})$. If equality holds, then the third is convinced that m was actually sent by the claimed sender (whose set of attributes satisfies Γ_s) else Bob fails.

6 Security of Construction 1

6.1 Perfect Privacy

Theorem 6.1. *The proposed attribute-based signcryption scheme in Section 5 is perfectly private (Definition 2.9).*

Proof. It is immediate from Theorem 4.1. □

6.2 Adaptive-Predicates IND-CCA Security

We prove the adaptive-predicates IND-CCA security of our basic ABSC scheme using dual system methodology of Brent Waters [47]. To utilize this methodology we define a new form of key called unsigncrypt query key. This key will be used to answer the unsigncrypt query in the security proof. In this methodology, we also define the semi-functional signcryptions, semi-functional unsigncrypt-query keys and semi-functional keys. Considered here are six types of semi-functional signcryptions, viz., type I, type II, type 1, type 2, type 3 and type 4. Two forms of keys are defined here – type 1 and type 2. Our semi-functional unsigncrypt-query keys consist of two forms, viz., type 1 and type 2. In the sequence of games, the challenge signcryption is first changed from normal to semi-functional type 1. Then, each queried key is changed from normal to semi-functional type 1, then semi-functional type 1 to type 2. Then, each replied signcryption is changed from normal to semi-functional type II via semi-functional type I. Similarly, each unsigncrypt-query key is changed from normal to semi-functional type 1, then from type 1 to type 2. Again, the challenge signcryption is changed from semi-functional type 1 to type 2 and then from type 2 to type 3. In the final game, the semi-functional type 3 challenge signcryption is changed to semi-functional type 4, where the decommitment part decom_b is masked with a random element from $\mathbb{G}_T$ to compute the C_{02} part of the challenge signcryption. Therefore, in the final game, the challenge message m_b is completely hidden unless the commitment part com_b of m_b leaks any information.

In the following material, the part framed by a box indicates that either it will be changed in next description or it has been changed from previous description. We use the abbreviation 'sf' and 'uq' for 'semi-functional' and 'unsigncrypt-query' respectively.

Semi-functional type I signcryption. This is same as normal signcryption except the signature component $\mathbf{S}_0$ described below:

$\mathbf{S}_0 := \left(g^{\alpha + a\tilde{r}} (u_s^{\hbar_s} v_s)^{r_s} \tilde{R} \boxed{g_2^{\tilde{d}}},\ g^{r_s} \tilde{R}_0 \boxed{g_2^{\tilde{b}}} \right)$, where $\tilde{b}, \tilde{d} \xleftarrow{\mathrm{U}} \mathbb{Z}_N$.

Semi-functional type II signcryption. This is same as sf-type I signcryption except $\tilde{b} = 0$, i.e.,
$\mathbf{S}_0 := \big(g^{\alpha+a\tilde{t}}(u_s^{\hbar_s} v_s)^{r_s} \tilde{R} g_2^{\tilde{d}},\ g^{r_s}\tilde{R}_0 \big)$.

Semi-functional type 1 signcryption. Pick $c, \iota \xleftarrow{U} \mathbb{Z}_N,\ \mathbf{v}_e \xleftarrow{U} \mathbb{Z}_N^{n_s}$. For each $i \in [\ell_e]$, pick $\gamma_e^{(i)} \xleftarrow{U} \mathbb{Z}_N$. For each $i \in \mathscr{U}$, choose $z_i \xleftarrow{U} \mathbb{Z}_N$. The sf-type 1 signcryption is obtained by modifying normal signcryption, viz., the ciphertext part but the signature part will be same as in normal signcryption. Let $\mathsf{U} = (\mathsf{com}, \delta, \mathsf{CT})$ be a normal signcryption, where the ciphertext part is $\mathsf{CT} = (\mathsf{C}, C_{\ell_e+1})$ and $\mathsf{C} = (\mathbf{C}_0, \{\mathbf{C}_i\}_{i\in[\ell_e]})$. So, the sf-type 1 signcryption is obtained by modifying the ciphertext part of the normal signcryption and which is given below:

$\mathbf{C}_0 := \big(g^{s_e} \boxed{g_2^c}, \mathsf{decom}.g_T^{\alpha s_e} \big)$,

$\mathbf{C}_i := \big(g^{a\lambda_e^{(i)}} T_{\rho_e(i)}^{-r_e^{(i)}} \boxed{g_2^{<\mathbf{M}_e^{(i)}, \mathbf{v}_e> + \gamma_e^{(i)} z_{\rho_e(i)}}},\quad g^{r_e^{(i)}} \boxed{g_2^{-\gamma_e^{(i)}}} \big)$, for $i \in [\ell_e]$,

$C_{\ell_e+1} := (u_e^{\hbar_e} v_e)^{s_e} \boxed{g_2^{\iota}}$.

Semi-functional type 2 signcryption. This is same as sf-type 1 signcryption except the signature part, i.e., $\delta = (\mathbf{S}_0, \{\mathbf{S}_i\}_{i\in[\ell_s]})$ gets changed from normal to the following form:

$\mathbf{S}_0 := \big(g^{\alpha+a\tilde{t}}(u_s^{\hbar_s} v_s)^{r_s} \tilde{R} \boxed{g_2^{\tilde{d}}},\ g^{r_s}\tilde{R}_0 \boxed{g_2^{\tilde{b}}} \big)$, where $\tilde{b}, \tilde{d} \xleftarrow{U} \mathbb{Z}_N$,

$\mathbf{S}_i := \big((g^{\tilde{t}})^{\alpha_s^{(i)}} (g^{\tau})^{\beta_i} \tilde{R}_i,\quad (T_{\rho_s(i)}^{\tilde{t}})^{\alpha_s^{(i)}} (T_{\rho_s(i)}^{\tau})^{\beta_i} \tilde{R'}_i \big)$.

Semi-functional type 3 signcryption. This is same as sf-type 2 signcryption except $\tilde{b} = 0$, i.e.,
$\mathbf{S}_0 := \big(g^{\alpha+a\tilde{t}}(u_s^{\hbar_s} v_s)^{r_s} \tilde{R} g_2^{\tilde{d}},\ g^{r_s}\tilde{R}_0 \big)$,
$\mathbf{S}_i := \big((g^{\tilde{t}})^{\alpha_s^{(i)}} (g^{\tau})^{\beta_i} \tilde{R}_i,\quad (T_{\rho_s(i)}^{\tilde{t}})^{\alpha_s^{(i)}} (T_{\rho_s(i)}^{\tau})^{\beta_i} \tilde{R'}_i \big)$.

Semi-functional type 4 signcryption. This is same as sf-type 3 signcryption except the following:
$\mathbf{C}_0 := \big(g^{s_e} g_2^c, \boxed{\mathsf{decom}.\hat{g}_t} \big)$, where $\hat{g}_t \xleftarrow{U} \mathbb{G}_T$.

Semi-functional type 1 key. Choose $b, d \xleftarrow{U} \mathbb{Z}_N$. First create a normal key
$\mathscr{SK}_A := [A,\ K := g^{\alpha+at} R,\ L := g^t R'_0,\ K_i := T_i^t R_i\ \forall i \in A]$
and then modify it to sf-type 1 key as shown below:
$\mathscr{SK}_A := [A,\ K := g^{\alpha+at} R \boxed{g_2^d},\ L := g^t R'_0 \boxed{g_2^b},\ K_i := T_i^t R_i \boxed{g_2^{bz_i}}\ \forall i \in A]$.

Semi-functional type 2 key. This is same as sf-type 1 key except $b = 0$, i.e.,
$\mathscr{SK}_A := [A,\ K := g^{\alpha+at} R g_2^d,\ L := g^t R'_0,\ K_i := T_i^t R_i\ \forall i \in A]$.

Note that the sf-type II signcryption can be computed using the original Signcrypt algorithm, where the input key is considered to be of sf-type 2.

Normal unsigncrypt query key. Let $(\mathsf{U}, A, \Gamma_s)$ be an unsigncrypt query made by $\mathscr{A}$, where $\mathsf{U} = (\mathsf{com}, \delta, \mathsf{CT})$ and $\mathsf{CT} = (\mathsf{C}, C_{\ell_e+1})$. Let $\hbar_e = H_e(\mathsf{com}, \mathsf{C}, \delta)$. Choose $r \xleftarrow{U} \mathbb{Z}_N,\ R_0 \xleftarrow{U} \mathbb{G}_{p_3}$. First create a normal key
$\mathscr{SK}_A := [A,\ K := g^{\alpha+at} R,\ L := g^t R'_0,\ K_i := T_i^t R_i\ \forall i \in A]$

and then modify it to normal unsigncrypt query key as shown below:

$$\mathscr{USK}_A := [A, K' := g^{\alpha+at} R \boxed{(u_e^{\hbar_e} v_e)^r}, \boxed{K_0 := g^r R_0}, L := g^t R'_0, K_i := T_i^t R_i \ \forall i \in A].$$

This $\mathscr{USK}_A$ will be used to unsigncrypt the queried signcryption.

Semi-functional type 1 unsigncrypt query key. Choose $b, d \xleftarrow{U} \mathbb{Z}_N$. First create a normal unsigncrypt query key as below

$$\mathscr{USK}_A := [A, K' := g^{\alpha+at} R (u_e^{\hbar_e} v_e)^r, K_0 := g^r R_0, L := g^t R'_0, K_i := T_i^t R_i \ \forall i \in A]$$

and then modify it to sf-type 1 unsigncrypt query key as shown below:

$$\mathscr{USK}_A := [A, K' := g^{\alpha+at} R (u_e^{\hbar_e} v_e)^r g_2^d, K_0 := g^r R_0, L := g^t R'_0, K_i := T_i^t R_i \ \forall i \in A].$$

Semi-functional type 2 unsigncrypt query key. This is same as sf-type 1 unsigncrypt query key except $b = 0$, i.e.,

$$\mathscr{USK}_A := [A, K' := g^{\alpha+at} R (u_e^{\hbar_e} v_e)^r g_2^d, K_0 := g^r R_0, L := g^t R'_0, K_i := T_i^t R_i \ \forall i \in A].$$

Answering Unsigncrypt Query Using uq-key. Let $(\mathsf{U}, A, \Gamma_s)$ be an unsigncrypt query. First, create a uq-key $\mathscr{USK}_A := [A,\ K',\ K_0,\ L,\ K_i\ \forall i \in A]$ of desired type. Then, the query will be handled in a similar manner as in Unsigncrypt algorithm, except the Decrypt algorithms, viz., the computation Δ_e. Described below is the required computation for Δ_e to answer unsigncrypt query.

$$\Delta_e := \frac{e(K', C_{01})}{e(K_0, C_{\ell_e+1}) \cdot \prod_{i \in \mathscr{I}_A} (e(L, C_{i1}) \cdot e(K_{\rho_e(i)}, C_{i2}))^{\alpha_e^{(i)}}}$$

Note that this modified Unsigncrypt is same as the original Unsigncrypt algorithm. The only difference is that in the original version, K_0 and K' are computed during unsigncrypt, whereas in the modified Unsigncrypt, K_0 and K' are supplied. Therefore, the unsigncrypt using normal uq-key gives the same result as the unsigncrypt using actual normal key.

A normal key can extract the message from a legitimate normal signcryption as well as sf-type 1 signcryption. But, if an sf-type 1 (resp. sf-type 2) key unsigncrypts a legitimate sf-type 1 signcryption, we have an additional factor $e(g_2, g_2)^{cd-bv_1}$ (resp. $e(g_2, g_2)^{cd}$) in Δ_e, where v_1 is the first component of $\mathbf{v}_e$. An sf-type 1 key is said to be nominally semi-functional if $cd - bv_1 = 0$. In this case, an sf-type 1 key can extract the message from a legitimate sf-type 1 signcryption.

Theorem 6.2. *If DSG1, DSG2 and DSG3 assumptions hold in $\mathscr{J}$, H_e is a collision resistant hash function and $\mathscr{C}$ has hiding property, then our proposed basic SCP-ABSC scheme in Section 5 is IND-CCA secure in adaptive-predicates model (Definition 2.10).*

Proof. Suppose there are at most ν_1 key queries, ν_2 unsigncrypt queries and ν_3 signcrypt queries made by an adversary $\mathscr{A}$. Then the security proof consists of hybrid argument over a sequence of $2(\nu_1 + \nu_2 + \nu_3) + 5$ games which are defined below:

– Game_{Real} := Original APs-IND-CCA game of ABSC scheme.

– Game_0 (:= Game_{1-0-2}) is same as Game_{Real} except the challenge signcryption is of sf-type 1.

– Game_{1-k-1} (for $1 \leq k \leq \nu_1$) is same as $\text{Game}_{1-(k-1)-2}$ except the k^{th} key is of sf-type 1.

– Game_{1-k-2} (for $1 \leq k \leq \nu_1$) is same as Game_{1-k-1} except the k^{th} key is sf-type 2.

– Game_{2-k-1} (for $1 \leq k \leq \nu_2$) is same as $\text{Game}_{2-(k-1)-2}$ except the k^{th} unsigncrypt query is answered by sf-type 1 uq-key. So, in this sequel, we define $\text{Game}_{2-0-2} := \text{Game}_{1-\nu_1-2}$.

– Game_{2-k-2} (for $1 \leq k \leq \nu_2$) is same as Game_{2-k-1} except the k^{th} unsigncrypt query is answered by sf-type 2 uq-key.

– Game_{3-k-1} (for $1 \leq k \leq \nu_3$) is same as $\text{Game}_{3-(k-1)-2}$ except the k^{th} replied signcryption is of sf-type I. So, in this sequel we define $\text{Game}_{3-0-2} := \text{Game}_{2-\nu_2-2}$.

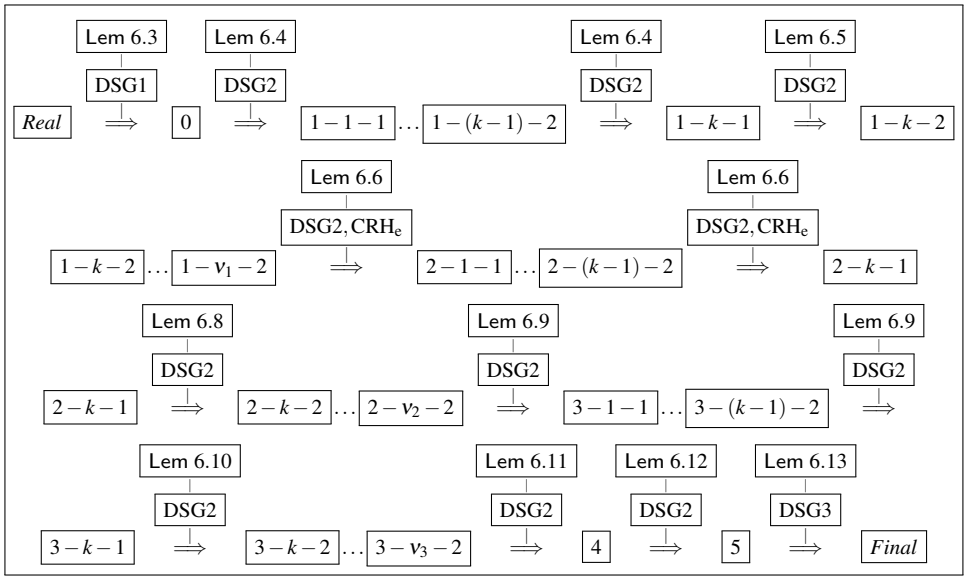
– Game_{3-k-2} (for $1 \leq k \leq \nu_3$) is same as Game_{3-k-1} except the k^{th} replied signcryption is of sf-type II.

– Game_4 is similar to $\text{Game}_{3-\nu_3-2}$ except that now the challenge signcryption is of sf-type 2.

– Game_5 is similar to Game_4 except that now the challenge signcryption is of sf-type 3.

– Game_{Final} is similar to Game_5 except that now 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 $\mathbb{G}_T$. This implies that the component C_{02} does not leak any information about decom_b. Since, the primitive commitment scheme $\mathscr{C}$ has hiding property, com_b does not reveal any information about 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 Section 6.3) and Lemma 6.14, we have the following reduction:

$$\mathsf{Adv}^{\text{APs-IND-CCA}}_{\mathscr{A},\text{ABSC}}(\kappa) \leq \mathsf{Adv}^{\text{DSG1}}_{\mathscr{B}_1}(\kappa) + 2(\nu_1+\nu_2+\nu_3)\mathsf{Adv}^{\text{DSG2}}_{\mathscr{B}_2}(\kappa) + \nu_2\mathsf{Adv}^{\text{CRH}_e}_{\mathscr{B}_3}(\kappa) + \\ + \mathsf{Adv}^{\text{DSG3}}_{\mathscr{B}_4}(\kappa) + \mathsf{Adv}^{\text{Hiding}}_{\mathscr{B}_5,\text{Commit}}(\kappa).$$

□

6.3 Lemmas Used in the Proof of Theorem 6.2

Lemma 6.3. Game_{Real} *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* $|\mathsf{Adv}^{\text{Real}}_{\mathscr{A},\text{ABSC}}(\kappa) - \mathsf{Adv}^{0}_{\mathscr{A},\text{ABSC}}(\kappa)| \leq \mathsf{Adv}^{\text{DSG1}}_{\mathscr{B}}(\kappa)$.

Proof. We construct a PPT algorithm $\mathscr{B}$ (called simulator) who receives an instance of DSG1, $(\mathscr{J}, g, Z_3, T_\beta)$ with $\beta \xleftarrow{\text{U}} \{0,1\}$ and depending on the distribution of β, it simulates either Game_{Real} or Game_0.

Setup: $\mathscr{B}$ runs $\mathsf{CSetup}(1^\kappa)$ to obtain the public commitment key $\mathscr{CK}$. $\mathscr{B}$ chooses $\alpha, a, a_s, a_e, b_s, b_e \xleftarrow{\text{U}} \mathbb{Z}_N$ and $t_i \xleftarrow{\text{U}} \mathbb{Z}_N$ for $i \in \mathscr{U}$. Then, it sets $u_s := g^{a_s}, u_e := g^{a_e}, v_s := g^{b_s}, v_e := g^{b_e}$ and $T_i := g^{t_i}$ for $i \in \mathscr{U}$. $\mathscr{B}$ selects hash functions $H_s, H_e : \{0,1\}^* \longrightarrow \mathbb{Z}_N$. It provides $\mathscr{PP} := (\mathscr{J}, g, g^a, u_s, u_e, v_s, v_e, g^\alpha_T, \{T_i\}_{i\in\mathscr{U}}, Z_3, H_s, H_e, \mathscr{CK})$ to $\mathscr{A}$ and keeps $\mathscr{MSK} := (\alpha)$ to itself.

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

- KeyGen: Let A be any key query made by $\mathscr{A}$. Since, $\mathscr{B}$ knows $\mathscr{MSK}$, it replies $\mathscr{SK}_A$ to $\mathscr{A}$.
- Signcrypt: It replies the normal signcryption. Let $\mathscr{A}$ makes a signcrypt query for the message $(m, A, \Gamma_s, \Gamma_e)$. $\mathscr{B}$ first computes $\mathscr{SK}_A$ using $\mathscr{MSK}$, then the normal signcryption is generated using $\mathscr{SK}_A$.
- Unsigncrypt: Let $(\mathsf{U}, B, \Gamma_s)$ be any unsigncrypt query made by $\mathscr{A}$, where $\mathsf{U} = (\mathsf{com}, \delta, \mathsf{CT})$. Since, the $\mathscr{MSK}$ is known to $\mathscr{B}$, it can construct the normal uq-key $\mathscr{USK}_B$ and then unsigncrypts it by the normal uq-key $\mathscr{USK}_B$ as described in Section 6.2.

Challenge: $\mathscr{A}$ provides two equal length messages m_0, m_1, a set of attributes A and the challenge access policies $\Gamma^*_s := (\mathbf{M}^*_s, \rho^*_s), \Gamma^*_e := (\mathbf{M}^*_e, \rho^*_e)$, where $\mathbf{M}^*_s$ (resp. $\mathbf{M}^*_e$) is an $\ell^*_s \times n^*_s$ (resp. $\ell^*_e \times n^*_e$) matrix. $\mathscr{B}$ picks $b \xleftarrow{\text{U}} \{0,1\}$. Then, it runs $(\mathsf{com}_b, \mathsf{decom}_b) \longleftarrow \mathsf{Commit}(m_b)$. It computes a normal key $\mathscr{SK}_A$ and executes $\mathsf{ABS.Sign}(\mathscr{ABSP}.\mathscr{PP}, \mathsf{com}_b || \Gamma^*_e, \mathscr{SK}_A, \Gamma^*_s)$ for the message $\mathsf{com}_b || \Gamma^*_e$ to have $\delta^* := (\mathbf{S}^*_0, \{\mathbf{S}^*_i\}_{i\in[\ell^*_s]})$, where the components are given by

$\mathbf{S}^*_0 := \big(g^{\alpha + a\tilde{t}} (u_s^{\hbar^*_s} v_s)^{r_s} \tilde{R}, g^{r_s} \tilde{R}_0 \big)$, where $\hbar^*_s := H_s(\mathsf{com}_b || \Gamma^*_e, \Gamma^*_s)$,

$\mathbf{S}^*_i := \big((g^{\tilde{t}})^{\alpha^{(i)}_s} (g^\tau)^{\beta_i} \tilde{R}_i,\ (T^{\tilde{t}}_{\rho_s(i)})^{\alpha^{(i)}_s} (T^\tau_{\rho_s(i)})^{\beta_i} \tilde{R'}_i \big)$.

It selects $\mathbf{v}'_e := (1, v'_2, \ldots, v'_{n^*_e})$, where $v'_2, \ldots, v'_{n^*_e} \xleftarrow{\text{U}} \mathbb{Z}_N$. For $i \in [\ell^*_e]$, it sets $\lambda^{*(i)}_e :=< \mathbf{M}^{*(i)}_e, \mathbf{v}'_e >$. It chooses $r'_i \xleftarrow{\text{U}} \mathbb{Z}_N$ for $i \in [\ell^*_e]$. $\mathscr{B}$ implicitly set g^{s_e} to be the $\mathbb{G}_{p_1}$ component of T_β. The ciphertext components of the signcryption are given by

$\mathbf{C}^*_0 := \big(T_\beta, \mathsf{decom}_b \cdot e(g^\alpha, T_\beta) \big)$,

$\mathbf{C}^*_i := \big(T_\beta^{a\lambda^{*(i)}_e} T_\beta^{-r'_i t_{\rho^*_e(i)}}, \quad T_\beta^{r'_i} \big)$ for $i \in [\ell^*_e]$.

Now, it sets $\mathsf{C}^* := (\mathbf{C}^*_0, \ldots, \mathbf{C}^*_{\ell^*_e})$ and computes $\hbar^*_e := H_e(\mathsf{com}_b, \mathsf{C}^*, \delta^*)$. Then it computes another ciphertext component as

$C_{\ell^*_e+1} := (T_\beta)^{a_e \hbar^*_e + b_e}$.

So, the ciphertext part of the signcryption is $\mathsf{CT}^* := (\mathsf{C}^*, C^*_{\ell^*_e+1})$. $\mathscr{B}$ returns the challenge signcryption $\mathsf{U}^* := (\mathsf{com}_b, \delta^*, \mathsf{CT}^*)$ to $\mathscr{A}$.

Query Phase-2: Similar to phase-1.

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

Analysis: First of all, note that the signature part δ^* of the challenge signcryption U^* is identical to that of the normal signcryption as well as sf-type 1 signcryption. Let us concentrate on the distribution of

ciphertext part CT^* of U^*. $\mathscr{B}$ implicitly sets $\mathbf{u}_e := s_e\mathbf{v}'_e = (s_e, s_e v'_2, \ldots, s_e v'_{n^*_e})$ and $r_e^{(i)} := s_e r'_i$ for $i \in [\ell^*_e]$. Since, $v'_2, \ldots, v'_{n^*_e}$ are chosen uniformly and independently from $\mathbb{Z}_N$, so the vector $\mathbf{u}_e$ is a random vector over $\mathbb{Z}_{p_1}$. Similarly, $r'_1, \ldots, r'_{\ell^*_e}$ are uniformly and independently distributed over $\mathbb{Z}_{p_1}$ as $r_e^{(1)}, \ldots, r_e^{(\ell^*_e)}$ are so over $\mathbb{Z}_N$. Therefore, U^* is a properly distributed normal signcryption if $\beta = 0$ (i.e., $T_\beta = g^{s_e}$).

Suppose $\beta = 1$, i.e., $T_\beta = g^{s_e} g_2^c$ for some $c \in \mathbb{Z}_N$. It implicitly sets $\mathbf{v}_e := ca\mathbf{v}'_e = (ca, cav'_2, \ldots, cav'_{n^*_e})$, $\iota := c(\hbar^*_e a_e + b_e)$, $\gamma_e^{(i)} := -cr'_i$ and $z_{\rho^*_e(i)} := t_{\rho^*_e(i)}$ for $i \in [\ell^*_e]$. By CRT, the values $v'_2, \ldots, v'_{n^*_e}$ and $r'_i, t_{\rho^*_e(i)}$ for $i \in [\ell^*_e]$ over $\mathbb{Z}_{p_1}$ are uncorrelated from those values over $\mathbb{Z}_{p_2}$. Therefore, U^* is a properly distributed sf-type 1 signcryption if $\beta = 1$. □

Lemma 6.4. $\mathrm{Game}_{1-(k-1)-2}$ *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* $|\mathsf{Adv}^{1-(k-1)-2}_{\mathscr{A},\mathrm{ABSC}}(\kappa) - \mathsf{Adv}^{1-k-1}_{\mathscr{A},\mathrm{ABSC}}(\kappa)| \leq \mathsf{Adv}^{\mathrm{DSG2}}_{\mathscr{B}}(\kappa)$ *for* $1 \leq k \leq \nu_1$.

Proof. $\mathscr{B}$ is given an instance of DSG2, $(\mathscr{J}, g, Z_1Z_2, W_2W_3, Z_3, T_\beta)$ with $\beta \xleftarrow{\mathrm{U}} \{0,1\}$ and depending on the distribution of β, $\mathscr{B}$ simulates either $\mathrm{Game}_{1-(k-1)-2}$ or Game_{1-k-1}.

Setup: Similar to Lemma 6.3.

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

- KeyGen: The first $(k-1)$ keys are of sf-type 2 and the last $(\nu_1 - k)$ are normal keys. The k^{th} key is normal in $\mathrm{Game}_{1-(k-1)-2}$ and sf-type 1 in Game_{1-k-1}. Let A_j be the j^{th} query set of attributes. $\mathscr{B}$ answers the key $\mathscr{SK}_{A_j}$ for A_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 sf-type 2 key. It picks $t \xleftarrow{\mathrm{U}} \mathbb{Z}_N$, $R'_0, R_i \xleftarrow{\mathrm{U}} \mathbb{G}_{p_3}$ for $i \in A_j$ and returns the following key to $\mathscr{A}$:
    $\mathscr{SK}_{A_j} := [A_j,\ K := g^{\alpha+at}(W_2W_3)^t,\ L := g^t R'_0,\ K_i := T_i^t R_i,\ \forall i \in A_j]$.
    Since, $t \mod p_2$ and $t \mod p_3$ are uncorrelated, so the key $\mathscr{SK}_{A_j}$ is properly distributed sf-type 2 key.
  - If $j = k$ then it is either normal or sf-type 1 key. $\mathscr{B}$ generates $\mathscr{SK}_{A_k}$ using T_β of the instance of DSG2. $\mathscr{B}$ implicitly sets g^t to be the $\mathbb{G}_{p_1}$ component of T_β. It chooses $R, R'_0, R_i \xleftarrow{\mathrm{U}} \mathbb{G}_{p_3}$ for $i \in A_k$ and returns the following key to $\mathscr{A}$:
    $\mathscr{SK}_{A_k} := [A_k,\ K := g^\alpha T_\beta^a R,\ L := T_\beta R'_0,\ K_i := T_\beta^{t_i} R_i,\ \forall i \in A_k]$.
- Signcrypt: Similar to Lemma 6.3.
- Unsigncrypt: Similar to Lemma 6.3.

Challenge: The signature part δ^* of U^* is generated in similar manner as in Lemma 6.3. The ciphertext part CT^* of the challenge signcryption U^* is constructed as follows. It selects $\mathbf{v}'_e := (1, v'_2, \ldots, v'_{n^*_e})$, where $v'_2, \ldots, v'_{n^*_e} \xleftarrow{\mathrm{U}} \mathbb{Z}_N$. For $i \in [\ell^*_e]$, it sets $\lambda_e^{*(i)} := < \mathbf{M}_e^{*(i)}, \mathbf{v}'_e >$. It chooses $r'_i \xleftarrow{\mathrm{U}} \mathbb{Z}_N$ for $i \in [\ell^*_e]$. The ciphertext components of the signcryption are given by:

$\mathbf{C}^*_0 := \big(Z_1Z_2,\ \mathsf{decom}_b \cdot e(g^\alpha, Z_1Z_2) \big)$,

$\mathbf{C}^*_i := \big((Z_1Z_2)^{a\lambda_e^{*(i)}} (Z_1Z_2)^{-r'_i t_{\rho^*_e(i)}},\quad (Z_1Z_2)^{r'_i} \big)$ for $i \in [\ell^*_e]$.

Now, it sets $\mathsf{C}^* := (\mathbf{C}_0^*, \ldots, \mathbf{C}_{\ell_e^*}^*)$ and then computes $\hbar_e^* := H_e(\mathsf{com}_b, \mathsf{C}^*, \delta^*)$. Then it computes another ciphertext component as $C_{\ell_e^*+1}^* := (Z_1Z_2)^{a_e\hbar_e^*+b_e}$. So, the ciphertext part of the challenge signcryption is $\mathsf{CT}^* := (\mathsf{C}^*, C_{\ell_e^*+1}^*)$.

Query Phase-2: Similar to phase-1.

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

Analysis: Let $Z_1Z_2 = g^{s_e}g_2^c$. $\mathscr{B}$ implicitly sets $\mathbf{u}_e := s_e\mathbf{v}'_e = (s_e, s_ev'_2, \ldots, s_ev'_{n_e^*})$ and $r_e^{(i)} := s_er'_i$ for $i \in [\ell_e^*]$. Since, $v'_2, \ldots, v'_{n_e^*}$ are chosen uniformly and independently from $\mathbb{Z}_N$, the vector $\mathbf{u}_e$ is a random vector over $\mathbb{Z}_{p_1}$. Similarly, $r'_1, \ldots, r'_{\ell_e^*}$ are uniformly and independently distributed over $\mathbb{Z}_{p_1}$ as $r_e^{(1)}, \ldots, r_e^{(\ell_e^*)}$ are so over $\mathbb{Z}_N$. It implicitly sets $\mathbf{v}_e := ca\mathbf{v}'_e = (ca, cav'_2, \ldots, cav'_{n_e^*})$, $\iota := c(\hbar_e^*a_e + b_e)$, $\gamma_e^{(i)} := -cr'_i$ and $z_{\rho_e^*(i)} := t_{\rho_e^*(i)}$ for $i \in [\ell_e^*]$. By CRT, the values $v'_2, \ldots, v'_{n_e^*}$ and $r'_i, t_{\rho_e^*(i)}$ for $i \in [\ell_e^*]$ over $\mathbb{Z}_{p_1}$ are uncorrelated from those values $\mathbb{Z}_{p_2}$. Hence, U^* is a properly distributed sf-type 1 signcryption. Therefore, the joint distribution of keys, signcryptions, uq-keys and challenge signcryption is identical to that of $\mathrm{Game}_{1-(k-1)-2}$ if $\beta = 0$ (i.e., $T_\beta = g^t g_3^\zeta$). Now, suppose $\beta = 1$, i.e., $T_\beta = g^t g_2^b g_3^\zeta$. $\mathscr{B}$ implicitly sets $d := ba$, $z_i := t_i$ for $i \in A_k$. Since, $a \mod p_1$ and $t_i \mod p_1$ are uncorrelated respectively from $a \mod p_2$ and $t_i \mod p_2$, $\mathscr{SK}_{A_k}$ is almost properly distributed sf-type 1 key except, the correlation between b and $d = ba$ (the exponents of g_2 in L and K resp.) also appears between c (the exponent of g_2 in C_{01}^*) and ac (first component of $\mathbf{v}_e$). Since, the adversary $\mathscr{A}$ is forbidden to ask for a key $\mathscr{SK}_A$ such that $\Gamma_e^*(A) = \mathsf{True}$ and ρ_e^* is injective, by Claim 4.5 the above correlation can be shown to be hidden to $\mathscr{A}$.

Therefore, the joint distribution of keys, signcryptions, uq-keys and challenge signcryption is identical to that of Game_{1-k-1} if $\beta = 1$ (i.e., $T_\beta = g^t g_2^b g_3^\zeta$). □

Lemma 6.5. Game_{1-k-1} *and* Game_{1-k-2} *are indistinguishable under the DSG2 assumption. That is, for every adversary* $\mathscr{A}$ *there exists a PPT algorithm* $\mathscr{B}$ *such that* $|\mathsf{Adv}^{1-k-1}_{\mathscr{A},\mathrm{ABSC}}(\kappa) - \mathsf{Adv}^{1-k-2}_{\mathscr{A},\mathrm{ABSC}}(\kappa)| \leq \mathsf{Adv}^{\mathrm{DSG2}}_{\mathscr{B}}(\kappa)$ *for* $1 \leq k \leq \nu_1$.

Proof. It is similar to that of Lemma 6.4, except the k^{th} key query answering. An instance of DSG2, $(\mathscr{J}, g, Z_1Z_2, W_2W_3, Z_3, T_\beta)$ with $\beta \xleftarrow{\mathrm{U}} \{0,1\}$ is given to the simulator $\mathscr{B}$ and depending on the distribution of β, it simulates either Game_{1-k-1} or Game_{1-k-2}. Described below is only the construction of k^{th} key.

- It is either sf-type 1 or sf-type 2 key. $\mathscr{B}$ generates $\mathscr{SK}_{A_k}$ using T_β of the instance of DSG2. $\mathscr{B}$ implicitly sets g^t to be the $\mathbb{G}_{p_1}$ component of T_β. It chooses $\zeta \xleftarrow{\mathrm{U}} \mathbb{Z}_N$, $R'_0, R_i \xleftarrow{\mathrm{U}} \mathbb{G}_{p_3}$ for $i \in A_k$ and returns the following key to $\mathscr{A}$:

$$\mathscr{SK}_{A_k} := [A_k,\ K := g^\alpha T_\beta^a (W_2W_3)^\zeta,\ L := T_\beta R'_0,\ K_i := T_\beta^{t_i} R_i,\ \forall i \in A_k].$$

In the above computation, an extra term $(W_2W_3)^\zeta$ is added to the component, K. Due to this additional part, $\mathbb{G}_{p_2}$ part of K becomes independent and uniform over $\mathbb{G}_{p_2}$. Hence, we do not require Claim 4.5. Rest of the proof is very straightforward. □

Lemma 6.6. $\mathrm{Game}_{2-(k-1)-2}$ *and* Game_{2-k-1} *are indistinguishable under DSG2 assumption and collision resistant property of* H_e. *That is, for every adversary* $\mathscr{A}$ *there exists PPT algorithm* $\mathscr{B}$ *such that*

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

Proof. Similar to previous lemma, $\mathscr{B}$ receives an instance of DSG2, $(\mathscr{J}, g, Z_1Z_2, W_2W_3, Z_3, T_\beta)$ with $\beta \xleftarrow{\mathrm{U}} \{0,1\}$ and depending on the distribution of β, $\mathscr{B}$ simulates either $\mathrm{Game}_{2-(k-1)-2}$ or Game_{2-k-1}.

Setup: Similar to Lemma 6.3.

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

– KeyGen: It is sf-type 2 key. Let A be a query set of attributes. It picks $t \xleftarrow{U} \mathbb{Z}_N,\ R'_0, R_i \xleftarrow{U} \mathbb{G}_{p_3}$ for $i \in A$ and returns the following key to $\mathscr{A}$:

$$\mathscr{SK}_A := [A,\ K := g^{\alpha+at}(W_2W_3)^t,\ L := g^t R'_0,\ K_i := T_i^t R_i,\ \forall i \in A].$$

Since, $t \mod p_2$ and $t \mod p_3$ are uncorrelated, so the key $\mathscr{SK}_A$ is properly distributed sf-type 2 key.

– Signcrypt: Similar to Lemma 6.3.

– Unsigncrypt: The first $(k-1)$ unsigncrypt queries are answered by sf-type 2 uq-keys and the last $(\nu_2 - k)$ are unsigncrypted by normal uq-keys. The k^{th} unsigncrypt query is handled by normal uq-key and sf-type 1 uq-key respectively in $\text{Game}_{2-(k-1)-2}$ and Game_{2-k-1}. Let $(\mathsf{U}_j, B_j, \Gamma_s^{(j)})$ be the j^{th} unsigncrypt query, where $\mathsf{U}_j = (\mathsf{com}_j, \delta_j, \mathsf{CT}_j)$, $\delta_j = (\mathbf{S}_0, \{\mathbf{S}_i\}_{i\in[\ell_s^{(j)}]})$, $\mathsf{CT}_j = (\mathsf{C}_j, C_{\ell_e^{(j)}+1})$, $\mathsf{C}_j = (\mathbf{C}_0, \{\mathbf{C}_i\}_{i\in[\ell_e^{(j)}]})$ and $\Gamma_e^{(j)}$ is the policy implicitly contained in U_j. $\mathscr{B}$ computes $\hbar_e^{(j)} := H_e(\mathsf{com}_j, \mathsf{C}_j, \delta_j)$. It picks $r, t \xleftarrow{U} \mathbb{Z}_N,\ R_0, R'_0, R_i \xleftarrow{U} \mathbb{G}_{p_3}$ for $i \in B_j$. Demonstrated below are the different types of uq-keys to be used to answer the unsigncrypt queries (as described in Section 6.2).

- If $j > k$, it is answered by normal uq-key. $\mathscr{B}$ can handle it using $\mathscr{MSK}$.
- If $j < k$, it is handled by sf-type 2 uq-key. $\mathscr{B}$ computes the sf-type 2 uq-key $\mathscr{USK}_{B_j}$ as below:

$$\mathscr{USK}_{B_j} := [B_j,\ K' := g^{\alpha+at}(u_e^{\hbar_e^{(j)}} v_e)^r (W_2W_3)^t,\ K_0 := g^r R_0,$$
$$L := g^t R'_0,\ K_i := T_i^t R_i\ \forall i \in B_j].$$

Since, $t \mod p_2$ and t over p_3 are uncorrelated, so the key $\mathscr{USK}_{B_j}$ is properly distributed sf-type 2 uq-key.

- If $j = k$, it is unsigncrypted either by normal or sf-type 1 uq-key. $\mathscr{B}$ generates $\mathscr{USK}_{B_k}$ using T_β of the instance of DSG2, where it implicitly sets g^r to be the $\mathbb{G}_{p_1}$ component of T_β. The components of uq-key are given below:

$$\mathscr{USK}_{B_k} := [B_k,\ K' := g^{\alpha+at}(T_\beta)^{h_e^{(k)}a_e+b_e},\ K_0 := T_\beta,$$
$$L := g^t R'_0,\ K_i := T_i^t R_i\ \forall i \in B_k].$$

Challenge: Similar to Lemma 6.4, but still we illustrate here the ciphertext part CT^* of the challenge signcryption U^*. $\mathscr{B}$ selects $\mathbf{v}'_e := (1, v'_2, \ldots, v'_{n_e^*})$, where $v'_2, \ldots, v'_{n_e^*} \xleftarrow{U} \mathbb{Z}_N$. For $i \in [\ell_e^*]$, it sets $\lambda_e^{*(i)} := < \mathbf{M}_e^{*(i)}, \mathbf{v}'_e >$. It chooses $r'_i \xleftarrow{U} \mathbb{Z}_N$ for $i \in [\ell_e^*]$ and computes the following ciphertext:

$\mathbf{C}_0^* := \big(Z_1Z_2,\ \mathsf{decom}_b \cdot e(g^\alpha, Z_1Z_2) \big)$,

$\mathbf{C}_i^* := \big((Z_1Z_2)^{a\lambda_e^{*(i)}} (Z_1Z_2)^{-r'_i t_{\rho_e^*(i)}},\quad (Z_1Z_2)^{r'_i} \big)$ for $i \in [\ell_e^*]$.

Now, it sets $\mathsf{C}^* := (\mathbf{C}_0^*, \ldots, \mathbf{C}_{\ell_e^*}^*)$ and then computes $\hbar_e^* := H_e(\mathsf{com}_b, \mathsf{C}^*, \delta^*)$. Then it computes the final component as

$C_{\ell_e^*+1}^* := (Z_1Z_2)^{a_e\hbar_e^* + b_e}$.

Query Phase-2: Similar to phase-1.

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

Analysis: Let $Z_1Z_2 = g^{s_e}g_2^c$. $\mathscr{B}$ implicitly sets $\mathbf{u}_e := s_e\mathbf{v}'_e = (s_e, s_e v'_2, \ldots, s_e v'_{n^*_e})$ and $r_e^{(i)} := s_e r'_i$ for $i \in [\ell^*_e]$. Since, $v'_2, \ldots, v'_{n^*_e}$ are chosen uniformly and independently from $\mathbb{Z}_N$, the vector $\mathbf{u}_e$ is a random vector over $\mathbb{Z}_{p_1}$. Similarly, $r'_1, \ldots, r'_{\ell^*_e}$ are uniformly and independently distributed over $\mathbb{Z}_{p_1}$ as $r_e^{(1)}, \ldots, r_e^{(\ell^*_e)}$ are so over $\mathbb{Z}_N$. It implicitly sets $\mathbf{v}_e := ca\mathbf{v}'_e = (ca, cav'_2, \ldots, cav'_{n^*_e})$, $\iota := c(\hbar^*_e a_e + b_e)$, $\gamma_e^{(i)} := -cr'_i$ and $z_{\rho^*_e(i)} := t_{\rho^*_e(i)}$ for $i \in [\ell^*_e]$. By CRT, the values $v'_2, \ldots, v'_{n^*_e}$ and $r'_i, t_{\rho^*_e(i)}$ for $i \in [\ell^*_e]$ over $\mathbb{Z}_{p_1}$ are uncorrelated from those values over $\mathbb{Z}_{p_2}$. It is easy to check that if $\beta = 0$, then the joint distribution of keys, signcryptions, uq-keys and challenge signcryption is identical to that of $\mathrm{Game}_{2-(k-1)-2}$.

Suppose $\beta = 1$, i.e., $T_\beta := g^r g_2^b g_3^\varsigma$. Let us take a look on the distributions of the exponents of g_2 in C^*_{01} and $C^*_{\ell^*_e+1}$, i.e., c and $\iota = c(\hbar^*_e a_e + b_e)$. This type of correlation does not hamper our task unless the similar type of correlation is found in other components. In k^{th} uq-key almost the similar correlation is found between the exponents of g_2 in K_0 and K', i.e., b and $d = b(h_e^{(k)} a_e + b_e)$. Using the Proposition 2.2 and Claim 6.7, we have $\hbar^*_e a_e + b_e$ and $h_e^{(k)} a_e + b_e$ are uniformly and independently distributed over $\mathbb{Z}_{p_2}$. Therefore, the joint distribution of keys, signcryptions, uq-keys and challenge signcryption is identical to that of Game_{2-k-1}.

Claim 6.7. $\hbar^*_e \neq h_e^{(k)}$.

Proof of Claim 6.7. Let us assume that

$$\boxed{\hbar^*_e = h_e^{(k)}} \tag{4}$$

By the natural restriction of the security game, we have

$$\boxed{\mathsf{U}^* \neq \mathsf{U}_k} \tag{5}$$

Since H is a collision resistant hash function, from the equation (4) and (5), we have

$$\boxed{C^*_{\ell^*_e+1} \neq C_{\ell_e^{(k)}+1} \text{ and } C^*_{01} = C_{01}} \tag{6}$$

From the definition of Unsigncrypt, we have

$$\boxed{C_{\ell_e^{(k)}+1}\big|_{\mathbb{G}_{p_3}} = 1_{\mathbb{G}} \text{ and } e(g, C_{\ell_e^{(k)}+1}) = e(u_e^{h_e^{(k)}} v_e, C_{01})} \tag{7}$$

From the challenge signcryption, we have

$$\boxed{C^*_{\ell^*_e+1}\big|_{\mathbb{G}_{p_3}} = 1_{\mathbb{G}} \text{ and } e(g, C_{\ell^*_e+1}) = e(u_e^{\hbar^*_e} v_e, C^*_{01})} \tag{8}$$

From the equation (4) and 2nd part of the equation (6), (7) and (8), we have

$$\boxed{C^*_{\ell^*_e+1}\big|_{\mathbb{G}_{p_1}} = C_{\ell_e^{(k)}+1}\big|_{\mathbb{G}_{p_1}}} \tag{9}$$

Since, $C^*_{\ell^*_e+1}\big|_{\mathbb{G}_{p_3}} = 1_{\mathbb{G}}$ and $C_{\ell_e^{(k)}+1}\big|_{\mathbb{G}_{p_3}} = 1_{\mathbb{G}}$, using equation (9), we have $Y_2 := (C^*_{\ell^*_e+1})^{-1}.C_{\ell_e^{(k)}+1} \in \mathbb{G}_{p_2}$. Since $C^*_{\ell^*_e+1} \neq C_{\ell_e^{(k)}+1}$ (by 1st part of the equation (6)), we have $Y_2 \neq 1_{\mathbb{G}}$. Therefore, $\mathscr{B}$ breaks the given instance of the DSG2 assumption.

□

Lemma 6.8. 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* $|\mathsf{Adv}^{2-k-1}_{\mathscr{A},\text{ABSC}}(\kappa) - \mathsf{Adv}^{2-k-2}_{\mathscr{A},\text{ABSC}}(\kappa)| \leq \mathsf{Adv}^{\text{DSG2}}_{\mathscr{B}}(\kappa)$ *for* $1 \leq k \leq \nu_2$.

Proof. It is similar to the proof of Lemma 6.6, except answering the k^{th} unsigncrypt query (i.e., k^{th} uq-key). $\mathscr{B}$ is given an instance of DSG2, $(\mathscr{J}, g, Z_1Z_2, W_2W_3, Z_3, T_\beta)$ with $\beta \xleftarrow{\text{U}} \{0,1\}$ and depending on the distribution of β, $\mathscr{B}$ simulates either Game_{2-k-1} or Game_{2-k-2}. Below, we only provide the simulation of k^{th} unsigncrypt query answering.

- The k^{th} signcryption is unsigncrypted either by sf-type 1 uq-key or sf-type 2 uq-key. $\mathscr{B}$ generates $\mathscr{US}\mathscr{K}_{B_k}$ using T_β of the instance of DSG2, where it implicitly sets g^r to be the $\mathbb{G}_{p_1}$ component of T_β.

$$\mathscr{US}\mathscr{K}_{B_k} := [B_k,\ K' := g^{\alpha+at}(T_\beta)^{h_e^{(k)}a_e+b_e}(W_2W_3)^t,\ K_0 := T_\beta,$$
$$L := g^t R'_0,\ K_i := T_i^t R_i\ \forall i \in B_k].$$

Note that an extra term, $(W_2W_3)^t$ is added to K'. Due to this additional term, the exponent of g_2 in K' can easily be shown to be independent without any condition. It is straightforward that if $\beta = 0$ (resp. $\beta = 1$), the joint distribution of keys, signcryptions, uq-keys and challenge signcryption is identical to that of Game_{2-k-1} (resp. Game_{2-k-2}). □

Lemma 6.9. $\text{Game}_{3-(k-1)-2}$ *and* Game_{3-k-1} *are indistinguishable under DSG2 assumption. That is, for every adversary* $\mathscr{A}$ *there exists a PPT algorithm* $\mathscr{B}$ *such that* $|\mathsf{Adv}^{3-(k-1)-2}_{\mathscr{A},\text{ABSC}}(\kappa) - \mathsf{Adv}^{3-k-1}_{\mathscr{A},\text{ABSC}}(\kappa)| \leq \mathsf{Adv}^{\text{DSG2}}_{\mathscr{B}}(\kappa)$ *for* $1 \leq k \leq \nu_3$.

Proof. An instance $(\mathscr{J}, g, Z_1Z_2, W_2W_3, Z_3, T_\beta)$ with $\beta \xleftarrow{\text{U}} \{0,1\}$ of DSG2 is given to $\mathscr{B}$ and depending on the distribution of β, $\mathscr{B}$ simulates either $\text{Game}_{3-(k-1)-2}$ or Game_{3-k-1}.

Setup: Similar to Lemma 6.3.

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

– KeyGen: Similar to Lemma 6.6.

– Signcrypt: The first $(k-1)$ replied signcryptions are of sf-type II and the last $(\nu_3 - k)$ are normal signcryptions. The k^{th} replied signcryption is normal in $\text{Game}_{3-(k-1)-2}$ and sf-type I in Game_{3-k-1}. Let $(m^{(j)}, A_j, \Gamma_s^{(j)}, \Gamma_e^{(j)})$ be the j^{th} signcrypt query made by $\mathscr{A}$. $\mathscr{B}$ answers the queries as follows.

- If $j > k$, then $\mathscr{B}$ first constructs a normal key $\mathscr{SK}_{A_j}$ by running KeyGen algorithm and then it computes j^{th} signcryption U_j using the key $\mathscr{SK}_{A_j}$ (as in Signcrypt algorithm). The simulator $\mathscr{B}$ replies the normal signcryption U_j to $\mathscr{A}$.
- If $j < k$, then it is sf-type II signcryption. It first computes the sf-type 2 key $\mathscr{SK}_{A_j}$, then using this key it produces sf-type II signcryption U_j (similar to the above case [5]).
- If $j = k$ then it is either normal or sf-type I signcryption. $\mathscr{B}$ generates signature part δ_j of the signcryption U_j using T_β of the instance of DSG2 and which is given below:

$$\mathbf{S}_0 := \left(g^{\alpha+a\tilde{t}}(T_\beta)^{h_s^{(j)}a_s+b_s}, T_\beta \right),$$
$$\mathbf{S}_i := \left((g^{\tilde{t}})^{\alpha_s^{(i)}}(g^\tau)^{\beta_i}\tilde{R}_i,\ (T^{\tilde{t}}_{\rho_s(i)})^{\alpha_s^{(i)}}(T^\tau_{\rho_s(i)})^{\beta_i}\tilde{R}'_i \right) \text{ for } i \in [\ell_s^{(j)}].$$

[5] Even the normal signcryption and sf-type II signcryption can be computed directly using $\mathscr{MSK}$ and the supplied parameters of the instance of DSG2

$\mathscr{B}$ implicitly sets g^{r_s} to be the $\mathbb{G}_{p_1}$ component of T_β. The rest part of the signcryption U_j is computed as described in Signcrypt algorithm.

– Unsigncrypt: It is unsigncrypted by sf-type 2 uq-key. Let $(\mathsf{U}, B, \Gamma_s)$ be an unsigncrypt query, where $\mathsf{U} = (\mathsf{com}, \delta, \mathsf{CT})$, $\delta = (\mathbf{S}_0, \{\mathbf{S}_i\}_{i\in[\ell_s]})$, $\mathsf{CT} = (\mathsf{C}, C_{\ell_e+1})$, $\mathsf{C} = (\mathbf{C}_0, \{\mathbf{C}_i\}_{i\in[\ell_e]})$ and Γ_e is the policy implicitly contained in U. $\mathscr{B}$ constructs the sf-type 2 uq-key $\mathscr{USK}_B$ (given below), then unsigncrypts U by $\mathscr{USK}_B$ as described in Section 6.2.

$$\mathscr{USK}_B := [B,\ K' := g^{\alpha+at}(u_e^{\hbar_e^{(j)}} v_e)^r (W_2W_3)^t,\ K_0 := g^r R_0,$$
$$L := g^t R_0',\ K_i := T_i^t R_i\ \forall i \in B].$$

Since, $t \mod p_2$ and $t \mod p_3$ are uncorrelated, so the key $\mathscr{USK}_B$ is properly distributed sf-type 2 uq-key.

Challenge: Similar to Lemma 6.6.

Query Phase-2: Similar to phase-1.

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

Analysis: It is very straightforward that if $\beta = 0$ (resp. $\beta = 1$) the distribution of the k^{th} replied signcryption is identical to normal (resp. sf-type I) signcryption. Therefore, the joint distribution of keys, signcryptions, uq-keys and the challenge signcryption is identical to that of $\mathrm{Game}_{3-(k-1)-2}$ (resp. Game_{3-k-1}) if $\beta = 0$ (resp. $\beta = 1$). □

Lemma 6.10. 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* $|\mathsf{Adv}^{3-k-1}_{\mathscr{A},\mathrm{ABSC}}(\kappa) - \mathsf{Adv}^{3-k-2}_{\mathscr{A},\mathrm{ABSC}}(\kappa)| \leq \mathsf{Adv}^{\mathrm{DSG2}}_{\mathscr{B}}(\kappa)$ *for* $1 \leq k \leq \nu_3$.

Proof. Similar to the proof of Lemma 6.9, except the S_{01} component of the signature part δ_k of the k^{th} signcrypt query. $\mathscr{B}$ is given an instance of DSG2, $(\mathscr{J}, g, Z_1Z_2, W_2W_3, Z_3, T_\beta)$ with $\beta \xleftarrow{\mathrm{U}} \{0,1\}$ and depending on the distribution of β, $\mathscr{B}$ simulates either Game_{3-k-1} or Game_{3-k-2}. Described here is only the $\mathbf{S}_0$ part of the signature part δ_k and which is given by:

$$\mathbf{S}_0 := \left(g^{\alpha+a\tilde{t}}(T_\beta)^{\hbar_s^* a_s + b_s}(W_2W_3)^{\tilde{t}}, T_\beta\ \right).$$

Due to the additional term $(W_2W_3)^{\tilde{t}}$, the exponent of g_2 in S_{01} becomes uniformly and independently distributed random variable over $\mathbb{Z}_{p_2}$. Rest of the proof can be handled in similar manner to as in the previous lemma. □

Lemma 6.11. $\mathrm{Game}_{3-\nu_3-2}$ *and* Game_4 *are indistinguishable under the DSG2 assumption. That is, for every adversary* $\mathscr{A}$ *there exists a PPT algorithm* $\mathscr{B}$ *such that* $|\mathsf{Adv}^{3-\nu_3-2}_{\mathscr{A},\mathrm{ABSC}}(\kappa) - \mathsf{Adv}^{4}_{\mathscr{A},\mathrm{ABSC}}(\kappa)| \leq \mathsf{Adv}^{\mathrm{DSG2}}_{\mathscr{B}}(\kappa)$

Proof. Similarly, $\mathscr{B}$ is given an instance of DSG2, $(\mathscr{J}, g, Z_1Z_2, W_2W_3, Z_3, T_\beta)$ with $\beta \xleftarrow{\mathrm{U}} \{0,1\}$ and depending on the distribution of β, $\mathscr{B}$ simulates either $\mathrm{Game}_{3-\nu_3-2}$ or Game_4.

Setup: Similar to Lemma 6.3.

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

– KeyGen: Similar to Lemma 6.6.

– Signcrypt: It is sf-type II signcryption. It is generated in similar manner as that of proof of Lemma 6.9.

– Unsigncrypt: Similar to Lemma 6.9.

Challenge: It is similar to Lemma 6.4 except the signature part δ^* of the challenge signcryption U^*. $\mathscr{B}$ generates δ^* using T_β of the instance of DSG2. The signature part δ^* is given below, where $\mathscr{B}$ implicitly sets g^{r_s} to be the $\mathbb{G}_{p_1}$ component of T_β:

$\mathbf{S}_0^* := \big(g^{\alpha+a\tilde{t}}(T_\beta)^{\hbar_s^* a_s+b_s},\ T_\beta \big)$,

$\mathbf{S}_i^* := \big((g^{\tilde{t}})^{\alpha_s^{(i)}}(g^{\tau})^{\beta_i}\tilde{R}_i,\quad (T^{\tilde{t}}_{\rho_s(i)})^{\alpha_s^{(i)}}(T^{\tau}_{\rho_s(i)})^{\beta_i}\tilde{R'}_i \big)$ for $i \in [\ell_s^*]$.

Query Phase-2: Similar to phase-1.

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

Analysis: Similar to Lemma 6.4, the distribution of the ciphertext part CT^* of the challenge signcryption U^* is identical to that of sf-type 1 and 2 signcryptions. Now, the rest of analysis depend upon the distribution of the signature part δ^* of U^*. It is easy to check that if $\beta = 0$ (resp. $\beta = 1$), the distribution of the signature part δ^* of U^* is identical to that of sf-type 1 (resp. sf-type 2) signcryption. Hence, if $\beta = 0$ (resp. $\beta = 1$) the distribution of the challenge signcryption U^* is identical to that of sf-type 1 (resp. sf-type 2) signcryption. Therefore, the joint distribution of keys, signcryptions, uq-keys and the challenge signcryption is identical to that of $\mathrm{Game}_{3-\nu_3-2}$ (resp. Game_4) if $\beta = 0$ (resp. $\beta = 1$). □

Lemma 6.12. Game_4 *and* Game_5 *are indistinguishable under the DSG2 assumption. That is, for every adversary* $\mathscr{A}$ *there exists a PPT algorithm* $\mathscr{B}$ *such that* $|\mathsf{Adv}^4_{\mathscr{A},\mathrm{ABSC}}(\kappa) - \mathsf{Adv}^5_{\mathscr{A},\mathrm{ABSC}}(\kappa)| \leq \mathsf{Adv}^{\mathrm{DSG2}}_{\mathscr{B}}(\kappa)$

Proof. It is similar to the proof of Lemma 6.11, except the signature part δ^* of the challenge signcryption U^*. $\mathscr{B}$ is given an instance of DSG2, $(\mathscr{J}, g, Z_1Z_2, W_2W_3, Z_3, T_\beta)$ with $\beta \xleftarrow{\mathrm{U}} \{0,1\}$ and depending on the distribution of β, $\mathscr{B}$ simulates either Game_4 or Game_5. Demonstrated here is only the signature part δ^* as given below:

$\mathbf{S}_0^* := \big(g^{\alpha+a\tilde{t}}(T_\beta)^{\hbar_s^* a_s+b_s}(W_2W_3)^{\tilde{t}}, T_\beta \big)$,

$\mathbf{S}_i^* := \big((g^{\tilde{t}})^{\alpha_s^{(i)}}(g^{\tau})^{\beta_i}\tilde{R}_i,\ (T^{\tilde{t}}_{\rho_s(i)})^{\alpha_s^{(i)}}(T^{\tau}_{\rho_s(i)})^{\beta_i}\tilde{R'}_i \big)$ for $i \in [\ell_s^*]$.

Rest of the proof are easily handled as it is similar to the previous lemma. □

Lemma 6.13. 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* $|\mathsf{Adv}^5_{\mathscr{A},\mathrm{ABSC}}(\kappa) - \mathsf{Adv}^{\mathrm{Final}}_{\mathscr{A},\mathrm{ABSC}}(\kappa)| \leq \mathsf{Adv}^{\mathrm{DSG3}}_{\mathscr{B}}(\kappa)$

Proof. The simulator $\mathscr{B}$ receives an instance of DSG3, $(\mathscr{J}, g, g^\alpha X_2, g^s Y_2, Z_2, Z_3, T_\beta)$ with $\beta \xleftarrow{\mathrm{U}} \{0,1\}$ and depending on the distribution of β, it simulates either Game_5 or Game_{Final}.

Setup: $\mathscr{B}$ runs $\mathsf{CSetup}(1^\kappa)$ to obtain the public commitment key $\mathscr{CK}$. $\mathscr{B}$ chooses $\alpha, a, a_s, a_e, b_s, b_e \xleftarrow{\mathrm{U}} \mathbb{Z}_N$ and $t_i \xleftarrow{\mathrm{U}} \mathbb{Z}_N$ for $i \in \mathscr{U}$. Then, it sets $u_s := g^{a_s}, u_e := g^{a_e}, v_s := g^{b_s}, v_e := g^{b_e}$ and $T_i := g^{t_i}$ for $i \in \mathscr{U}$. $\mathscr{B}$ selects hash functions $H_s, H_e : \{0,1\}^* \longrightarrow \mathbb{Z}_N$. It provides $\mathscr{PP} := (\mathscr{J}, g, g^a, u_s, u_e, v_s, v_e, g_T^\alpha := e(g, g^\alpha X_2), \{T_i\}_{i\in\mathscr{U}}, Z_3, H_s, H_e, \mathscr{CK})$ to $\mathscr{A}$. But $\mathscr{B}$ does not know the master secret $\mathscr{MSK}$.

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

– KeyGen: It is sf-type 2 key. Let A be a query set of attributes. It picks $t \xleftarrow{\mathrm{U}} \mathbb{Z}_N,\ R, R'_0, R_i \xleftarrow{\mathrm{U}} \mathbb{G}_{p_3}$ for $i \in A$ and returns the following key to $\mathscr{A}$:

$$\mathscr{SK}_A := [A,\ K := (g^\alpha X_2)(g^a Z_2)^t R,\ L := g^t R'_0,\ K_i := T_i^t R_i,\ \forall i \in A].$$

Since, $t \mod p_1$ and $t \mod p_2$ are uncorrelated, so the key $\mathscr{SK}_A$ is properly distributed sf-type 2 key.

– Signcrypt: It is sf-type II signcryption. Let $\mathscr{A}$ make a signcrypt query for the message $(m, A, \Gamma_s, \Gamma_e)$. $\mathscr{B}$ first computes an sf-type 2 key $\mathscr{SK}_A$ (as above), then using this key it produces the sf-type II signcryption U and returns it to $\mathscr{A}$.

– Unsigncrypt: It is unsigncrypted by sf-type 2 uq-key. Let $(\mathsf{U}, B, \Gamma_s)$ be an unsigncrypt query, where Γ_e is the policy implicitly contained in U. $\mathscr{B}$ constructs the sf-type 2 uq-key $\mathscr{USK}_B$ (given below), then unsigncrypts it by $\mathscr{USK}_B$.

$$\mathscr{USK}_B := [B,\ K' := (g^{\alpha} X_2)(g^a Z_2)^t (u_e^{\hbar_e^{(j)}} v_e)^r R,\ K_0 := g^r R_0,$$
$$L := g^t R'_0,\ K_i := T_i^t R_i\ \forall i \in B].$$

Since, $t \mod p_2$ and $t \mod p_2$ are uncorrelated, so the key $\mathscr{USK}_{B_j}$ is properly distributed sf-type 2 uq-key.

Challenge: The initial part similar to previous lemma. $\mathscr{B}$ generates CT^* using T_β of the instance of DSG3. The signature part $\delta^* = (\mathbf{S}_0^*, \{\mathbf{S}_i^*\}_{i \in [\ell_s^*]})$ is computed below:

$\mathbf{S}_0^* := \big((g^{\alpha} X_2)(g^a Z_2)^{\tilde{t}} (u_s^{\hbar_s^*} v_s)^{r_s} \tilde{R}_0,\ \ g^{r_s} \tilde{R}'_0 \big)$,

$\mathbf{S}_i^* := \big((g^{\tilde{t}})^{\alpha_s^{(i)}} (g^{\tau})^{\beta_i} \tilde{R}_i,\ \ (T_{\rho_s(i)}^{\tilde{t}})^{\alpha_s^{(i)}} (T_{\rho_s(i)}^{\tau})^{\beta_i} \tilde{R'}_i \big)$ for $i \in [\ell_s^*]$.

It selects $\mathbf{v}'_e := (1, v'_2, \ldots, v'_{n_e^*})$, where $v'_2, \ldots, v'_{n_e^*} \xleftarrow{\mathrm{U}} \mathbb{Z}_N$. For $i \in [\ell_e^*]$, it sets $\lambda_e^{*(i)} := < \mathbf{M}_e^{*(i)}, \mathbf{v}'_e >$. It chooses $r'_i \xleftarrow{\mathrm{U}} \mathbb{Z}_N$ for $i \in [\ell_e^*]$. The ciphertext components of the signcryption are given by

$\mathbf{C}_0^* := \big(g^s Y_2, \mathsf{decom}_b \cdot T_\beta \big)$,

$\mathbf{C}_i^* := \big((g^s Y_2)^{a \lambda_e^{*(i)}} (g^s Y_2)^{-r'_i t_{\rho_s^*(i)}},\ \ (g^s Y_2)^{r'_i} \big)$, for $i \in [\ell_e^*]$.

Now, it sets $\mathsf{C}^* := (\mathbf{C}_0^*, \ldots, \mathbf{C}_{\ell_e^*}^*)$ and then computes $\hbar_e^* := H_e(\mathsf{com}_b, \mathsf{C}^*, \delta^*)$. Then it computes the final component as

$C_{\ell_e^*+1}^* := (g^s Y_2)^{a_e \hbar_e^* + b_e}$.

So, the ciphertext part of the signcryption is $\mathsf{CT}^* := (\mathsf{C}^*, C_{\ell_e^*+1}^*)$. $\mathscr{B}$ returns the challenge signcryption $\mathsf{U}^* := (\mathsf{com}_b, \delta^*, \mathsf{CT}^*)$ to $\mathscr{A}$.

Query Phase-2: Similar to phase-1.

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

Analysis: It is obvious that if $\beta = 0$, i.e., $T_\beta := g_T^{\alpha s}$ (resp. if $\beta = 1$, i.e., $T_\beta \xleftarrow{\mathrm{U}} \mathbb{G}_T$) form of the challenge signcryption U^* is identical to that of sf-type 2 (resp. sf-type 3) signcryption. Therefore, the joint distribution of keys, signcryptions, uq-keys and challenge signcryption is identical to that of Game_5 (resp. Game_{Final}) if $\beta = 0$ (resp. $\beta = 1$). □

Lemma 6.14. *For every adversary $\mathscr{A}$ there exists a PPT algorithm $\mathscr{B}$ such that* $\mathsf{Adv}^{\mathrm{Final}}_{\mathscr{A},\mathrm{ABSC}}(\kappa) \leq \mathsf{Adv}^{\mathrm{Hiding}}_{\mathscr{B},\mathrm{Commit}}(\kappa)$

Proof. In Game_{Final}, the decommitment part decom_b is masked with random element from $\mathbb{G}_T$ or in other words we can say that the ciphertext part in the challenge signcryption is a encryption of a random message from the decommitment space. So, the ciphertext part of the challenge signcryption does not carry any information about the challenge message m_b. Therefore, the commitment part com_b may only have the information about m_b. Now, we show that if the primitive commitment scheme $\mathscr{C}$ has the hiding property, then the adversary $\mathscr{A}$ has no advantage in Game_{Final}. Suppose an adversary $\mathscr{A}$ has an

advantage in Game_{Final}, then we will construct an PPT algorithm $\mathscr{B}$ for breaking the hiding property of the commitment scheme $\mathscr{C}$. Let $\mathscr{CH}$ be the challenger for the commitment scheme $\mathscr{C}$.

Setup: $\mathscr{CH}$ runs $\mathsf{CSetup}(1^\kappa)$ and gives the public commitment key $\mathscr{CK}$ to $\mathscr{B}$. Now $\mathscr{B}$ executes $\mathscr{J} := (N := p_1p_2p_3, \mathbb{G}, \mathbb{G}_T, e) \longleftarrow \mathscr{G}_{\mathsf{cbg}}(1^\kappa)$ to have a composite order bilinear groups with known factorization p_1, p_2 and p_3 of N. Then, $\mathscr{B}$ sets the public parameters $\mathscr{PP} := (\mathscr{J}, g, g^a, u_s, u_e, v_s, v_e, g_T^\alpha, \{T_i\}_{i\in\mathscr{U}}, Z_3, H_s, H_e, \mathscr{CK})$ as per rule of the original setup and it gives $\mathscr{PP}$ to $\mathscr{A}$. Note that the simulator $\mathscr{B}$ knows all the secrets even including the generator of $\mathbb{G}_{p_2}$.

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

- KeyGen: It is of sf-type 2 key. $\mathscr{B}$ can generate the key as it knows all the secrets of signcryption.
- Signcrypt: It is sf-type II signcryption. By similar argument above, it is easily computable.
- Unsigncrypt: It is unsigncrypted by sf-type 2 uq-key. Let $(\mathsf{U}, B, \Gamma_s)$ be an unsigncrypt query, where Γ_e is the policy implicitly contained in U. $\mathscr{B}$ generates the sf-type 2 uq-key $\mathscr{USK}_B$ and then unsigncrypts it by $\mathscr{USK}_B$.

Challenge: $\mathscr{A}$ provides two equal length messages m_0, m_1, a set of attributes A and the challenge access policies $\Gamma_s^* := (\mathbf{M}_s^*, \rho_s^*), \Gamma_e^* := (\mathbf{M}_e^*, \rho_e^*)$, where $\mathbf{M}_s$ (resp. $\mathbf{M}_e$) is an $\ell_s^* \times n_s^*$ (resp. $\ell_e^* \times n_e^*$) matrix to the simulator $\mathscr{B}$. Then, $\mathscr{B}$ sends m_0, m_1 to $\mathscr{CH}$. Now $\mathscr{CH}$ chooses $m_b \xleftarrow{\text{U}} \{m_0, m_1\}$, runs $(\mathsf{com}_b, \mathsf{decom}_b) \longleftarrow \mathsf{Commit}(m_b)$ and returns challenge commitment part com_b to $\mathscr{B}$. Note that $\mathscr{B}$ does not know the decommitment part decom_b of the challenge message m_b, but it will not hamper the task of $\mathscr{B}$. $\mathscr{B}$ picks a random element d_r from the decommitment space. Then it runs the rest of Signcrypt algorithm on (com_b, d_r) using the policies Γ_s^* and Γ_e^* to produce the challenge signcryption[6] U^* of sf-type 4 and returns it to $\mathscr{A}$.

Guess: $\mathscr{A}$ sends a guess b' to $\mathscr{B}$ and $\mathscr{B}$ replies with the same guess b' to $\mathscr{CH}$.

Analysis: In sf-type 4 signcryption, the decommitment part decom_b is masked with a uniformly and independently chosen element form $\mathbb{G}_T$ which is same as masking a random element d_r with $g_T^{\alpha s_e}$. It is straightforward to check that the keys, signcryptions, uq-keys and the challenge signcryption are properly distributed. If $\mathscr{A}$ guesses correctly, then this guess will work for breaking hiding property as well. □

6.4 Adaptive-Predicates Weak Unforgeability

The unforgeability model used here is similar to the Definition 2.13 except that the adversary is not allowed to access the unsigncrypt oracle.

Theorem 6.15. *If DSG1, DSG2 and DSG3 assumptions hold in $\mathscr{J}$, the primitive commitment scheme $\mathscr{C}$ has relaxed-binding property and H_s is a collision resistant hash function, then the proposed basic SCP-ABSC scheme in Section 5 is existential unforgeable in adaptive-predicates model.*

Proof. It follows from Theorem 6.16 and Theorem 4.2. □

Theorem 6.16. *If the primitive attribute-based signature scheme* ABS *is adaptive-predicate existential unforgeable and the primitive commitment scheme $\mathscr{C}$ has relaxed-binding property, then the proposed SCP-ABSC scheme in Section 5 is existential unforgeable in adaptive-predicates model.*

[6] It first produces normal signcryption and then converts it to sf-type 4 using the generator of $\mathbb{G}_{p_2}$

Proof. Suppose an adversary $\mathscr{A}$ can break the adaptive-predicates weak unforgeability of the proposed SCP-ABSC scheme with non-negligible advantage ε. We assume that $\mathscr{A}$ has made ν number of signcrypt oracle queries. Let $(m^{(i)}, A_i, \Gamma_s^{(i)}, \Gamma_e^{(i)})$ be the i^{th} query and $\mathsf{U}_i = (\mathsf{com}_i, \delta_i, \mathsf{CT}_i)$ be the corresponding replied signcryption. Let $\mathsf{U}^* = (\mathsf{com}^*, \delta^*, \mathsf{CT}^*)$ be the forgery by $\mathscr{A}$ for the message $(m^*, \Gamma_s^*, \Gamma_e^*)$. Let Forged be the event that $\mathsf{com}^*||\Gamma_e^*||\Gamma_s^* \notin \{\mathsf{com}_i||\Gamma_e^{(i)}||\Gamma_s^{(i)} \mid i \in [\nu]\}$. Then, we have

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

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

1. Case: Forged**.** We establish a PPT algorithm $\mathscr{B}_{\mathsf{ABS}}$ (simulator) for forging to the primitive attribute-based signature scheme ABS with advantage at least $\varepsilon/2$. In this simulation, the algorithm $\mathscr{B}_{\mathsf{ABS}}$ will make use of the adversary $\mathscr{A}$. Let $\mathscr{CH}$ be the challenger for the primitive attribute-based signature scheme ABS.

Setup: First, the challenger $\mathscr{CH}$ publishes the public parameters $\mathscr{ABSPP}$ of ABS. Then, simulator $\mathscr{B}_{\mathsf{ABS}}$ runs $\mathsf{CSetup}(1^\kappa)$ to produce $\mathscr{CK}$. $\mathscr{B}_{\mathsf{ABS}}$ chooses $a_e, b_e \xleftarrow{\mathrm{U}} \mathbb{Z}_N$ and sets $u_e := g^{a_e}, v_e := g^{b_e}$. $\mathscr{B}_{\mathsf{ABS}}$ selects a hash function $H_e : \{0,1\}^* \longrightarrow \mathbb{Z}_N$. It provides $\mathscr{PP} := (\mathscr{J}, g, g^a, u_s, u_e, v_s, v_e, g_T^\alpha, \{T_i\}_{i\in\mathscr{U}}, Z_3, H_s, H_e, \mathscr{CK})$ to $\mathscr{A}$. Interpretation of the variables described here are same as in the ABSC scheme in Section 5.

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

– KeyGen: Since the ABSC scheme and the primitive ABS scheme have the identical key distribution, the key queries from $\mathscr{A}$ will be forwarded to the challenger $\mathscr{CH}$. Similarly, the answers (keys) will be reversed back to $\mathscr{A}$.

– $\mathsf{Signcrypt}$: Let us see how $\mathscr{B}_{\mathsf{ABS}}$ will answer the signcrypt queries of $\mathscr{A}$. Let $(m^{(i)}, A_i, \Gamma_s^{(i)}, \Gamma_e^{(i)})$ be the i^{th} signcrypt query to $\mathscr{B}_{\mathsf{ABS}}$ by $\mathscr{A}$. $\mathscr{B}_{\mathsf{ABS}}$ runs $(\mathsf{com}_i, \mathsf{decom}_i) \longleftarrow \mathsf{Commit}(m^{(i)})$. Then, $\mathscr{B}_{\mathsf{ABS}}$ makes a signature query for $(\mathsf{com}_i||\Gamma_e^{(i)}, A_i, \Gamma_s^{(i)})$ to $\mathscr{CH}$ and gets the replied signature δ_i from $\mathscr{CH}$. Then, $\mathscr{B}_{\mathsf{ABS}}$ runs encryption algorithm of the $\mathsf{Signcrypt}$ algorithm to produce the ciphertext part CT_i and it returns the i^{th} signcryption $\mathsf{U}_i := (\mathsf{com}_i, \delta_i, \mathsf{CT}_i)$ to $\mathscr{A}$.

$\mathsf{Forgery}$: $\mathscr{A}$ outputs a tuple $(\mathsf{U}^*, \Gamma_s^*, \Gamma_e^*)$. Then, $\mathscr{B}_{\mathsf{ABS}}$ forges the signature δ^* for $(\mathsf{com}^*||\Gamma_e^*, \Gamma_s^*)$ to the primitive attribute-based signature scheme ABS.

$\mathsf{Analysis}$: From the event Forged, it implies that $(\mathsf{com}^*||\Gamma_e^*, \Gamma_s^*)$ has not been queried for signature to $\mathscr{CH}$.

2. Case: $\neg\mathsf{Forged}$**.** We set up an algorithm $\mathscr{B}_{\mathsf{Commit}}$ (simulator) for breaking the relaxed-binding of the primitive commitment scheme $\mathscr{C}$ with advantage at least $\varepsilon/2\nu$. Let $\mathscr{CH}$ be the challenger for the primitive commitment scheme $\mathscr{C}$.

Setup: First, the challenger $\mathscr{CH}$ publishes the public commitment key $\mathscr{CK}$. Then, $\mathscr{B}_{\mathsf{Commit}}$ runs $\mathsf{ABS.Setup}(1^\kappa)$ (Setup algorithm of an ABS scheme $\mathscr{B}_{\mathsf{ABS}}$) to produce $\mathscr{ABSPP}$. Rest are same as above. Note that in this case, $\mathscr{B}_{\mathsf{Commit}}$ knows the $\mathscr{MSK}$. $\mathscr{B}_{\mathsf{Commit}}$ picks $j \xleftarrow{\mathrm{U}} [\nu]$ as a guess such that $\mathsf{com}^*||\Gamma_e^*||\Gamma_s^* = \mathsf{com}_j||\Gamma_e^{(j)}||\Gamma_s^{(j)}$.

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

– KeyGen: The simulator $\mathscr{B}_{\mathsf{Commit}}$ can handle the key queries as $\mathscr{MSK}$ is known to itself.

– Signcrypt: Let $(m^{(i)}, A_i, \Gamma_s^{(i)}, \Gamma_e^{(i)})$ be the i^{th} signcrypt query to $\mathscr{B}_{\mathsf{Commit}}$ by $\mathscr{A}$. If $i = j$, then $\mathscr{B}_{\mathsf{Commit}}$ makes a commitment query for the message $m^{(i)}$ to $\mathscr{C}\mathscr{H}$ to have a pair $(\mathsf{com}_i, \mathsf{decom}_i)$ else $\mathscr{B}_{\mathsf{Commit}}$ itself computes $(\mathsf{com}_i, \mathsf{decom}_i)$. After that, $\mathscr{B}_{\mathsf{Commit}}$ follows the rest of Signcrypt algorithm in Section 5 to produce the signcryption $\mathsf{U}_i := (\mathsf{com}_i, \delta_i, \mathsf{CT}_i)$ for $(m^{(i)}, \Gamma_s^{(i)}, \Gamma_e^{(i)})$ and returns it to $\mathscr{A}$.

Forgery: $\mathscr{A}$ outputs a tuple $(\mathsf{U}^*, \Gamma_s^*, \Gamma_e^*)$. By $\neg$Forged, we have $\mathsf{com}^*||\Gamma_e^*||\Gamma_s^* = \mathsf{com}_j||\Gamma_e^{(j)}||\Gamma_s^{(j)}$. Then, $\mathscr{B}_{\mathsf{Commit}}$ submits decom^* to $\mathscr{C}\mathscr{H}$ as a witness of breaking relaxed-binding property (for $(\mathsf{com}_j(=\mathsf{com}^*), \mathsf{decom}_j)$) of commitment scheme $\mathscr{C}$. Note that $\mathscr{B}_{\mathsf{Commit}}$ obtains decom^* by running the Unsigncrypt algorithm on input U^* as it knows the $\mathscr{M}\mathscr{S}\mathscr{K}$.

Analysis: With probability $1/\nu$, $\mathscr{B}_{\mathsf{Commit}}$ can correctly guess $j \in [\nu]$ such that $\mathsf{com}^*||\Gamma_e^*||\Gamma_s^* = \mathsf{com}_j||\Gamma_e^{(j)}||\Gamma_s^{(j)}$. It is easy to see that $m^* = \mathsf{Open}(\mathsf{com}^*, \mathsf{decom}^*)$ and $m^{(j)} = \mathsf{Open}(\mathsf{com}_j, \mathsf{decom}_j)$. To draw the conclusion, we have to show that $m^* \neq m^{(j)}$. Indeed, if $m^* = m^{(j)}$ and we already have $\mathsf{com}^*||\Gamma_e^*||\Gamma_s^* = \mathsf{com}_j||\Gamma_e^{(j)}||\Gamma_s^{(j)}$ implying $(m^{(j)}, \Gamma_s^{(j)}, \Gamma_e^{(j)}) = (m^*, \Gamma_s^*, \Gamma_e^*)$. Hence, it shows that $\mathsf{U}^* := (\mathsf{com}^*, \delta^*, \mathsf{CT}^*)$ is a forgery on $(m^{(j)}, \Gamma_s^{(j)}, \Gamma_e^{(j)})$, which is a contradiction to the definition of existential unforgeability of ABSC scheme. □

7 Proposed ABSC: Construction 2

In Section 5, we have provided weakly unforgeable and IND-CCA secure attribute-based signcryption in signcryption-policy form. In this section, we explore strongly unforgeable and IND-CCA secure SCP-ABSC scheme for the monotone span programs. The construction follows the similar paradigm as that of the construction in Section 5, except a strong OTS is applied at the outside layer. As mentioned earlier in this paper, we call this new paradigm as "Commit then Encrypt and Sign then Sign" ($\mathscr{C}t\mathscr{E}\&\mathscr{S}t\mathscr{S}$) paradigm. The technique [22] weakly unforgeable ABS is lifted to strongly unforgeable ABS, weakly unforgeable ABSC in Section 5 is lifted to strongly unforgeable ABSC scheme. But to maintain IND-CCA security of ABSC in Section 5, the OTS is to be applied cautiously in this new construction.

We give a short description of our strongly unforgeable and IND-CCA secure SCP-ABSC construction, since it follows from the SCP-ABSC in Section 5 and the idea of strongly unforgeable SP-ABS stated above. Let $\mathscr{C} :=$ (CSetup, Commit, Open) be a commitment scheme with hiding property (relaxed-binding property is not required). Let wABS := (wABS.Setup, wABS.KeyGen, wABS.Sign, wABS.Ver) and ABE := (ABE.Setup, ABE.KeyGen, ABE.Encrypt, ABE.Decrypt) be the ABS scheme and ABE scheme respectively used in Section 5. Let OTS := (OTS.Gen, OTS.Sign, OTS.Ver) be a strongly unforgeable one-time signature scheme. Demonstrated in Figure 4 are only two algorithms, Signcrypt and Unsigncrypt and rest are same as in Section 5. Let $\Delta_s := \mathsf{Signcrypt}(m, \mathscr{S}\mathscr{K}_A, \Gamma_s, \Gamma_e)$ and $\Delta_u := \mathsf{Unsigncrypt}(\mathsf{U}, \mathscr{S}\mathscr{K}_B, \Gamma_s)$.

Correctness. It follows from that of Section 5.

Remark 7.1. For strong unforgeability of the proposed ABSC, we do not require the relax-bonding property of the commitment scheme. But if we want the non-repudiation, then we have to assume the relax-binding property of the commitment scheme. In the proposed construction, if we replace the decom by the message m and ignore the commitment scheme, then the modified version will have the same performance as the proposed construction except the modified version does not guarantee the non-repudiation.

$$
\begin{aligned}
- \ \Delta_s := & \left(\begin{array}{c} (\mathsf{com}, \mathsf{decom}) \longleftarrow \mathsf{Commit}(m); \| (\mathsf{vk}, \mathsf{signk}) \longleftarrow \mathsf{OTS.Gen}(1^{\kappa}); \\ \delta_w \longleftarrow \mathsf{wABS.Sign}(\mathsf{vk}, \mathscr{SK}_A, \Gamma_s); \| \mathsf{C} \longleftarrow \mathsf{ABE.Encrypt}(\mathsf{decom}, \Gamma_e); \\ \text{where } \delta_w := (\mathbf{S}_0, \ldots, \mathbf{S}_{\ell_s}) \text{ and } \mathsf{C} := (\mathbf{C}_0, \ldots, \mathbf{C}_{\ell_e}); \\ \text{let } \hbar_e := H_e(\mathsf{com}, \mathsf{C}, \delta_w); C_{\ell_e+1} \longleftarrow \mathsf{fun}(\mathscr{PP}, \hbar_e, s_e); \\ \delta_o \longleftarrow \mathsf{OTS.Sign}(\hbar_e || C_{\ell_e+1} || \Gamma_e || \Gamma_s, \mathsf{signk}); \\ \text{returns } \mathsf{U} := (\mathsf{com}, \delta_s := (\delta_w, \delta_o, \mathsf{vk}), \mathsf{CT} := (\mathsf{C}, C_{\ell_e+1})) \end{array} \right) \\
- \ \Delta_u := & \begin{cases} \mathsf{Open}(\mathsf{com}, \mathsf{decom}) & \text{if } \left(\begin{array}{c} \mathsf{OTS.Ver}(\hbar_e || C_{\ell_e+1} || \Gamma_e || \Gamma_s, \delta_o, \mathsf{vk}) = 1; \| \\ \mathsf{wABS.Ver}(\mathsf{vk}, \delta_w, \Gamma_s) = 1; \| \\ \mathsf{decom} \longleftarrow \mathsf{ABE.Decrypt}(\mathsf{CT}, \mathscr{SK}_B, \Gamma_e), \\ \text{where} \\ \mathsf{U} = (\mathsf{com}, \delta_s = (\delta_w, \delta_o, \mathsf{vk}), \mathsf{CT} = (\mathsf{C}, C_{\ell_e+1})) \end{array} \right) \\ \perp & \text{otherwise.} \end{cases}
\end{aligned}
$$

Figure 4: Attribute-based signcryption: construction 2

8 Security of Construction 2

8.1 Perfect Privacy

Theorem 8.1. *The proposed SCP-ABSC scheme in Section 7 is perfectly private.*

Proof. It is similar to Theorem 6.1. □

8.2 Adaptive-Predicates IND-CCA security

Theorem 8.2. *If DSG1, DSG2 and DSG3 assumptions hold in $\mathscr{J}$, H_e is a collision resistant hash function, $\mathscr{C}$ has hiding property and* OTS *is a strongly unforgeable one-time signature scheme, then our proposed SCP-ABSC scheme in Section 7 is IND-CCA secure in adaptive-predicates model (Definition 2.10).*

Proof. The proof can be obtained by the similar approach as in proof of Theorem 6.2 and the argument used for proving CCA security in [11]. □

8.3 Adaptive-Predicates Strong Unforgeability

The strong unforgeability model used here is similar to what is stated in Definition 2.14 except the adversary is not allowed to access the unsigncrypt oracle.

Theorem 8.3. *If DSG1, DSG2 and DSG3 assumptions hold in $\mathscr{J}$,* OTS *is a strong OTS scheme and H_s, H_e are collision resistant hash functions, then the proposed basic SCP-ABSC scheme in Section 7 is strongly existential unforgeable in adaptive-predicates model.*

Proof. It is straightforward from Theorem 8.4 and Theorem 4.2. □

Theorem 8.4. *If the primitive attribute-based signature scheme* wABS *is adaptive-predicate weakly existential unforgeable (Definition 2.6),* OTS *is a strong OTS scheme and H_e is a collision resistant hash function, then the proposed basic SCP-ABSC scheme in section 7 is strongly existential unforgeable in adaptive-predicates model.*

Proof. Let $\mathscr{A}$ be an adversary that breaks the adaptive-predicates strong existential unforgeability of the proposed SCP-ABSC scheme with non-negligible advantage ε. Suppose $\mathscr{A}$ has made ν number of signcrypt oracle queries. Let $\mathsf{U}_i = (\mathsf{com}_i, \delta_s^{(i)}, \mathsf{CT}_i)$, where $\delta_s^{(i)} = (\delta_w^{(i)}, \delta_o^{(i)}, \mathsf{vk}^{(i)})$ be the replied signcryption to the i^{th} query message $(m^{(i)}, A_i, \Gamma_s^{(i)}, \Gamma_e^{(i)})$ for $i \in [\nu]$. Let $\mathsf{U}^* = (\mathsf{com}^*, \delta_s^*, \mathsf{CT}^*)$ be the forgery by $\mathscr{A}$ on the message (m, Γ_s, Γ_e), where $\delta_s^* = (\delta_w^*, \delta_o^*, \mathsf{vk}^*)$. We define an event as

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

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: Forged**.** We establish a PPT algorithm $\mathscr{B}_{\mathsf{wABS}}$ for forging to the primitive ABS scheme wABS with advantage at least $\varepsilon/2$.

Setup: First, the challenger $\mathscr{CH}$ publishes the public parameters $\mathscr{ABSP}.\mathscr{PP}$ of wABS. Then, the simulator $\mathscr{B}_{\mathsf{wABS}}$ runs $\mathsf{CSetup}(1^\kappa)$ to produce $\mathscr{CK}$. $\mathscr{B}_{\mathsf{wABS}}$ chooses $a_e, b_e \xleftarrow{\text{U}} \mathbb{Z}_N$ and sets $u_e := g^{a_e}, v_e := g^{b_e}$. $\mathscr{B}_{\mathsf{wABS}}$ selects a hash function $H_e : \{0,1\}^* \longrightarrow \mathbb{Z}_N$. It provides $\mathscr{PP} := (\mathscr{J}, g, g^a, u_s, u_e, v_s, v_e, g_T^\alpha, \{T_i\}_{i\in\mathscr{U}}, Z_3, H_s, H_e, \mathscr{CK})$ to $\mathscr{A}$.

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

- KeyGen: Since the ABSC scheme and the primitive ABS scheme have the identical key distribution, the key queries from $\mathscr{A}$ will be forwarded to the challenger $\mathscr{CH}$. Similarly, the answers (keys) will be reversed back to $\mathscr{A}$.
- Signcrypt: Let $(m^{(i)}, A_i, \Gamma_s^{(i)}, \Gamma_e^{(i)})$ be the i^{th} signcrypt query to $\mathscr{B}_{\mathsf{wABS}}$ by $\mathscr{A}$. $\mathscr{B}_{\mathsf{wABS}}$ executes $(\mathsf{com}_i, \mathsf{decom}_i) \longleftarrow \mathsf{Commit}(m^{(i)})$ and $(\mathsf{vk}^{(i)}, \mathsf{signk}^{(i)}) \longleftarrow \mathsf{OTS.Gen}(1^\kappa)$. Then, $\mathscr{B}_{\mathsf{wABS}}$ makes a signature query for $(\mathsf{vk}^{(i)}, A_i, \Gamma_s^{(i)})$ to $\mathscr{CH}$ and gets the replied signature $\delta_w^{(i)}$ from $\mathscr{CH}$. Then, $\mathscr{B}_{\mathsf{wABS}}$ runs $\mathsf{C}_i \longleftarrow \mathsf{ABE.Encrypt}(\mathscr{PP}, \mathsf{decom}_i, \Gamma_e^{(i)})$. Computes $\hbar_e^{(i)} := H_e(\mathsf{com}_i, \mathsf{C}_i, \delta_w^{(i)})$ and $C_{\ell_e^{(i)}+1} \longleftarrow \mathsf{fun}(\mathscr{PP}, \hbar_e^{(i)}, s_e^{(i)})$. Then $\mathscr{B}_{\mathsf{wABS}}$ executes $\delta_o^{(i)} \longleftarrow \mathsf{OTS.Sign}(\ \hbar_e^{(i)} || C_{\ell_e^{(i)}+1} || \Gamma_e^{(i)} || \Gamma_s^{(i)}, \mathsf{signk}^{(i)})$ and sets $\mathsf{U}_i := (\mathsf{com}, \delta_s^{(i)}, \mathsf{CT}_i)$, where $\delta_s^{(i)} := (\delta_w^{(i)}, \delta_o^{(i)}, \mathsf{vk}^{(i)})$ and $\mathsf{CT}_i := (\mathsf{C}_i, C_{\ell_e^{(i)}+1})$. It returns the i^{th} signcryption U_i to $\mathscr{A}$.

Forgery: $\mathscr{A}$ outputs a tuple $(\mathsf{U}^*, \Gamma_s^*, \Gamma_e^*)$. Then, $\mathscr{B}_{\mathsf{wABS}}$ forges the signature δ_w^* for $(\mathsf{vk}^*, \Gamma_s^*)$ to the primitive ABS scheme wABS.

Analysis: By the event Forged, we have $\mathsf{vk}^* \neq \mathsf{vk}^{(i)}$ for $i \in [\nu]$. Therefore, $(\mathsf{vk}^*, \Gamma_s^*)$ has not been queried for signature to $\mathscr{CH}$.

Case: ¬Forged**.** In this case, we will develop an algorithm $\mathscr{B}_{\mathsf{OTS}}$ for forging to the primitive strongly unforgeable one-time signature scheme OTS with advantage at least $\varepsilon/2\nu$. Let $\mathscr{CH}$ be the challenger for the primitive one-time signature scheme OTS. The challenger $\mathscr{CH}$ runs $(\mathsf{vk}^*, \mathsf{signk}^*) \longleftarrow \mathsf{OTS.Gen}(1^\kappa)$ and gives vk^* to $\mathscr{B}_{\mathsf{OTS}}$. The simulator $\mathscr{B}_{\mathsf{OTS}}$ picks $j \xleftarrow{\text{U}} [\nu]$ as a guess such that $\mathsf{vk}^* = \mathsf{vk}^{(j)}$.

Setup: Similar to that of Section 5.

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

– KeyGen: $\mathscr{B}_{\mathsf{OTS}}$ can handle the key queries as it knows the $\mathscr{MSK}$.

– Signcrypt: Let $(m^{(i)}, A_i, \Gamma_s^{(i)}, \Gamma_e^{(i)})$ be the i^{th} signcrypt query to $\mathscr{B}_{\mathsf{OTS}}$ by $\mathscr{A}$.

▷ $(i \neq j)$ **:**

$\mathscr{B}_{\mathsf{OTS}}$ runs $\mathsf{Commit}(m^{(i)})$ and $\mathsf{OTS.Gen}(1^\kappa)$ to obtain $(\mathsf{com}_i, \mathsf{decom}_i)$ and $(\mathsf{vk}^{(i)}, \mathsf{signk}^{(i)})$ respectively. Then, runs $\delta_w^{(i)} \longleftarrow \mathsf{wABS.Sign}(\mathscr{PP}, \mathsf{vk}^{(i)}, \mathscr{SK}_A, \Gamma_s^{(i)})$, $\mathsf{C}_i \longleftarrow \mathsf{ABE.Encrypt}(\mathscr{PP}, \mathsf{decom}_i, \Gamma_e^{(i)})$, $C_{\ell_e^{(i)}+1} \longleftarrow \mathsf{fun}(\mathscr{PP}, \hbar_e^{(i)}, s_e^{(i)})$, $\delta_o^{(i)} \longleftarrow \mathsf{OTS.Sign}(\hbar_e^{(i)} || C_{\ell_e^{(i)}+1} || \Gamma_e^{(i)} || \Gamma_s^{(i)}, \mathsf{signk}^{(i)})$. It sets $\delta_s^{(i)} := (\delta_w^{(i)}, \delta_o^{(i)}, \mathsf{vk}^{(i)})$, $\mathsf{CT}_i := (\mathsf{C}_i, C_{\ell_e^{(i)}+1})$ and returns the i^{th} signcryption $\mathsf{U}_i := (\mathsf{com}_i, \delta_s^{(i)}, \mathsf{CT}_i)$ to $\mathscr{A}$.

▷ $(i = j)$ **:**

Same as above except $\mathscr{B}_{\mathsf{OTS}}$ does not run $\mathsf{OTS.Gen}(1^\kappa)$, but it sets $\mathsf{vk}^{(i)} := \mathsf{vk}^*$ and it makes a query to the challenger $\mathscr{CH}$ on the message $\hbar_e^{(i)} || C_{\ell_e^{(i)}+1} || \Gamma_e^{(i)} || \Gamma_s^{(i)}$ and gets the replied signature $\delta_o^{(i)}$.

Forgery: $\mathscr{A}$ outputs a tuple $(\mathsf{U}^*, \Gamma_s^*, \Gamma_e^*)$. Then, $\mathscr{B}_{\mathsf{OTS}}$ forges the signature δ_o^* for $\hbar_e^* || C_{\ell_e^*+1}^* || \Gamma_e^* || \Gamma_s^*$ to the primitive one-time signature scheme OTS. We note that $\mathscr{B}_{\mathsf{OTS}}$ knows the $\mathscr{MSK}$, so it can compute $\hbar_e^*$ from U^*.

Analysis: With probability $1/\nu$, $\mathscr{B}_{\mathsf{OTS}}$ correctly guesses $j \in [\nu]$ such that this case is happened. Now we only have to show that

$$(\hbar_e^* || C_{\ell_e^*+1}^* || \Gamma_e^* || \Gamma_s^*, \delta_o^*) \neq (\hbar_e^{(j)} || C_{\ell_e^{(j)}+1} || \Gamma_e^{(j)} || \Gamma_s^{(j)}, \delta_o^{(j)}).$$

Indeed, if

$$(\hbar_e^* || C_{\ell_e^*+1}^* || \Gamma_e^* || \Gamma_s^*, \delta_o^*) = (\hbar_e^{(j)} || C_{\ell_e^{(j)}+1} || \Gamma_e^{(j)} || \Gamma_s^{(j)}, \delta_o^{(j)})$$

we have $\hbar_e^* = \hbar_e^{(j)}, C_{\ell_e+1}^* = C_{\ell_e^{(j)}+1}, \delta_o^* = \delta_o^{(j)}$. Since H_e is collision resistant, we have $\mathsf{com}^* = \mathsf{com}_j, \delta_w^* = \delta_w^{(j)}$, $\mathsf{C}^* = \mathsf{C}_j$ and which implies that $\mathsf{decom}^* = \mathsf{decom}_j$. Then, using $\mathsf{com}^* = \mathsf{com}_j$, we have $m^* = m^{(j)}$. Altogether, we have $(\mathsf{U}^*, m^*, \Gamma_s^*, \Gamma_e^*) = (\mathsf{U}_j, m^{(j)}, \Gamma_s^{(j)}, \Gamma_e^{(j)})$, which is contradiction to the definition of strong existential unforgeability of SCP-ABSC scheme.

□

9 Strong Unforgeability of Construction 2 in Presence of Unsigncrypt Oracle

The strong unforgeability of the construction 2 in Section 7 is proven without giving the unsigncrypt oracle access to $\mathscr{A}$. The main reason for not providing the unsigncrypt oracle access is that the proof of 8.4 (in Section 8.3) uses the black-box access to the ABS. In this section, we provide the proof of strong unforgeability, where $\mathscr{A}$ is provided the access to unsigncrypt oracle.

The proof style is just an extension of the unforgeability proof of ABS (Section 4.1) and the different oracle queries are handled in similar manner as in confidentiality proof of ABSC (Section 6.2). We consider two forms of semi-functional keys (resp. uq-keys), viz., sf-type 1 and sf-type 2. We also consider two forms of signcryptions, namely sf-type I and sf-type II. Description of all these semi-functional objects are similar to Section 6.2, so we skip them here. We note that Ver and Decrypt algorithms are run using the verification text and an uq-key respectively in the Unsigncrypt algorithm.

We define a new object, called verification text key (in short vTextKey) which consists of vText and uq-key. We consider four forms of semi-functional vTextKeys which are defined through the different forms of the vTexts and uq-keys. We already have defined two forms of semi-functional uq-keys in Section 6.2, so we skip them as well. Now, we define the two forms of vText (as defined in Section 4.1) as follows.

Semi-functional type 1 vText. Pick $c, \iota \xleftarrow{U} \mathbb{Z}_N$, $\mathbf{v}_s \xleftarrow{U} \mathbb{Z}_N^{n_s}$. For each $i \in [\ell_s]$, pick $\gamma_s^{(i)} \xleftarrow{U} \mathbb{Z}_N$. For each $i \in \mathscr{U}$, choose $z_i \xleftarrow{U} \mathbb{Z}_N$. The sf-type 1 vText is obtained by modifying normal vText $\mathscr{V} = (\mathbf{V}_0, \{\mathbf{V}_i\}_{i\in[\ell_s]})$ as given below:

$\mathbf{V}_0 := \big(g^s \boxed{g_2^c}, (u_s^{\hbar_s} v_s)^s \boxed{g_2^\iota}, g_T^{\alpha s} \big)$,

$\mathbf{V}_i := \big(g^{a\lambda_s^{(i)}} T_{\rho_s(i)}^{-r_s^{(i)}} \boxed{g_2^{\mathbf{M}_s^{(i)}.\mathbf{v}_s + \gamma_s^{(i)} z_{\rho_s(i)}}}, \quad g^{r_s^{(i)}} \boxed{g_2^{-\gamma_s^{(i)}}} \big)$, for $i \in [\ell_s]$.

Semi-functional type 2 vText. This is same as sf-type 1 vText except the following:

$\mathbf{V}_0 := \big(g^s g_2^c, (u_s^{\hbar_s} v_s)^s g_2^\iota, \boxed{\hat{g}_t} \big)$, where $\hat{g}_t \xleftarrow{U} \mathbb{G}_T$.

Below we define semi-functional forms of vTextKey which depend on the forms of the underlying uq-key and vText.

– **Normal vTextKey.** The uq-key and vText are of normal form.

– **sf-type 1 vTextKey.** The uq-key is normal and vText is sf-type 1.

– **sf-type 2 vTextKey.** The uq-key is sf-type 1 and vText is sf-type 1.

– **sf-type 3 vTextKey.** The uq-key is sf-type 2 and vText is sf-type 1.

– **sf-type 4 vTextKey.** The uq-key and vText are of sf-type 2.

Theorem 9.1. *If DSG1, DSG2 and DSG3 assumptions hold in $\mathscr{J}$, OTS is a strong OTS scheme and H_s, H_e are collision resistant hash functions, then the proposed basic SCP-ABSC scheme in Section 7 is strongly existential unforgeable in adaptive-predicates model (Definition 2.14).*

Proof. Suppose an adversary $\mathscr{A}$ can break the adaptive-predicates strong unforgeability of the proposed SCP-ABSC scheme with non-negligible advantage ε. We assume that $\mathscr{A}$ has made ν_3 number of signcrypt oracle queries. Let $(m^{(i)}, A_i, \Gamma_s^{(i)}, \Gamma_e^{(i)})$ be the i^{th} signcrypt query and $\mathsf{U}_i = (\mathsf{com}_i, \delta_s^{(i)} = (\delta_w^{(i)}, \delta_o^{(i)}, \mathsf{vk}^{(i)}), \mathsf{CT}_i)$ be the corresponding replied signcryption. Let $\mathsf{U}^* = (\mathsf{com}^*, \delta_s^* = (\delta_w^*, \delta_o^*, \mathsf{vk}^*), \mathsf{CT}^*)$ be the forgery by $\mathscr{A}$ on the message $(m^*, \Gamma_s^*, \Gamma_e^*)$. We define an event as

$$\mathsf{Forged} := \mathsf{vk}^* \notin \{\mathsf{vk}^{(i)} \,\big|\, i \in [\nu_3]\}.$$

Then, we have

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

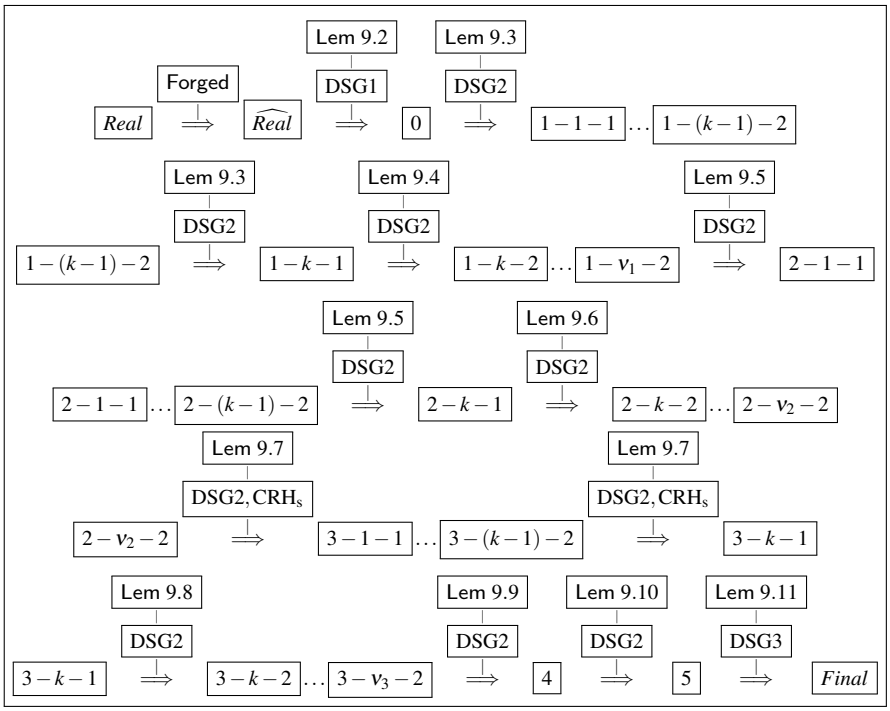
$$\implies \Pr[\mathscr{A} \text{ Succeeds} \wedge \mathsf{Forged}] \geq \varepsilon/2 \text{ or } \Pr[\mathscr{A} \text{ Succeeds} \wedge \neg\mathsf{Forged}] \geq \varepsilon/2$$

Case: $\neg$Forged**.** This case is similar to that of Theorem 8.4. We note that in this case the simulator can handle the unsigncrypt queries as it knows the $\mathscr{MSK}$.

Case: Forged**.** Suppose there are at most ν_1 key queries and ν_2 unsigncrypt queries, then the security proof consists of hybrid argument over a sequence of $2(\nu_1+\nu_2+\nu_3)+6$ games. The games are defined below:

- Game_{Real} := Original APs-sUF-CMA game of ABSC scheme.
- $\text{Game}_{\widehat{Real}}$:= This is same as game Game_{Real} except the event Forged always happens.
- Game_0 (:= Game_{1-0-2}) is just like $\text{Game}_{\widehat{Real}}$ except that the vTextKey for verifying the forgery is of sf-type 1
- Game_{1-k-1} (for $1 \leq k \leq \nu_1$) is same as $\text{Game}_{1-(k-1)-2}$ except the k^{th} key is sf-type 1.
- Game_{1-k-2} (for $1 \leq k \leq \nu_1$) is same as Game_{1-k-1} except k^{th} key is sf-type 2.
- In Game_{2-k-1} (for $1 \leq k \leq \nu_2$) is same as $\text{Game}_{2-(k-1)-2}$ except the k^{th} unsigncrypt query is answered by sf-type 1 uq-key. (So, in this sequel we define $\text{Game}_{2-0-2} := \text{Game}_{1-\nu_1-2}$)
- Game_{2-k-2} (for $1 \leq k \leq \nu_2$) is same as Game_{2-k-1} except the k^{th} unsigncrypt query is answered by sf-type 2 uq-key.
- In Game_{3-k-1} (for $1 \leq k \leq \nu_3$) is same as $\text{Game}_{3-(k-1)-2}$ except the replied signcryption to the k^{th} signcrypt oracle query is sf-type I. (So, in this sequel we define $\text{Game}_{3-0-2} := \text{Game}_{2-\nu_2-2}$)
- Game_{3-k-2} (for $1 \leq k \leq \nu_3$) is same as Game_{3-k-1} except the replied signcryption to the k^{th} signcrypt oracle query sf-type II.
- Game_4 is similar to $\text{Game}_{3-\nu_3-2}$ except that the vTextKey for verifying the forgery is of sf-type 2.
- Game_5 is similar to Game_4 except that the vTextKey for verifying the forgery is of sf-type 3.
- Game_{Final} is similar to Game_5 except that the vTextKey for verifying the forgery is of sf-type 4.

In Game_{Final}, the component V_{03} 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 Section 9.1), we have the following reduction:

$$\begin{aligned}\mathsf{Adv}^{\mathrm{APs-sUF-CMA}}_{\mathscr{A},\mathrm{ABSC}}(\kappa) \leq \nu_2\mathsf{Adv}^{\mathrm{sUF-CMA}}_{\mathscr{B}_1,\mathrm{OTS}}(\kappa) + \nu_2\mathsf{Adv}^{\mathrm{CRH_e}}_{\mathscr{B}_2}(\kappa) + \mathsf{Adv}^{\mathrm{DSG1}}_{\mathscr{B}_3}(\kappa) + \\ (2\nu_1+2\nu_2+2\nu_3+2)\mathsf{Adv}^{\mathrm{DSG2}}_{\mathscr{B}_4}(\kappa) + \\ \nu_3\mathsf{Adv}^{\mathrm{CRH_s}}_{\mathscr{B}_5}(\kappa) + \mathsf{Adv}^{\mathrm{DSG3}}_{\mathscr{B}_6}(\kappa)\end{aligned}$$

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.

□

9.1 Lemmas Used in the Proof of Theorem 9.1

The following lemmas can be proven similarly as that of SP-ABS and SCP-ABSC.

Lemma 9.2. $\mathrm{Game}_{\widehat{Real}}$ *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* $|\mathsf{Adv}^{\widehat{\mathrm{Real}}}_{\mathscr{A},\mathrm{ABSC}}(\kappa) - \mathsf{Adv}^{0}_{\mathscr{A},\mathrm{ABSC}}(\kappa)| \leq \mathsf{Adv}^{\mathrm{DSG1}}_{\mathscr{B}}(\kappa)$.

Lemma 9.3. $\mathrm{Game}_{1-(k-1)-2}$ *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* $|\mathsf{Adv}^{1-(k-1)-2}_{\mathscr{A},\mathrm{ABSC}}(\kappa) - \mathsf{Adv}^{1-k-1}_{\mathscr{A},\mathrm{ABSC}}(\kappa)| \leq \mathsf{Adv}^{\mathrm{DSG2}}_{\mathscr{B}}(\kappa)$ *for* $1 \leq k \leq \nu_1$.

Lemma 9.4. Game_{1-k-1} *and* Game_{1-k-2} *are indistinguishable under the DSG2 assumption. That is, for every adversary* $\mathscr{A}$ *there exists a PPT algorithm* $\mathscr{B}$ *such that* $|\mathsf{Adv}^{1-k-1}_{\mathscr{A},\mathrm{ABSC}}(\kappa) - \mathsf{Adv}^{1-k-2}_{\mathscr{A},\mathrm{ABSC}}(\kappa)| \leq \mathsf{Adv}^{\mathrm{DSG2}}_{\mathscr{B}}(\kappa)$ *for* $1 \leq k \leq \nu_1$.

Lemma 9.5. $\text{Game}_{2-(k-1)-2}$ *and* Game_{2-k-1} *are indistinguishable under DSG2 assumption and collision resistant property of* H_e*. That is, for every adversary* $\mathscr{A}$ *there exists a PPT algorithm* $\mathscr{B}$ *such that* $|\mathsf{Adv}^{2-(k-1)-2}_{\mathscr{A},\text{ABSC}}(\kappa) - \mathsf{Adv}^{2-k-1}_{\mathscr{A},\text{ABSC}}(\kappa)| \leq \mathsf{Adv}^{\text{DSG2}}_{\mathscr{B}}(\kappa)$ *for* $1 \leq k \leq \nu_2$.

Lemma 9.6. 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* $|\mathsf{Adv}^{2-k-1}_{\mathscr{A},\text{ABSC}}(\kappa) - \mathsf{Adv}^{2-k-2}_{\mathscr{A},\text{ABSC}}(\kappa)| \leq \mathsf{Adv}^{\text{DSG2}}_{\mathscr{B}}(\kappa)$ *for* $1 \leq k \leq \nu_2$.

Lemma 9.7. $\text{Game}_{3-(k-1)-2}$ *and* Game_{3-k-1} *are indistinguishable under DSG2 assumption and collision resistant property of* H_s*. That is, for every adversary* $\mathscr{A}$ *there exists PPT algorithm* $\mathscr{B}$ *such that*

$$|\mathsf{Adv}^{3-(k-1)-2}_{\mathscr{A},\text{ABSC}}(\kappa) - \mathsf{Adv}^{3-k-1}_{\mathscr{A},\text{ABSC}}(\kappa)| \leq \mathsf{Adv}^{\text{DSG2}}_{\mathscr{B}}(\kappa) + \mathsf{Adv}^{\text{CRH}_s}_{\mathscr{B}}(\kappa) \text{ for } 1 \leq k \leq \nu_3.$$

Lemma 9.8. 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* $|\mathsf{Adv}^{3-k-1}_{\mathscr{A},\text{ABSC}}(\kappa) - \mathsf{Adv}^{3-k-2}_{\mathscr{A},\text{ABSC}}(\kappa)| \leq \mathsf{Adv}^{\text{DSG2}}_{\mathscr{B}}(\kappa)$ *for* $1 \leq k \leq \nu_3$.

Lemma 9.9. $\text{Game}_{3-\nu_3-2}$ *and* Game_4 *are indistinguishable under the DSG2 assumption. That is, for every adversary* $\mathscr{A}$ *there exists a PPT algorithm* $\mathscr{B}$ *such that* $|\mathsf{Adv}^{3-\nu_3-2}_{\mathscr{A},\text{ABSC}}(\kappa) - \mathsf{Adv}^{4}_{\mathscr{A},\text{ABSC}}(\kappa)| \leq \mathsf{Adv}^{\text{DSG2}}_{\mathscr{B}}(\kappa)$

Lemma 9.10. Game_4 *and* Game_5 *are indistinguishable under the DSG2 assumption. That is, for every adversary* $\mathscr{A}$ *there exists a PPT algorithm* $\mathscr{B}$ *such that* $|\mathsf{Adv}^{4}_{\mathscr{A},\text{ABSC}}(\kappa) - \mathsf{Adv}^{5}_{\mathscr{A},\text{ABSC}}(\kappa)| \leq \mathsf{Adv}^{\text{DSG2}}_{\mathscr{B}}(\kappa)$

Lemma 9.11. 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* $|\mathsf{Adv}^{4}_{\mathscr{A},\text{ABSC}}(\kappa) - \mathsf{Adv}^{\text{Final}}_{\mathscr{A},\text{ABSC}}(\kappa)| \leq \mathsf{Adv}^{\text{DSG3}}_{\mathscr{B}}(\kappa)$

10 Mechanism for Complete Construction

Although the technique is available in [25] but for self-containment, in this section we briefly demonstrate it. The mechanism described here is for both SP-ABS and SCP-ABSC supporting MSPs. For complete construction, the row labeling functions of span programs are not assumed to be injective. If we allow an attribute to repeat in the span programs at most φ time and the size of the universe $\mathscr{U}$ is n, then the size of new universe $\mathscr{U}'$ for the complete construction will be $n\varphi$. In complete construction, for each attribute $\chi \in \mathscr{U}$, we consider φ copies of χ in $\mathscr{U}'$. To enumerate each copy, we assign a label say j to the attribute say χ, i.e., $\mathscr{U}' := \{(\chi, j) | \chi \in \mathscr{U}, j \in [\varphi]\}$. Similarly, for any access policy $\Gamma := (\mathbf{M}, \rho)$ if $\rho(i) = \chi$ and the attribute χ appears j^{th} time, then we label the i^{th} row by (χ, j), i.e., we have a new row labeling function ρ' defined by $\rho'(i) := (\chi, j)$. Likewise if A is a set of attributes corresponding to $\mathscr{U}$, then $A' := \{(\chi, j) | \chi \in A, j \in [\varphi]\}$ is the set of attributes for $\mathscr{U}'$. Then, we have that the set of attributes A satisfies the policy $(\mathbf{M}, \rho)$ if and only if A' satisfies $(\mathbf{M}, \rho')$. Due to this technique, the sizes of public parameters and key increase by a factor linear to φ, but the size of signature (resp. signcryption) and the cost of sign and ver (resp. signcrypt and unsigncrypt) for SP-ABS (resp. SCP-ABSC) remain unchanged.

11 Conclusion

We have presented an ABSC scheme in $\mathscr{C}t\mathscr{E}\&\mathscr{S}t\mathscr{S}$ paradigm using our proposed AP-UF-CMA secure SP-ABS and AP-IND-CPA secure CP-ABE of [25]. Since the algorithms of ABS and ABE run in

parallel in Signcrypt and Unsigncrypt, the execution of the scheme is comparatively faster. To best of our knowledge, this is the first ABSC scheme whose sUF-CMA security and IND-CCA security have been proven in adaptive-predicates models. The scheme also has other features, signer-privacy and non-repudiation, and supports combined-setup.

References

[1] J. H. An, Y. Dodis, and T. Rabin. On the security of joint signature and encryption. In *Advances in Cryptology — EUROCRYPT 2002, Proc. of the 2002 International Conference on the Theory and Applications of Cryptographic Techniques, Amsterdam, The Netherlands*, volume 2332 of *Lecture Notes in Computer Science*, pages 83–107. Springer Berlin Heidelberg, April 2002.

[2] N. Attrapadung, B. Libert, and E. de Panafieu. Expressive key-policy attribute-based encryption with constant-size ciphertexts. In *Public Key Cryptography – PKC 2011, Proc. of the 14th International Conference on Practice and Theory in Public Key Cryptography, Taormina, Italy*, volume 6571 of *Lecture Notes in Computer Science*, pages 90–108. Springer Berlin Heidelberg, March 2011.

[3] J. Baek, R. Steinfeld, and Y. Zheng. Formal proofs for the security of signcryption. In *Proc. of the 5th International Workshop on Practice and Theory in Public Key Cryptosystems (PKC'02), Paris, France*, volume 2274 of *Lecture Notes in Computer Science*, pages 80–98. Springer Berlin Heidelberg, February 2002.

[4] A. Beimel. Secure schemes for secret sharing and key distribution, 1996. `http://www.shoup.net/papers/` [Online; Accessed on August 5, 2016].

[5] J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attribute-based encryption. In *Proc. of the 2007 IEEE Symposium on Security and Privacy (S&P'07), CA, USA*, pages 321–334. IEEE, May 2007.

[6] D. Boneh and X. Boyen. Efficient selective-ID secure identity-based encryption without random oracles. *Journal of Cryptology*, 24(4):659–693, October 2011.

[7] D. Boneh and M. Franklin. Identity-based encryption from the weil pairing. In *Advances in Cryptology — CRYPTO 2001, Proc. of the 21st Annual International Cryptology Conference, Santa Barbara, CA, USA*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer Berlin Heidelberg, August 2001.

[8] D. Boneh, E.-J. Goh, and K. Nissim. Evaluating 2-dnf formulas on ciphertexts. In *Proc. of the 2nd Theory of Cryptography Conference (TCC'05), Cambridge, MA, USA*, volume 3378 of *Lecture Notes in Computer Science*, pages 325–341. Springer Berlin Heidelberg, February 2005.

[9] D. Boneh and J. Katz. Improved efficiency for CCA-secure cryptosystems built using identity-based encryption. In *Topics in Cryptology – CT-RSA 2005, Proc. of the The Cryptographers' Track at the RSA Conference 2005, San Francisco, CA, USA*, volume 3376 of *Lecture Notes in Computer Science*, pages 87–103. Springer Berlin Heidelberg, February 2005.

[10] X. Boyen. Multipurpose identity-based signcryption. In *Advances in Cryptology - CRYPTO 2003, Proc. of the 23rd Annual International Cryptology Conference, Santa Barbara, CA, USA*, volume 2729 of *Lecture Notes in Computer Science*, pages 383–399. Springer Berlin Heidelberg, August 2003.

[11] R. Canetti, S. Halevi, and J. Katz. Chosen-ciphertext security from identity-based encryption. In *Advances in Cryptology - EUROCRYPT 2004, Proc. of the 2004 International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland*, volume 3027 of *Lecture Notes in Computer Science*, pages 207–222. Springer Berlin Heidelberg, May 2004.

[12] C. Chen, J. Chen, H. W. Lim, Z. Zhang, and D. Feng. Combined public-key schemes: The case of ABE and ABS. In *Proc. of the 6th International Conference on Provable Security (ProvSec'12), Chengdu, China*, volume 7496 of *Lecture Notes in Computer Science*, pages 53–69. Springer Berlin Heidelberg, September 2012.

[13] L. Chen and J. Malone-Lee. Improved identity-based signcryption. In *Proc. of the 8th International Workshop on Theory and Practice in Public Key Cryptography (PKC'05), Les Diablerets, Switzerland*, volume 3386 of *Lecture Notes in Computer Science*, pages 362–379. Springer Berlin Heidelberg, January 2005.

[14] I. Damgård and E. Fujisaki. A statistically-hiding integer commitment scheme based on groups with hidden order. In *Advances in Cryptology — ASIACRYPT 2002, Proc. of the 8th International Conference on the*

Theory and Application of Cryptology and Information Security Queenstown, New Zealand, volume 2501 of *Lecture Notes in Computer Science*, pages 125–142. Springer Berlin Heidelberg, December 2002.

[15] A. W. Dent, M. Fischlin, M. Manulis, and D. Schröder. Confidential signatures and deterministic signcryption. In *Proc. of the 13th International Conference on Practice and Theory in Public Key Cryptography (PKC'10), Paris, France*, volume 6056 of *Lecture Notes in Computer Science*, pages 462–479. Springer Berlin Heidelberg, May 2010.

[16] K. Emura, A. Miyaji, and M. S. Rahman. Dynamic attribute-based signcryption without random oracles. *International Journal of Applied Cryptography*, 2(11):199–211, 2012.

[17] M. Gagné, S. Narayan, and R. Safavi-Naini. Threshold attribute-based signcryption. In *Proc. of the 7th International Conference on Security and Cryptography for Networks (SCN'10), Amalfi, Italy*, volume 6280 of *Lecture Notes in Computer Science*, pages 154–171. Springer Berlin Heidelberg, September 2010.

[18] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proc. of the 13th ACM Conference on Computer and Communications Security (CCS'06), Alexandria, VA, USA*, pages 89–98. ACM, October-November 2006.

[19] S. Haber and B. Pinkas. Securely combining public-key cryptosystems. In *Proc. of the 2001 ACM Conference on Computer and Communications Security (CCS'01), Philadelphia, PA, USA*, pages 215–224. ACM, November 2001.

[20] S. Halevi and S. Micali. Practical and provably-secure commitment schemes from collision-free hashing. In *Advances in Cryptology — CRYPTO 1996, Proc. of the 16th Annual International Cryptology Conference Santa Barbara, CA, USA*, volume 1109 of *Lecture Notes in Computer Science*, pages 201–215. Springer Berlin Heidelberg, August 1996.

[21] F. Hess. Efficient identity based signature schemes based on pairings. In *Revised Papers, the 9th Annual International Workshop on Selected Areas in Cryptography (SAC'02), St. John's, Newfoundland, Canada*, volume 2595 of *Lecture Notes in Computer Science*, pages 310–324. Springer Berlin Heidelberg, August 2002.

[22] Q. Huang, D. S. Wong, and Y. Zhao. Generic transformation to strongly unforgeable signatures. In *Proc. of the 5th International Conference on Applied Cryptography and Network Security (ACNS'07), Zhuhai, China*, volume 4521 of *Lecture Notes in Computer Science*, pages 1–17. Springer Berlin Heidelberg, June 2007.

[23] A. Lewko and B. Waters. New techniques for dual system encryption and fully secure HIBE with short ciphertexts. In *Theory of Cryptography, Proc. of the 7th Theory of Cryptography Conference, Zurich, Switzerland*, volume 5978 of *Lecture Notes in Computer Science*, pages 455–479. Springer Berlin Heidelberg, February 2010.

[24] A. Lewko and B. Waters. New proof methods for attribute-based encryption: Achieving full security through selective techniques. In *Advances in Cryptology – CRYPTO 2012, Proc. of the 32nd Annual Cryptology Conference, Santa Barbara, CA, USA*, volume 7417 of *Lecture Notes in Computer Science*, pages 180–198. Springer Berlin Heidelberg, August 2012.

[25] A. B. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In *Advances in Cryptology – EUROCRYPT 2010, Proc. of the 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera*, volume 6110 of *Lecture Notes in Computer Science*, pages 62–91. Springer Berlin Heidelberg, May 2010.

[26] J. Li, M. H. Au, W. Susilo, D. Xie, and K. Ren. Attribute-based signature and its applications. In *Proc. of the 2010 ACM Conference on Computer and Communications Security (CCS'10), Chicago, IL, USA*, pages 60–69. ACM, October 2010.

[27] B. Libert and J.-J. Quisquater. Efficient signcryption with key privacy from gap diffie-hellman groups. In *Public Key Cryptography – PKC 2004, Proc. of the 7th International Workshop on Theory and Practice in Public Key Cryptography, Singapore*, volume 2947 of *Lecture Notes in Computer Science*, pages 187–200. Springer Berlin Heidelberg, March 2004.

[28] B. Libert and J.-J. Quisquater. Improved signcryption from q-diffie-hellman problems. In *Revised Selected Papers, the 4th International Conference on Security in Communication Networks (SCN'04), Amalfi, Italy*, volume 3352 of *Lecture Notes in Computer Science*, pages 220–234. Springer Berlin Heidelberg, September

2004.

[29] H. Maji, M. Prabhakaran, and M. Rosulek. Attribute-based signatures: Achieving attribute-privacy and collusion-resistance. Cryptology ePrint Archive, Report 2008/328, 2008. `http://eprint.iacr.org/`.

[30] H. Maji, M. Prabhakaran, and M. Rosulek. Attribute-based signatures. Cryptology ePrint Archive, Report 2010/595, 2010. `http://eprint.iacr.org/`.

[31] H. Maji, M. Prabhakaran, and M. Rosulek. Attribute-based signatures. In *Topics in Cryptology – CT-RSA 2011, Proc. of the Cryptographers' Track at the RSA Conference 2011, San Francisco, CA, USA*, volume 6558 of *Lecture Notes in Computer Science*, pages 376–392. Springer Berlin Heidelberg, February 2011.

[32] J. Malone-Lee and W. Mao. Two birds one stone: Signcryption using RSA. In *Topics in Cryptology — CT-RSA 2003, Proc. of the Cryptographers' Track at the RSA Conference 2003, San Francisco, CA, USA*, volume 2612 of *Lecture Notes in Computer Science*, pages 211–226. Springer Berlin Heidelberg, April 2003.

[33] T. Matsuda, K. Matsuura, and J. C. N. Schuldt. Efficient constructions of signcryption schemes and signcryption composability. In *Progress in Cryptology - INDOCRYPT 2009, Proc. of the 10th International Conference on Cryptology in India, New Delhi, India*, volume 5922 of *Lecture Notes in Computer Science*, pages 321–342. Springer Berlin Heidelberg, December 2009.

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

[35] T. Okamoto and K. Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In *Advances in Cryptology – CRYPTO 2010, Proc. of the 30th Annual Cryptology Conference, Santa Barbara, CA, USA*, volume 6223 of *Lecture Notes in Computer Science*, pages 191–208. Springer Berlin Heidelberg, August 2010.

[36] T. Okamoto and K. Takashima. Efficient attribute-based signatures for non-monotone predicates in the standard model. In *Public Key Cryptography – PKC 2011, Proc. of the 14th International Conference on Practice and Theory in Public Key Cryptography, Taormina, Italy*, volume 6571 of *Lecture Notes in Computer Science*, pages 35–52. Springer Berlin Heidelberg, March 2011.

[37] T. Okamoto and K. Takashima. Fully secure unbounded inner-product and attribute-based encryption. In *Advances in Cryptology – ASIACRYPT 2012, Proc. of the 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China*, volume 7658 of *Lecture Notes in Computer Science*, pages 349–366. Springer Berlin Heidelberg, December 2012.

[38] R. Ostrovsky, A. Sahai, and B. Waters. Attribute-based encryption with non-monotonic access structures. In *Proc. of the the 14th ACM conference on Computer and Communications Security (CCS'07), Alexandria, VA, USA*, pages 195–203. ACM, October-November 2007.

[39] T. Pandit, S. K. Pandey, and R. Barua. Attribute-based signcryption: Signer privacy, strong unforgeability and IND-CCA2 security in adaptive-predicates attack. In *Proc. of the 8th International Conference on Provable Security (ProvSec'14), Hong Kong, China*, volume 8782 of *Lecture Notes in Computer Science*, pages 274–290. Springer Berlin Heidelberg, October 2014.

[40] K. G. Paterson, J. C. N. Schuldt, M. Stam, and S. Thomson. On the joint security of encryption and signature, revisited. In *Advances in Cryptology – ASIACRYPT 2011, Proc. of the 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea*, volume 7073 of *Lecture Notes in Computer Science*, pages 161–178. Springer Berlin Heidelberg, December 2011.

[41] T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology — CRYPTO 1991, Proc. of the 11th Annual International Cryptology Conference, Santa Barbara, California, USA*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140. Springer Berlin Heidelberg, August 1991.

[42] Y. S. Rao and R. Dutta. Expressive bandwidth-efficient attribute based signature and signcryption in standard model. In *Proc. of the 19th Australasian Conference on Information Security and Privacy (ACISP'14), Wollongong, NSW, Australia*, volume 8544 of *Lecture Notes in Computer Science*, pages 209–225. Springer Berlin Heidelberg, December 2014.

[43] A. Sahai and B. Waters. Fuzzy identity-based encryption. In *Advances in Cryptology – EUROCRYPT 2005, Proc. of the 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark*, volume 3494 of *Lecture Notes in Computer Science*, pages 457–473. Springer

Berlin Heidelberg, May 2005.

[44] S. F. Shahandashti and R. Safavi-Naini. Threshold attribute-based signatures and their application to anonymous credential systems. In *Progress in Cryptology – AFRICACRYPT 2009, Proc. of the 2nd International Conference on Cryptology in Africa, Gammarth, Tunisia*, volume 5580 of *Lecture Notes in Computer Science*, pages 198–216. Springer Berlin Heidelberg, June 2009.

[45] M. I. G. Vasco, F. Hess, and R. Steinwandt. Combined (identity-based) public key schemes. Cryptology ePrint Archive, Report 2008/466, 2008. `http://eprint.iacr.org/`.

[46] C. Wang and J. Huang. Attribute-based signcryption with ciphertext-policy and claim-predicate mechanism. In *Proc. of the 7th International Conference on Computational Intelligence and Security (CIS'11), Sanya, Hainan, China*, pages 905–909. IEEE, December 2011.

[47] B. Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In *Advances in Cryptology - CRYPTO 2009, Proc. of the 29th Annual International Cryptology Conference, Santa Barbara, CA, USA*, volume 5677 of *Lecture Notes in Computer Science*, pages 619–636. Springer Berlin Heidelberg, August 2009.

[48] B. Waters. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In *Public Key Cryptography – PKC 2011, Proc. of the 14th International Conference on Practice and Theory in Public Key Cryptography, Taormina, Italy*, volume 6571 of *Lecture Notes in Computer Science*, pages 53–70. Springer Berlin Heidelberg, March 2011.

[49] S. Yamada, N. Attrapadung, G. Hanaoka, and N. Kunihiro. Generic constructions for chosen-ciphertext secure attribute based encryption. In *Public Key Cryptography – PKC 2011, Proc. of the 14th International Conference on Practice and Theory in Public Key Cryptography, Taormina, Italy*, volume 6571 of *Lecture Notes in Computer Science*, pages 71–89. Springer Berlin Heidelberg, March 2011.

[50] Y. Zheng. Digital signcryption or how to achieve cost(signature & encryption) $\ll$ cost(signature) + cost(encryption). In *Advances in Cryptology — CRYPTO 1997, Proc. of the 17th Annual International Cryptology Conference Santa Barbara, CA, USA*, volume 1294 of *Lecture Notes in Computer Science*, pages 165–179. Springer Berlin Heidelberg, August 1997.

Author Biography

Tapas Pandit received the M.Sc degree in Pure Mathematics from Calcutta University in 2005 and M.Tech degree in Computer Science from Indian Statistical Institute, Kolkata in 2008. Currently he is pursuing Ph.D in Cryptography under the supervision of Prof. Rana Barua at Indian Statistical Institute, Kolkata, India. His research interests include public key cryptography, attribute-based cryptography and functional cryptography.

Sumit Kumar Pandey did his Master of Science in Mathematics from Indian Institute of Technology, Bombay, India in the year 2005. Then, in the year 2007, he completed his Master of Technology in Computer Science from Indian Statistical Institute, Kolkata, India. Then, in the year 2015, he obtained his PhD degree in the field of Computer Science from Indian Statistical Institute, Kolkata, India. His research interest is in both public key and private key cryptography which include generic constructions of different cryptographic primitives like encryptions, signatures. signcryptions etc. over various public key paradigms viz. traditional public key, identity based and attribute based setting. Moreover, he has worked in the design of cryptographically significant MDS matrices which are, in particular, used in the diffusion layer of many block ciphers and hash functions. During

the year 2012 to 2015, he served as a faculty member in CR RAO AIMSCS, Hyderabad, India. Currently, he is working as a Research Fellow in School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore.

Rana Barua received his Ph.D. under the supervision of Ashok Maitra in Descriptive Set Theory in 1987 from Indian Statistical Institute, Kolkata. Since then he has been at Indian Statistical Institute in various capacities and is currently a Professor of Mathematics in the Stat-Math Unit. He was also the Acting Head of the R.C.Bose Centre of Cryptology and Security, Indian Statistical Institute, Kolkata. He has been a visiting faculty at the University of Miami, Coral Gables during 1999-2000. Apart from Descriptive Set Theory, he has worked in Recursion Theory, Automata Theory, DNA Computing and Simple Voting Games and has made significant contributions in these areas. His current research interest is in Cryptography, in general, and cryptographic protocols, in particular.