Attribute-Based Signcryption : Signer Privacy, Strong Unforgeability and IND-CCA2 Security in Adaptive-Predicates Attack[*]

Tapas Pandit
Stat-Math Unit
Indian Statistical Institute, Kolkata
tapasgmmath@gmail.com

Sumit Kumar Pandey
C R Rao, AIMSCS, Hyderbad
emailpandey@gmail.com

Rana Barua
Stat-Math Unit
Indian Statistical Institute, Kolkata
rana@isical.ac.in

Abstract

An Attribute-Based Signcryption (ABSC) is a natural extension of Attribute-Based Encryption (ABE) and Attribute-Based Signature (ABS), where we 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, IND-CCA2* secure in *adaptive-predicates* attack and achieves *signer privacy*. Secondly, 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 $\mathcal{C}t\mathcal{E}\&\mathcal{S}$ 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 $\mathcal{C}t\mathcal{E}\&\mathcal{S}$), we call it "Commit then Encrypt and Sign then Sign" ($\mathcal{C}t\mathcal{E}\&\mathcal{S}t\mathcal{S}$). The last signature is done using a strong OTS scheme. Since the non-repudiation is achieved by $\mathcal{C}t\mathcal{E}\&\mathcal{S}$ paradigm, our systems also achieve the same.

Keywords: Attribute-based encryption, Attribute-based signature, Attribute-based signcryption, Commitment scheme.

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) [SW05] till to date many schemes have been proposed, some of them are CP-ABE [BSW07, LOS$^+$10, OT10, Wat11, LW12], some of them are KP-ABE [GPSW06, OSW07, LOS$^+$10, OT10, ALdP11], most of them are selectively secure

[*]This article is the full version of the paper that appeared in the

under chosen plaintext attack (CPA) [GPSW06, Wat11, OSW07, ALdP11], few of them are adaptively secure under CPA [OT10, LOS$^+$10, OT12] and very few of them are secure under chosen ciphertext attack (CCA) [OT10] for general policies. But, there are techniques [CHK04, BK05, YAHK11, NP15] to convert a CPA secure scheme to CCA secure scheme. 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 CP-ABS [OT11, MPR08, LAS$^+$10, MPR10] and its dual form, where the role of the policy and the set of attributes are reversed, is called KP-ABS [SSN09]. Similar to the traditional signature, ABS can be weak existential unforgeable[1] [OT11, MPR08, MPR10, LAS$^+$10] or strong existential unforgeable under chosen message attack (CMA). Most of the ABS [SSN09] proposed so far are weak existential unforgeable. But, by a simple technique [HWZ07] one can obtain strongly unforgeable signature scheme from weak unforgeable scheme. Since here the message is signed under a policy, similar to ABE there are two types of unforgeability, selective-predicate [SSN09, LAS$^+$10] and adaptive-predicate [OT11, MPR08, MPR10].

Zheng [Zhe97] introduced the concept of signcryption that provides an efficient way of achieving the message confidentiality and an authenticity together as compared to "Sign then Encrypt" approach. But they have not given any formal security proof as no formal security model was known to them. Then J.Baek et al. [BSZ02] first formalized the security notion for signcryption. Later An et al. [ADR02] proposed the generic constructions of signcryption in three paradigm, "Sign then Encrypt ($\mathcal{S}t\mathcal{E}$)", "Encrypt then Sign ($\mathcal{E}t\mathcal{S}$)" and "Commit then Encrypt and Sign ($\mathcal{C}t\mathcal{E}\&\mathcal{S}$)". As compared to $\mathcal{S}t\mathcal{E}$ and $\mathcal{E}t\mathcal{S}$ paradigms, $\mathcal{C}t\mathcal{E}\&\mathcal{S}$ 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 $\mathcal{C}t\mathcal{E}\&\mathcal{S}$ paradigm both Signcrypt and Unsigncrypt run faster as compared to other two paradigms. The generic constructions in [ADR02] were proven in two users model in PKI setting, but using some minor modification one can have the same security in multi user setting. Since it's debut several signcryption schemes [MMS09, MLM03, LQ04a, LQ04b, DFMS10, CML05, Boy03] have been proposed either in PKI setting or in IBE setting.

Meanwhile S.Haber et al. [HP01] first proposed the idea of combining public-key schemes, where an encryption scheme and a signature scheme are combined to have the common public parameters and the key. But the Encrypt and Decrypt (resp. Sign and Ver) of the encryption (resp. signature) scheme were kept unchanged in the combined scheme. The security model is called joint security of the combined public-key schemes, where in message confidentiality the adversary is given only the encryption component of the challenge message but not the signature and in authenticity the adversary is has to forge a signature. In both cases, the adversary will get access to some oracles. Later, Vasco et al. showed in [VHS08] that the IBE scheme [BF01] and the IBS scheme [Hes02] can be combined in the joint security model. Then, Paterson et al. showed in [PSST11] how to combine an IBE, a short signature and the data encapsulation mechanism (DEM) to have a combined public-key schemes. However, in this joint security model semantic security of the message is not possible if the signature of the challenge message is additionally given with the challenge ciphertext.

It is natural to ask whether signcryption can be extended to the context of attribute-based cryptography. It was Gagné et. al. [GNSN10] who first answered the question but the policy considered in their construction (called attribute-based signcryption) was a threshold policy. Basically in their construction, the structure of Fuzzy IBE in [SW05] and a new efficient threshold ABS were used as encryption primitive and signature primitive respectively. Subsequently, Emura et al. [EMR12] proposed a dynamic attribute-

[1] Unless stated, existential unforgeable means weak existential unforgeable throughout this paper

based signcryption (ABSC), where access structures of encryptor can be changed without re-issuing the secret keys of the users. Both the signcryption scheme were shown to be secure (confidentiality and authenticity) under selective-predicate attack. Since ABSC is a natural extension of both ABE and ABS, and the signer privacy is preserved in ABS, so the signer privacy property is supposed to be inherited in ABSC as well. But the later ABSC scheme lacks the property of signer privacy.

Chen et al. [CCL^{+}12] proposed a scheme in combined public-key framework but in attribute-based flavor. In their scheme the ABE and ABS modules have the same public parameters and the key distribution. Their scheme is based on the construction of Waters [Wat11] and was shown to be secure (selectively) in the joint security model. Then they extended it to have a combined attribute-based signcryption ($\mathcal{S}t\mathcal{E}$ paradigm).

Recently, Rao et al. [RD14] proposed a ABSC scheme with the constant size signcryption using the technique of KP-ABE [RD13] and a new KP-ABS [RD14]. Moreover, the number of pairings involved in unsigncryption is 6, but the key size is $\mathcal{O}(|\mathcal{U}_s|\ell_s + |\mathcal{U}_e|\ell_e)$, where $|\mathcal{U}_s|$ (resp. $|\mathcal{U}_e|$) is size of the signer (resp. receiver) attribute universe $\mathcal{U}_s$ (resp. $\mathcal{U}_e$) and ℓ_s (resp. ℓ_e) is size of the signer (resp. receiver) policy. Both the confidentiality and unforgeability of their proposed scheme were shown to be secure in selective-predicate model. The signer-privacy considered in their paper is a weaker version as compared to the signer-privacy given in this full version. The unsigncryption time for [RD14] shown in the Table 1 is basically the number of exponentiations.

Table 1: Performance of our CP-ABSC scheme. CS and $|A|$ stand for the common setup and cardinality of the set A respectively. The schemes supporting the common setup have the single key extraction algorithm and in this case, we use A to indicate the user set of attributes. Otherwise two set of attributes, A_s and A_e are used respectively for signcryption and unsigncryption. In later case, the individual key sizes are separated by comma (,). Let ℓ_s and ℓ_e respectively denote the size of the signer policy Γ_s and receiver policy Γ_e. $\mathfrak{M}$ stands for maximum # repetition of an attribute in an access policy. Let ω_s, ω_e and d respectively represent the signing set of attributes, encryption set of attributes and threshold value in [GNSN10]. $\mathcal{U}_e$ and $\Im$ respectively denote the attribute universe involved in encryption and length of verification key for OTS. $\theta_e = 2|A_e| + 2\Im + 1$. The sizes of the commitment and the one-time signature are described by $\wp$. Let $|\mathcal{I}_B|$ (resp. $|\mathcal{I}_A|$) be the minimum # row in the policy Γ_e (resp. Γ_s) labeled by the attribute set B (resp. A) to compute the target vector $\vec{1}$. Let ℓ be the length of bit string in the range of a hash function H_2 involved in the scheme [RD14]. The key size and signcryption size are measured by # group elements involved in the key and signcryption respectively. The time for signcrypt is # exponentiations to construct a signcryption, whereas the time for unsigncrypt is both # exponentiations and # pairings.

Scheme	CS	Key size	Signcryption size	Signcrypt time	Unsigncrypt time
[GNSN10]	No	$2\|A_s\|, 3\|A_e\|$	$\mathcal{O}(\|\omega_s\| + \|\omega_e\|)$	$\mathcal{O}(\|\omega_s\| + \|\omega_e\|)$	$\mathcal{O}(\|\omega_s\| + d)$
[EMR12]	No	$2\|A_s\|, \theta_e$	$\mathcal{O}(\ell_s + \|\mathcal{U}_e\| + \Im)$	$\mathcal{O}(\ell_s + \|\mathcal{U}_e\| + \Im)$	$\mathcal{O}(\ell_s + \|\mathcal{U}_e\| + \Im)$
[CCL^{+}12]	Yes	$\mathfrak{M}\|A\| + 2$	$2\ell_s + \ell_e + 4$	$\mathcal{O}(\ell_s) + \mathcal{O}(\ell_e)$	$\mathcal{O}(\ell_s) + \mathcal{O}(\|\mathcal{I}_B\|)$
[RD14]	No	$\|\mathcal{U}_s\|\ell_s, \|\mathcal{U}_e\|\ell_e$	6	$\mathcal{O}(\mathcal{I}_A) + \mathcal{O}(\ell)$	$\mathcal{O}(\mathcal{I}_B) + \mathcal{O}(\ell)$
Our	Yes	$\mathfrak{M}\|A\| + 2$	$2\ell_s + 2\ell_e + 5 + \wp$	$Max\{\mathcal{O}(\ell_s), \mathcal{O}(\ell_e)\}$	$Max\{\mathcal{O}(\ell_s), \mathcal{O}(\|\mathcal{I}_B\|)\}$

1.1 Our Approach and Contribution

Our constructions are almost in the flavor of $\mathcal{C}t\mathcal{E}\&\mathcal{S}$ paradigm. In $\mathcal{C}t\mathcal{E}\&\mathcal{S}$ paradigm, a message m is first committed to $(\breve{c}, \breve{d})$, then the commitment part $\breve{c}$ and decommitment part $\breve{d}$ are respectively signed to σ and encrypted to ϱ in parallel to produce the signcryption $\Upsilon := (\breve{c}, \sigma, \varrho)$. Similarly, in unsigncryption

Table 2: Security features of our CP-ABSC scheme. the abbreviations SAS, EAS, Auth., Conf., NR, SP, APM, NK, MAT, MSP, AGW, KP and CP stand for signing access structure, encryption access structure, signcryption unforgeability, confidentiality of message, non-repudiation, signer-privacy, adaptive-predicates model, not known, monotone access tree, monotone span program, AND-gate with wildcard respectively, key-policy and ciphertext-policy.

Scheme	Type	SAS	EAS	Auth.	Conf.	NR	SP	APM
[GNSN10]	KP	Threshold	Threshold	wUF-CMA	IND-CCA2	No	NK	No
[EMR12]	CP	MAT	AGW	sUF-CMA	IND-CCA2	Yes	No	No
[CCL$^+$12]	CP	MSP	MSP	sUF-CMA	IND-CCA2	Yes	Yes	No
[RD14]	KP	MSP	MSP	sUF-CMA	IND-CCA2	Yes	Yes	No
Our	CP	MSP	MSP	sUF-CMA	IND-CCA2	Yes	Yes	Yes

the verification (to verify σ) and the decryption (to get the $\breve{d}$) run in parallel to extract the message as $m := \mathsf{Open}(\breve{c}, \breve{d})$. But this generalized construction [ADR02] never achieves strong unforgeability (resp. CCA2 security) in the insider security model as long as the primitive encryption algorithm (resp. the primitive sign algorithm) is probabilistic.

Our first CP-ABSC construction achieves *signer privacy*, *adaptive-predicates weak unforgeability*, and *adaptive-predicates IND-CCA2* security in the standard model. Moreover, our constructions support the combined public-key environment of "Combined Public-Key scheme", viz, both the primitives, encryption and signature have a common setup, i.e., the public parameters and key are identical. Suppose we want a signcryption for a message m under the policies[2] (Γ_s, Γ_e). Let $\sigma := (\vec{S}_0, \ldots, \vec{S}_{\ell_s})$ be the signature for $(\breve{c}, \Gamma_s)$, generated by a primitive CP-ABS, where $(\breve{c}, \breve{d}) \longleftarrow \mathsf{Commit}(m)$. Let $\varrho_0 := (\vec{C}_0, \ldots, \vec{C}_{\ell_e})$ be the ciphertext generated by a primitive CP-ABE that conceals $\breve{d}$ under a policy Γ_e. To achieve the CCA2 security, we first bind all the components $\vec{S}_i$'s and $\vec{C}_i$'s through a collision resistant hash function $H_e : \{0,1\}^* \longrightarrow \mathbb{Z}_N$ to $h_e := H_e(\Gamma_e, \Gamma_s, \breve{c}, \varrho_0, \sigma)$. Then we encode h_e using a secret s_e involved in the encryption of the primitive CP-ABE and Boneh-Boyen hash technique [BB04] to an additional ciphertext component C_{ℓ_e+1}. This basically prevents the adversary $\mathscr{A}$ from changing the challenge signcryption except the component C_{ℓ_e+1}, but if it gets changed then it will be recognized via a verification process. If the primitive CP-ABS scheme is weak unforgeable and the commitment scheme has relaxed-binding property, then proposed CP-ABSC scheme is shown to be weak unforgeable.

Our second CP-ABSC scheme additionally achieves *strong unforgeability* in *adaptive-predicates* attack[3]. First notice that in the former scheme the adversary can modify the replied signcryption for a message (m, Γ_s, Γ_e) : since $\mathscr{A}$ has access to key $\mathcal{SK}_A$ with $\Gamma_e(A) = \mathsf{True}$, so it can extract $\breve{d}$ from ϱ and then re-encrypts it to get modified (new) signcryption for the same message (m, Γ_s, Γ_e). Therefore, the former scheme does not achieve the strong unforgeability. The later scheme is obtained by combining the former scheme and a strong one-time signature (OTS) scheme. Essentially, we sign $h_e || C_{\ell_e+1}$ using strong OTS scheme to guarantee that the signcryption for a message can not be altered even if the adversary knows the unsigncryption key. Surprisingly, the strong unforgeability of this CP-ABSC scheme relies only on the weak unforgeability of the primitive CP-ABS scheme and the strong unforgeability of the primitive strong OTS scheme, i.e., no more relaxed-binding property of the primitive commitment scheme is required.

[2] Γ_s and Γ_e are respectively signer policy (i.e., on whom behalf, signer signs m) and receiver policy (i.e., who will be eligible for this plaintext m)

[3] We remark that adaptive-predicates IND-CCA2 security (resp. existential unforgeability) and IND-CCA2 security (resp. existential unforgeability) in adaptive-predicates attack both carry the same meaning

The primitive CP-ABE scheme considered here is a (CCA2) variant[4] of CP-ABE scheme of Lewko et al. in [LOS$^+$10]. Our primitive CP-ABS scheme (in section 3) has the similar structure as of ABS scheme in the combined public-key framework [CCL$^+$12] except - (a) the encoding from hash of message to group element, and (b) the bilinear pairing groups. The ABS scheme of [CCL$^+$12] was proven in selective-predicate model, whereas ours is shown to be secure in adaptive-predicate model. Since the adaptive security (confidentiality and authenticity) is one of the main motivations of our work, we must require the adaptive-unforgeability of the primitive CP-ABS scheme. Therefore, the ABS of [CCL$^+$12] can not be applied directly to our CP-ABSC schemes. Another reason for moving prime to composite order pairing groups is to fit the ABS scheme to CP-ABE scheme of [LOS$^+$10]. There are many commitment schemes [DF02, HM96, Ped91] suitable for our systems, but we use them as a black box in our constructions.

Summary of our contribution. To the best of our knowledge, this is the first scheme having strong unforgeability and IND-CCA2 security in adaptive-predicates model. Since our solution supports $\mathcal{C}t\mathcal{E}\&\mathcal{S}$ paradigm, Signcrypt and Unsigncrypt run faster as compared to other paradigms, viz, $\mathcal{E}t\mathcal{S}$ and $\mathcal{S}t\mathcal{E}$. Our system is based on the common setup, i.e the public parameters and key are same for both the encryption and signature module. In addition it supports non-repudiation, dynamic property and signer privacy. A details comparisons of performance and the security features between our scheme and others are given in Table 1 and Table 2. The proofs of confidentiality and unforgeability are based on the dual system methodology of [Wat09]. The unforgeability of the CP-ABSC scheme appeared in ProvSec, 2014 was proven without giving unsigncryption oracle access to the adversary $\mathscr{A}$. However, in Appendix E we provide the proof of unforgeability of CP-ABSC scheme (in section 7), where $\mathscr{A}$ is given access to unsigncryption oracle.

Discussion First of all note that our proposed solution is not generic. One may think that using the generic construction of An et al. [ADR02] such constructions are possible, but this is not possible for the following reasons : We first emphasize that $\mathcal{C}t\mathcal{E}\&\mathcal{S}$ paradigm preserves only weak unforgeability and IND-gCCA2 (see footnote [5]). But here our proposed scheme attains both strong unforgeability and IND-CCA2 security in adaptive-predicates attack. Secondly, our solution supports the common setup for encryption and signature in combined public-key environment and so the security proof can not carry through in $\mathcal{C}t\mathcal{E}\&\mathcal{S}$ paradigm. Therefore, we would say our first construction is almost in $\mathcal{C}t\mathcal{E}\&\mathcal{S}$ paradigm, but unlike to $\mathcal{C}t\mathcal{E}\&\mathcal{S}$, an extra component is computed using signature and ciphertext components. And our construction is in the flavor of $\mathcal{C}t\mathcal{E}\&\mathcal{S}t\mathcal{S}$ paradigm, where the last sign is done using a strong one time signature.

1.2 Organization

This paper is organized as follows. For better readable, we provide notations, composite order pairing and hardness assumptions in section 2. An adaptive-predicate existential unforgeable CP-ABS scheme and its security are provided respectively in section 3 and 4. An adaptive-predicates weak existential unforgeable and adaptive-predicates IND-CCA2 secure CP-ABSC scheme and its security are given respectively in section 5 and 6. In section 7, our adaptive-predicates strong existential unforgeable and adaptive-predicates IND-CCA2 secure CP-ABSC scheme and its security are demonstrated.

[4] This is not explicitly given but the signcryption scheme implicitly contains it

[5] IND-gCCA2 is a weaker security notion than IND-CCA2. For details refer to [ADR02]

2 Preliminaries

Basic notation, definitions and hardness assumptions are provided in this section. For definition and security model of Strongly Unforgeable One-Time Signature, CP-ABS and CP-ABSC, refer to Appendix A, B and C respectively.

Notation The notations $[\ell]$ and g_T respectively stand for $\{i \in \mathbb{N} : 1 \leq i \leq \ell\}$ and the set $e(g, g)$, where e is a bilinear pairing. Let the vectors $\vec{1}$ and $\vec{0}$ respectively denote $(1, 0, \ldots, 0)$ and $(0, 0, \ldots, 0)$, where the length of the vectors will be understood from the context. Let $\vec{Y} := (y_1, \ldots, y_n)$ and $\vec{W} := (w_1, \ldots, w_n)$ be two vectors, then $\vec{Y}.\vec{W}$ denotes the dot product of $\vec{Y}$ and $\vec{W}$, i.e., $\vec{Y}.\vec{W} := \sum_{i=1}^{n} y_i w_i$. For $S \subset \mathbb{Z}_N^{\ell_s}$ and $\vec{\alpha} \in \mathbb{Z}_N^{\ell_s}$, we define $\vec{\alpha} + S := \{\vec{\alpha} + \vec{\beta} \mid \vec{\beta} \in S\}$. For a set X, $x \xleftarrow{\text{R}} X$ denotes that x is randomly picked from X according to the distribution R. Likewise, $x \xleftarrow{\text{U}} X$ indicates x is uniformly selected from X. To better understand the schemes, we use two subscripts, s and e respectively for encryption and signature. Through out this paper, we will use the symbol $\Gamma := (M, \rho)$ for the monotone span programs, where $\ell \times n$ stands for the order of the matrix M. For an access structure Γ and a set attributes A, $\Gamma(A)$ stands for boolean variable, i.e, $\Gamma(A) = \mathsf{True}$ if A satisfies Γ, else $\Gamma(A) = \mathsf{False}$. For a matrix M_e (resp. M_s), the symbol $\vec{M}_e^{(i)}$ (resp. $\vec{M}_s^{(i)}$) represents the i^{th} row of the matrix M_e (resp. M_s). The subscripts i and superscript (i) will be used for indexing. Let $str_1 || \ldots || str_n$ denote the concatenation of the strings, $str_1, \ldots, str_n \in \{0, 1\}^*$. $Alg_1 \parallel \ldots \parallel Alg_n$ stands for the parallel execution of the algorithms, $Alg_1, \ldots, Alg_n$.

Definition 2.1 (Access Structure)**.** Let $\mathcal{P} = \{P_1, P_2, ..., P_n\}$ be a set of attributes. A collection $\Gamma \subset 2^{\mathcal{P}}$ 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 (respectively, MAS) is a collection (respectively, monotone collection) Γ of non-empty subsets of $\mathcal{P}$, i.e., $\Gamma \subset 2^{\mathcal{P}} \setminus \{\emptyset\}$.

Definition 2.2 (Linear Secret Sharing Scheme)**.** [Bei96] A secret sharing scheme Π for an access structure Γ over a set of parties $\mathcal{P} = \{P_1, P_2, ..., P_n\}$ is called linear (over $\mathbb{Z}_p$) if

- The shares for each party form a vector over $\mathbb{Z}_p$.
- There exists a matrix, A called the share generating matrix for Π. The matrix A has l rows and n columns. For all $i = 1, 2, .., l$, the i^{th} row of A is labeled by a party $\rho(i)$ (ρ is the function from $\{1, 2, ..., l\}$ to $\mathcal{P}$). When we consider the column vector $\vec{v} = (s, r_2, ..., r_n)$, where $s \in \mathbb{Z}_p$ is the secret to be shared and $r_2, ..., r_n \in^R k$ are chosen, then $A\vec{v}$ is the vector of l shares of the secret s according to Π. The share $(A\vec{v})_i$ belongs to party $\rho(i)$.

Every Linear Secret Sharing Scheme (LSSS) enjoys the linear reconstruction property by [Bei96]: Suppose that Π is an LSSS for an access structure Γ. Let $S \in \Gamma$ be an authorized set. Let $\mathcal{I} = \{i \in [l] : \rho(i) \in S\}$. Then there exists constants $\{\alpha_i \in \mathbb{Z}_p\}_{i \in \mathcal{I}}$ such that if $\{s_i\}$ are valid shares of any secret s according to Π, then $\sum_{i \in \mathcal{I}} \alpha_i s_i = s$. These constants $\{\alpha_i\}$ can be found in time polynomial in the size of the share-generating matrix A. This LSSS will be used in our constructions, both ABS and CP-ABSC.

2.1 Commitment scheme

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

- Setup: It takes a security parameter κ and outputs a public commitment key $\mathcal{CK}$.

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

For correctness, it is required that[6] $\mathsf{Open}(\mathsf{Commit}(m)) = m$ for all message $m \in \mathcal{M}$, where $\mathcal{M}$ is the message space.

2.2 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:

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

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

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

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

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

Remark 2.1. The relaxed-binding property is weaker than the binding property.

2.3 Composite Order Bilinear Groups

Let $\mathcal{G}$ be an algorithm which takes 1^κ as a security parameter and returns a description of a composite order bilinear groups, $\mathcal{J} := (N := p_1 p_2 p_3, \mathbb{G}, \mathbb{G}_T, e)$, 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 such that

1. (Bilinear) $\forall g, h \in \mathbb{G}, a, b \in \mathbb{Z}_N, e(g^a, h^b) = e(g, h)^{ab}$
2. (Non-degenerate) $\exists g \in \mathbb{G}$ such that $e(g, g)$ has order N in $\mathbb{G}_T$

Let $\mathbb{G}_{p_1}, \mathbb{G}_{p_2}$ and $\mathbb{G}_{p_3}$ respectively denote the subgroups of $\mathbb{G}$ of order p_1, p_2 and p_3. Let $h_i \in \mathbb{G}_{p_i}$ and $h_j \in \mathbb{G}_{p_j}$ be arbitrary elements with $i \neq j$, then $e(h_i, h_j) = 1$. This property is called orthogonal property of $\mathbb{G}_{p_1}, \mathbb{G}_{p_2}, \mathbb{G}_{p_3}$.

2.4 Hardness Assumptions

We describe here three Decisional SubGroup (DSG) assumptions for 3 primes, DSG1, DSG2 and DSS3 in composite order bilinear groups. These assumptions were used by Lewko et al. in [LOS+10, LW10] to prove the adaptive security of their schemes in the standard model.

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

DSG1 Assumption
Define the following distribution :

$$\mathcal{J} := (N = p_1p_2p_3, \mathbb{G}, \mathbb{G}_T, e) \xleftarrow{\text{U}} \mathcal{G}(1^\lambda),\ g \xleftarrow{\text{U}} \mathbb{G}_{p_1}, X_3 \xleftarrow{\text{U}} \mathbb{G}_{p_3}$$

$$\mathcal{D} := (\mathcal{J}, g, X_3),\ T_0 \xleftarrow{\text{U}} \mathbb{G}_{p_1},\ T_1 \xleftarrow{\text{U}} \mathbb{G}_{p_1p_2}$$

Now, the advantage of an algorithm $\mathscr{A}$ in breaking DSG1 Assumption is defined by

$$\mathsf{Adv}_{\mathscr{A}}^{\text{DSG1}}(\kappa) = |Pr[\mathscr{A}(\mathcal{D}, T_0) = 1] - Pr[\mathscr{A}(\mathcal{D}, T_1) = 1]|$$

We say that the DSG1 assumption holds if for every PPT algorithm $\mathscr{A}$, the advantage $\mathsf{Adv}_{\mathscr{A}}^{\text{DSG1}}(\kappa)$ is a negligible function of the security parameter κ.

DSG2 Assumption
Define the following distribution :

$$\mathcal{J} := (N = p_1p_2p_3, \mathbb{G}, \mathbb{G}_T, e) \xleftarrow{\text{U}} \mathcal{G}(1^\lambda),\ g, X_1 \xleftarrow{\text{U}} \mathbb{G}_{p_1},\ X_2, Y_2 \xleftarrow{\text{U}} \mathbb{G}_{p_2},\ X_3, Y_3 \xleftarrow{\text{U}} \mathbb{G}_{p_3}$$

$$\mathcal{D} := (\mathcal{J}, g, X_1X_2, Y_2Y_3, X_3),\ T_0 \xleftarrow{\text{U}} \mathbb{G}_{p_1p_3},\ T_1 \xleftarrow{\text{U}} \mathbb{G}$$

Now, the advantage of an algorithm $\mathscr{A}$ in breaking DSG2 Assumption is defined by

$$\mathsf{Adv}_{\mathscr{A}}^{\text{DSG2}}(\kappa) = |Pr[\mathscr{A}(\mathcal{D}, T_0) = 1] - Pr[\mathscr{A}(\mathcal{D}, T_1) = 1]|$$

We say that the DSG2 assumption holds if for every PPT algorithm $\mathscr{A}$, the advantage $\mathsf{Adv}_{\mathscr{A}}^{\text{DSG2}}(\kappa)$ is a negligible function of the security parameter κ.

DSG3 Assumption
Define the following distribution :

$$\mathcal{J} := (N = p_1p_2p_3, \mathbb{G}, \mathbb{G}_T, e) \xleftarrow{\text{U}} \mathcal{G}(1^\lambda),\ \alpha, s \xleftarrow{\text{U}} \mathbb{Z}_N,\ g \xleftarrow{\text{U}} \mathbb{G}_{p_1},\ X_2, Y_2, Z_2 \xleftarrow{\text{U}} \mathbb{G}_{p_2},\ X_3 \xleftarrow{\text{U}} \mathbb{G}_{p_3}$$

$$\mathcal{D} := (\mathcal{J}, g, g^\alpha X_2, g^s Y_2, Z_2, X_3),\ T_0 := e(g,g)^{\alpha s},\ T_1 \xleftarrow{\text{U}} \mathbb{G}_T$$

Now, the advantage of an algorithm $\mathscr{A}$ in breaking DSG3 Assumption is defined by

$$\mathsf{Adv}_{\mathscr{A}}^{\text{DSG3}}(\kappa) = |Pr[\mathscr{A}(\mathcal{D}, T_0) = 1] - Pr[\mathscr{A}(\mathcal{D}, T_1) = 1]|$$

We say that the DSG3 assumption holds if for every PPT algorithm $\mathscr{A}$, the advantage $\mathsf{Adv}_{\mathscr{A}}^{\text{DSG3}}(\kappa)$ is a negligible function of the security parameter κ.

3 Basic Ciphertext-Policy Attribute-Based Signature

Illustrated here is a basic ciphertext-policy attribute-based signature (CP-ABS) scheme for monotone span program (MSP) in the composite order pairing groups $(N := p_1p_2p_3, \mathbb{G} := \mathbb{G}_{p_1} \times \mathbb{G}_{p_2} \times \mathbb{G}_{p_2}, \mathbb{G}_T, e)$, 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. As we mentioned earlier that the proposed CP-ABS scheme has the similar structure to that of [CCL$^+$12] except some minor modifications, viz., the encoding function from hash of messages to group elements and pairing groups. To guarantee the unforgeability of the ABS scheme in adaptive-predicate model, we allow such modifications. In this basic CP-ABS construction, the policies, i.e., MSPs are restricted to have each entry of row labeling function ρ_s to be distinct. In other word, the row labeling functions ρ_s of the monotone span programs $\Gamma_s := (M_s, \rho_s)$ are injective. From this basic CP-ABS construction one can easily lift to full CP-ABS construction by a mechanism described in appendix D.

$\mathsf{Setup}(1^\kappa, \mathcal{U})$: It executes $\mathcal{G}(1^\kappa)$ to have composite order bilinear groups descriptor, $\mathcal{J} := (N := p_1p_2p_3, \mathbb{G}, \mathbb{G}_T, e)$ with known factorization p_1, p_2 and p_3 of N. It chooses $g \xleftarrow{\text{U}} \mathbb{G}_{p_1}, X_3 \xleftarrow{\text{U}} \mathbb{G}_{p_3}$, $a, a_s, b_s, \alpha \xleftarrow{\text{U}} \mathbb{Z}_N$ and $t_i \xleftarrow{\text{U}} \mathbb{Z}_N$ for each attribute $i \in \mathcal{U}$. It then sets $u_s := g^{a_s}, v_s := g^{b_s}, T_i := g^{t_i}$ for $i \in \mathcal{U}$. Let $H_s : \{0,1\}^* \longrightarrow \mathbb{Z}_N$ be a hash function. The public parameters and master secret are given by

$$\mathcal{PP} \ := (\mathcal{J}, g, g^a, u_s, v_s, g_T^\alpha, \{T_i\}_{i \in \mathcal{U}}, X_3, H_s)$$
$$\mathcal{MSK} := (\alpha).$$

$\mathsf{KeyGen}(\mathcal{PP}, \mathcal{MSK}, A)$: It picks $t \xleftarrow{\text{U}} \mathbb{Z}_N, R, R'_0 \xleftarrow{\text{U}} \mathbb{G}_{p_3}$. For each attribute $i \in A$, the algorithm chooses $R_i \xleftarrow{\text{U}} \mathbb{G}_{p_3}$ and outputs the secret key

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

$\mathsf{Sign}(\mathcal{PP}, m, \mathcal{SK}_A, \Gamma_s := (M_s, \rho_s))$: Let M_s be an $\ell_s \times n_s$ matrix. Suppose $\Gamma_s(A) = \mathsf{True}$, then there exist $\mathcal{I}_A \subset [\ell_s]$ and $\{\alpha_s^{(i)}\}_{i \in \mathcal{I}_A}$ such that $\sum_{i \in \mathcal{I}_A} \alpha_s^{(i)} \vec{M}_s^{(i)} = \vec{1}$. It selects $\vec{\beta} \xleftarrow{\text{U}} \{\vec{\beta} = (\vec{\beta}_1, \ldots, \vec{\beta}_{\ell_s}) \in \mathbb{Z}_N^{\ell_s} \mid \sum_{i \in [\ell_s]} \beta_i \vec{M}_s^{(i)} = \vec{0}\}$. Suppose $\mathcal{SK}_A := [A,\ K := g^{\alpha+at} R,\ L := g^t R'_0,\ K_i := {T_i}^t R_i,\ \forall i \in A]$, then it re-randomizes the key $\mathcal{SK}_A$ as follows:

$$\begin{aligned} \tilde{\mathcal{SK}}_A &:= [A,\ \tilde{K} := K.g^{a\hat{t}},\ \tilde{L} := L.g^{\hat{t}},\ \tilde{K}_i := K_i.{T_i}^{\hat{t}},\ \forall i \in A], \text{where } \hat{t} \xleftarrow{\text{U}} \mathbb{Z}_N \\ &:= [A,\ \tilde{K} := g^{\alpha + a\tilde{t}} R,\ \tilde{L} := g^{\tilde{t}} R'_0,\ \tilde{K}_i := {T_i}^{\tilde{t}} R_i,\ \forall i \in A], \text{where } \tilde{t} := t + \hat{t} \end{aligned}$$

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

$$\vec{S}_0 := (\ \tilde{K}(u_s^{h_s} v_s)^{r_s} \bar{R},\ g^{r_s} \bar{R}_0\ )$$
$$\vec{S}_i := (\ \tilde{L}^{\alpha_s^{(i)}} (g^\tau)^{\beta_i} \bar{R}_i,\ (\tilde{K}_{\rho_s(i)})^{\alpha_s^{(i)}} (T_{\rho_s(i)})^{\tau\beta_i} \bar{R}'_i\ ) \text{ for } i \in [\ell_s]. \text{ For } i \notin \mathcal{I}_A,\ \alpha_s^{(i)} := 0$$

After simplification, it gives

$$\vec{S}_0 := (\ g^{\alpha + a\tilde{t}} (u_s^{h_s} v_s)^{r_s} \tilde{R},\ g^{r_s} \tilde{R}_0\ ), \text{ where } \tilde{R} := R\bar{R},\ \tilde{R}_0 := \bar{R}_0$$
$$\vec{S}_i := (\ (g^{\tilde{t}})^{\alpha_s^{(i)}} (g^\tau)^{\beta_i} \tilde{R}_i,\ \ (T^{\tilde{t}}_{\rho_s(i)})^{\alpha_s^{(i)}} (T^\tau_{\rho_s(i)})^{\beta_i} \tilde{R}'_i\ ), \text{ where } \tilde{R}_i := (R'_0)^{\alpha_s^{(i)}} \bar{R}_i,\ \tilde{R}'_i := R_{\rho_s(i)}^{\alpha_s^{(i)} + \tau\beta_i} \bar{R}'_i$$

The final output (signature) is $\sigma := (\vec{S}_0, \{\vec{S}_i\}_{i \in [\ell_s]})$

$\mathsf{Ver}(\mathcal{PP}, m, \sigma, \Gamma_s)$: 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 $\vec{u}_s := (s, u_2, \ldots, u_{n_s}) \xleftarrow{\text{U}} \mathbb{Z}_N^{n_s}$ and $r_s^{(i)} \xleftarrow{\text{U}} \mathbb{Z}_N$ for $i \in [\ell_s]$. It computes $h_s := H_s(m || \Gamma_s)$. Let $M_s^{(i)}$ denote the i^{th} row of the matrix, M_s and let $\lambda_s^{(i)} := \vec{M}_s^{(i)}.\vec{u}_s$. The components of verification text are given by

$$\vec{V}_0 := (\ g^s, (u_s^{h_s} v_s)^s, g_T^{\alpha s}\ )$$
$$\vec{V}_i := (\ g^{a\lambda_s^{(i)}} T_{\rho_s(i)}^{-r_s^{(i)}},\quad g^{r_s^{(i)}}\ ), \text{ for } i \in [\ell_s]$$

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

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

Correctness.

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

4 Security Proof of CP-ABS

Theorem 4.1. *The proposed attribute-based signature scheme in section 3 is perfectly private.*

Proof. Let A_1 and A_2 be two sets of attributes and $\Gamma_s := (M_s, \rho_s)$ be an access policy such that $\Gamma_s(A_1) = \Gamma_s(A_2) = \mathsf{True}$. Then, there exist sets $\mathcal{I}_{A_1} \subset [\ell_s]$, $\mathcal{I}_{A_2} \subset [\ell_s]$ and $\{\alpha_{s1}^{(i)}\}_{i\in[\ell_s]}$, $\{\alpha_{s2}^{(i)}\}_{i\in[\ell_s]}$ such that $\sum_{i\in\mathcal{I}_{A_1}} \alpha_{s1}^{(i)}\vec{M}_s^{(i)} = \vec{1}$ and $\sum_{i\in\mathcal{I}_{A_2}} \alpha_{s2}^{(i)}\vec{M}_s^{(i)} = \vec{1}$. In other word, there exist vectors $\vec{\alpha}_{s1}$ and $\vec{\alpha}_{s2}$ such that $\sum_{i=1}^{\ell_s} \alpha_{s1}^{(i)}\vec{M}_s^{(i)} = \sum_{i=1}^{\ell_s} \alpha_{s2}^{(i)}\vec{M}_s^{(i)} = \vec{1}$, where $\alpha_{s1}^{(i)} = 0$ if $i \notin \mathcal{I}_{A1}$ and $\alpha_{s2}^{(i)} = 0$ if $i \notin \mathcal{I}_{A2}$. We will show that the signatures σ_1 and σ_2 generated respectively by the keys $\mathcal{SK}_{A_1}$ and $\mathcal{SK}_{A_2}$ 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 parameter and $\vec{S}_0$, but since these values are chosen uniformly and independently random from $\mathbb{Z}_N$, these values also do not carry any information regarding the attribute set used to sign the message. So, $\vec{S}_0$ of any two signatures are identical. The only parts of the signature carrying the information of attributes are $\vec{S}_i$ for $i \in [\ell_s]$. Now consider two system of equations, homogeneous and non-homogeneous given below

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

$$\sum_{i=1}^{\ell_s} \alpha_s^{(i)}\vec{M}_s^{(i)} = \vec{1} \tag{2}$$

Let $\mathcal{Y}_0 := \{\vec{\beta}\ \Big|\ \sum_{i=1}^{\ell_s} \beta_i\vec{M}_s^{(i)} = \vec{0}\}$ and $\mathcal{Y}_1 := \{\vec{\alpha}_s\ \Big|\ \sum_{i=1}^{\ell_s} \alpha_s^{(i)}\vec{M}_s^{(i)} = \vec{1}\}$ respectively denote the solution set of equation 1 and 2. Let $\vec{\alpha}_{s1}$ and $\vec{\alpha}_{s2}$ be any two solutions of system 2, then we can write $\mathcal{Y}_1 = \vec{\alpha}_{s1} + \mathcal{Y}_0 = \vec{\alpha}_{s2} + \mathcal{Y}_0$. So, the distribution of $\vec{\alpha}_{s1} + \vec{\beta}$ and $\vec{\alpha}_{s2} + \vec{\beta}$ are identical for $\vec{\beta} \xleftarrow{\text{U}} \mathcal{Y}_0$. 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 $\vec{S}_i$ is $(g^{\tilde{t}\alpha_s^{(i)}+\tau\beta_i},\ g^{(\tilde{t}\alpha_s^{(i)}+\tau\beta_i)t_{\rho_s(i)}})$, the distribution of the signatures σ_1 and σ_2 are identical. □

Theorem 4.2. *The proposed basic CP-ABS scheme is adaptive-predicate existential unforgeable if DSG1, DSG2 and DSG3 assumptions hold and H_s is a collision resistant hash function.*

Due to the space limitation, the proof is provided in the Appendix B.3

5 Basic Ciphertext-Policy Attribute-Based Signcryption

In this section, we present our basic ciphertext-policy attribute-based signcryption (CP-ABSC) supporting monotone span programs. The scheme is based on the composite order bilinear pairing groups. Here we consider two policies, sender policy $\Gamma_s := (M_s, \rho_s)$ and receiver policy $\Gamma_e := (M_e, \rho_e)$. Similar to section 3, in our basic CP-ABSC scheme, both the row labeling functions ρ_s and ρ_e are assumed to be injective. By applying the mechanism illustrated in appendix D, a full CP-ABSC construction is easily obtained.

This construction is almost in the flavor of $\mathcal{C}t\mathcal{E}\&\mathcal{S}$ paradigm. To construct our scheme, we use a commitment scheme with hiding and relaxed-binding properties, CCA2 version (implicit) encryption scheme of [LOS$^+$10] and the ABS scheme described in section 3. Let $\Pi_{\mathsf{ABS}} := (\mathsf{ABS.Setup}, \mathsf{ABS.KeyGen}, \mathsf{ABS.Sign}, \mathsf{ABS.Ver})$ and $\Pi_{\mathsf{Commit}} := (\mathsf{C.Setup}, \mathsf{Commit}, \mathsf{Open})$ be respectively the ABS scheme described in section 3 and commitment scheme.

$\mathsf{Setup}(1^\kappa, \mathcal{U})$: $\mathsf{C.Setup}(1^\kappa) \longrightarrow \mathcal{CK}$, $\mathsf{ABS.Setup}(1^\kappa, \mathcal{U}) \longrightarrow (\mathcal{ABS.PP}, \mathcal{ABS.MSK})$. It chooses $a_e, b_e \overset{\mathrm{U}}{\longleftarrow} \mathbb{Z}_N$ 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 $\mathcal{ABS.PP}, \mathcal{CK}$ and u_e, v_e, H_e) and master secret are given by

$$\mathcal{PP} \;\;:= (\mathcal{I}, g, g^a, u_s, u_e, v_s, v_e, g_T^\alpha, \{T_i\}_{i \in \mathcal{U}}, X_3, H_s, H_e, \mathcal{CK})$$
$$\mathcal{MSK} := \mathcal{ABS.MSK} = (\alpha)$$

$\mathsf{KeyGen}(\mathcal{PP}, \mathcal{MSK}, A)$: $\mathcal{SK}_A \longleftarrow \mathsf{ABS.KeyGen}(\mathcal{ABS.PP}, \mathcal{MSK}, A)$

$\mathsf{Signcrypt}(\mathcal{PP}, m, \mathcal{SK}_A, \Gamma_s := (M_s, \rho_s), \Gamma_e := (M_e, \rho_e))$: Let M_s (resp. M_e) be an $\ell_s \times n_s$ (resp. $\ell_e \times n_e$) matrix. It runs $(\breve{c}, \breve{d}) \longleftarrow \mathsf{Commit}(m)$. The $\mathsf{Signcrypt}$ algorithm has two part, Sign and Encrypt, both run in parallel except one component of the ciphertext part, viz, C_{ℓ_e+1}.

***Sign*:** $\sigma := (\vec{S}_0, \{\vec{S}_i\}_{i \in [\ell_s]}) \longleftarrow \mathsf{ABS.Sign}(\mathcal{ABS.PP}, (\breve{c}||\Gamma_e), \mathcal{SK}_A, \Gamma_s := (M_s, \rho_s))$, where the components are given by

$$\vec{S}_0 := \big(\; g^{\alpha + a\tilde{t}}(u_s^{h_s} v_s)^{r_s} \tilde{R},\; g^{r_s} \tilde{R}_0 \;\big), \text{ where } h_s := H_s((\breve{c}||\Gamma_e)||\Gamma_s)$$
$$\vec{S}_i := \big(\; (g^{\tilde{t}})^{\alpha_s^{(i)}} (g^{\tau})^{\beta_i} \tilde{R}_i,\;\; (T^{\tilde{t}}_{\rho_s(i)})^{\alpha_s^{(i)}} (T^{\tau}_{\rho_s(i)})^{\beta_i} \tilde{R'}_i \;\big)$$

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

$$\vec{C}_0 := \big(\; g^{s_e},\; \breve{d}.g_T^{\alpha s_e} \;\big)$$
$$\vec{C}_i := \big(\; g^{a\lambda_e^{(i)}} T_{\rho_e(i)}^{-r_e^{(i)}}, \quad g^{r_e^{(i)}} \;\big), \text{ for } i \in [\ell_e]$$

Now, it sets $\varrho_0 := (\vec{C}_0, \{\vec{C}_i\}_{i \in [\ell_e]})$ and computes $h_e := H_e(\breve{c}, \varrho_0, \sigma)$. Then, it computes the last component

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

So, the ciphertext part of the signcryption is $\varrho := (\varrho_0, C_{\ell_e+1})$.

It outputs the signcryption $\Upsilon := (\breve{c}, \sigma, \varrho)$

$\mathsf{Unsigncrypt}(\mathcal{PP}, \Upsilon, \mathcal{SK}_B, \Gamma_s := (M_s, \rho_s))$: Let $\Gamma_e := (M_e, \rho_e)$ be the policy for receiver implicitly contained in Υ. M_s (resp. M_e) be an $\ell_s \times n_s$ (resp. $\ell_e \times n_e$) matrix. This algorithm consists of two routines, Ver and Decrypt run in parallel.

***Ver*:** $\mathsf{flag} \longleftarrow \mathsf{ABS.Ver}(\mathcal{PP}, (\breve{c}||\Gamma_e), \sigma, \Gamma_s)$. If $\mathsf{flag} = 0$, it returns $\perp$

***Decrypt*:** It computes $h_e := H_e(\breve{c}, \varrho_0, \sigma)$. Then it checks $e(g, C_{\ell_e+1}) \overset{?}{=} e(u_e^{h_e} v_e, C_{01})$ and if the equality does not hold, it returns $\perp$. If $\Gamma_e(B) \neq \mathsf{True}$, it returns $\perp$, else there exist $\mathcal{I}_B \subset [\ell_e]$ and $\{\alpha_e^{(i)}\}_{i \in \mathcal{I}_B}$ such that $\sum_{i \in \mathcal{I}_B} \alpha_e^{(i)} \vec{M}_e^{(i)} = \vec{1}$. Then, it picks $r \xleftarrow{\text{U}} \mathbb{Z}_N,\ R_0 \xleftarrow{\text{U}} \mathbb{G}_{p_3}$ and computes

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

Finally it returns the message m

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

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

$$\mathsf{Open}(\breve{c}, C_{02}/\Delta_e) = \mathsf{Open}(\breve{c}, \breve{d}) = m$$

A high level description of our scheme. In the construction above, we implicitly employ a variant of CP-ABE scheme of [LOS$^+$10] which has the same public parameters and key as the primitive ABS scheme Π_{ABS}. Let $\Pi_{\mathsf{ABE}} := (\mathsf{ABE.Setup}, \mathsf{ABE.KeyGen}, \mathsf{ABE.Encrypt}, \mathsf{ABE.Decrypt})$ be the CP-ABE scheme involved in above construction. The descriptions of $\mathsf{Signcrypt}$ and $\mathsf{Unsigncrypt}$ are given below.

$\mathsf{Signcrypt}(\mathcal{PP}, m, \mathcal{SK}_A, \Gamma_s, \Gamma_e)$: It runs $(\breve{c}, \breve{d}) \longleftarrow \mathsf{Commit}(m)$. Then, it executes in parallel $\varrho_0 \longleftarrow \mathsf{ABE.Encrypt}(\mathcal{PP}, \breve{d}, \Gamma_e)$ and $\sigma \longleftarrow \mathsf{ABS.Sign}(\mathcal{PP}, \breve{c} || \Gamma_e, \Gamma_s)$. Then, it computes $h_e := H_e(\breve{c}, \varrho_0, \sigma)$ and $C_{\ell_e+1} \longleftarrow \mathsf{fun}(\mathcal{PP}, h_e, s_e)$, for some function fun (see footnote [7]). It sets $\varrho := (\varrho_0, C_{\ell_e+1})$ and outputs $\Upsilon := (\breve{c}, \sigma, \varrho)$

$\mathsf{Unsigncrypt}(\mathcal{PP}, \Upsilon, \mathcal{SK}_B, \Gamma_s, \Gamma_e)$: It runs in parallel $\mathsf{flag} \longleftarrow \mathsf{ABS.Ver}(\mathcal{PP}, \breve{c} || \Gamma_e, \sigma, \Gamma_s)$ and $\breve{d} \longleftarrow \mathsf{ABE.Decrypt}(\mathcal{PP}, \varrho, \mathcal{SK}_B, \Gamma_e)$. If $\mathsf{flag} = 1$, it returns $\mathsf{Open}(\breve{c}, \breve{d})$ else $\perp$.

Non-Repudiation (Publicly Verifiability). Receiver gives a signcryption $\Upsilon := (\breve{c}, \sigma, \varrho)$ for (m, Γ_s, Γ_e) and the decommitment part $\breve{d}$ extracted from ϱ to a third party. Since, the primitive commitment scheme Π_{Commit} has relaxed-binding property, for given a valid pair $(\breve{c}, \breve{d})$ for m, the receiver can not fool the third party by giving $\breve{d}'$ such that $(\breve{c}, \breve{d}')$ is also a valid pair for m' with $m' \neq m$. The objective of the receiver is to convince the third party that the signcryption is actually produced by the sender. Now, the third party can verify the signature part σ of Υ against $\breve{c}$ by the verification algorithm of ABS. If it fails, it means that the receiver fails to convince the third party. Otherwise, the third part checks $m \overset{?}{=} \mathsf{Open}(\breve{c}, \breve{d})$ and if it is True then, the third is convinced that the signcryption Υ is actually sent by the claim sender else receiver fails.

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

Dynamic property. In Dynamic attribute-based system, a new attribute can be added dynamically to the system without re-issuing the whole secret key of the user. Here a user sends it's one secret key component, viz, $L := g^t R_0'$ to the PKG and then PKG will send the secret key component corresponding to the new attribute : Suppose att is a new attribute, then PKG computes $T_{att} := g^{t_{att}}$ by choosing $t_{att} \xleftarrow{\text{U}} \mathbb{Z}_N$, keeps t_{att} to itself and adds T_{att} to $\mathcal{PP}$. Then, it sets $K_{att} := L^{t_{att}} R_{att}$ by picking $R_{att} \xleftarrow{\text{U}} \mathbb{G}_{p_3}$ and returns it to the user.

6 Security Proof of CP-ABSC

6.1 Perfect Privacy of CP-ABSC

Theorem 6.1. *The proposed attribute-based signcryption scheme in section 5 is perfectly private.*

Proof. It is immediate from Theorem 4.1. □

6.2 Adaptive-Predicates IND-CCA2 security of CP-ABSC

Since, the CP-ABSC has perfect privacy, we replace through out the proofs the actual Signcrypt algorithm by an alternative algorithm, AltSigncrypt defined in section C. For notational simplicity, we do not use the term "AltSigncrypt" in the proofs. We note that all the Signcrypt oracles appeared in the proofs are basically AltSigncrypt oracles. We prove the adaptive-predicates IND-CCA2 security of our basic ABSC scheme using dual system methodology of Brent Waters [Wat09]. To utilize this methodology we define a new form of key, unsigncryption-query key. This key will be used to answer the unsigncryption-query in the security proof. In this methodology, we also define semi-functional signcryptions, semi-functional unsigncryption-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 unsigncryption-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 each queried signcryption is changed from normal to semi-functional type II via semi-functional type I. Similarly, each unsigncryption-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 $\breve{d}_b$ 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 $\breve{c}_b$ 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. Also, we use the abbreviation 'sf' and 'uq' for 'semi-functional' and 'unsigncryption-query' respectively.

Semi-functional Type I Signcryption. This is same as normal signcryption except the signature component $\vec{S}_0$ described below

$$\vec{S}_0 := \left(g^{\alpha + a\tilde{t}} (u_s^{h_s} v_s)^{r_s} \tilde{R} \boxed{g_2^{\tilde{d}}}, \; g^{r_s} \tilde{R}_0 \boxed{g_2^{\tilde{b}}} \right), \text{ where } \tilde{b}, \tilde{d} \xleftarrow{\text{U}} \mathbb{Z}_N$$

Semi-functional Type II Signcryption. This is same as sf-type I signcryption except $\tilde{b} = 0$, i.e.,

$$\vec{S}_0 := \big(\ g^{\alpha + a\tilde{t}}(u_s^{h_s} v_s)^{r_s} \tilde{R} g_2^{\tilde{d}},\ g^{r_s} \tilde{R}_0\ \big)$$

Semi-functional Type 1 Signcryption. Pick $c, \iota \overset{\text{U}}{\longleftarrow} \mathbb{Z}_N,\ \vec{v}_e \overset{\text{U}}{\longleftarrow} \mathbb{Z}_N^{n_s}$. For each $i \in [\ell_e]$, pick $\gamma_e^{(i)} \overset{\text{U}}{\longleftarrow} \mathbb{Z}_N$. For each $i \in \mathcal{U}$, choose $z_i \overset{\text{U}}{\longleftarrow} \mathbb{Z}_N$. 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 $\Upsilon := (\breve{c}, \sigma, \varrho)$ be a normal signcryption, where the ciphertext part is $\varrho = (\varrho_0, C_{\ell_e+1})$ and $\varrho_0 := (\vec{C}_0, \{\vec{C}_i\}_{i \in [\ell_e]})$. So, the sf-type 1 signcryption is obtained by modifying the ciphertext part of the normal signcryption. (See footnote [8])

$$\begin{aligned} \vec{C}_0 \quad &:= \big(\ g^{s_e} \boxed{g_2^c}, \breve{d}.g_T^{\alpha s_e}\ \big) \\ \vec{C}_i \quad &:= \Big(\ g^{a\lambda_e^{(i)}} T_{\rho_e(i)}^{-r_e^{(i)}} \boxed{g_2^{\vec{M}_e^{(i)}.\vec{v}_e + \gamma_e^{(i)} z_{\rho_e(i)}}}, \quad g^{r_e^{(i)}} \boxed{g_2^{-\gamma_e^{(i)}}}\ \Big), \text{ for } i \in [\ell_e] \\ C_{\ell_e+1} &:= (u_e^{h_e} v_e)^{s_e} \boxed{g_2^{\iota}} \end{aligned}$$

Semi-functional Type 2 Signcryption. This is same as sf-type 1 signcryption except the signature part, i.e., $\sigma := (\vec{S}_0, \{\vec{S}_i\}_{i \in [\ell_s]})$ gets change from normal to the following form : Choose $\tilde{b}, \tilde{d} \overset{\text{U}}{\longleftarrow} \mathbb{Z}_N$.

$$\begin{aligned} \vec{S}_0 &:= \big(\ g^{\alpha + a\tilde{t}}(u_s^{h_s} v_s)^{r_s} \tilde{R} \boxed{g_2^{\tilde{d}}},\ g^{r_s} \tilde{R}_0 \boxed{g_2^{\tilde{b}}}\ \big) \\ \vec{S}_i &:= \big(\ (g^{\tilde{t}})^{\alpha_s^{(i)}} (g^{\tau})^{\beta_i} \tilde{R}_i, \qquad (T_{\rho_s(i)}^{\tilde{t}})^{\alpha_s^{(i)}} (T_{\rho_s(i)}^{\tau})^{\beta_i} \tilde{R}'_i\ \big) \end{aligned}$$

Semi-functional Type 3 Signcryption. This is same as sf-type 2 signcryption except $\tilde{b} = 0$, i.e.,

$$\begin{aligned} \vec{S}_0 &:= \big(\ g^{\alpha + a\tilde{t}}(u_s^{h_s} v_s)^{r_s} \tilde{R} g_2^{\tilde{d}},\ g^{r_s} \tilde{R}_0\ \big) \\ \vec{S}_i &:= \big(\ (g^{\tilde{t}})^{\alpha_s^{(i)}} (g^{\tau})^{\beta_i} \tilde{R}_i, \quad (T_{\rho_s(i)}^{\tilde{t}})^{\alpha_s^{(i)}} (T_{\rho_s(i)}^{\tau})^{\beta_i} \tilde{R}'_i\ \big) \end{aligned}$$

Semi-functional Type 4 Signcryption. This is same as sf-type 3 signcryption except the following

$$\vec{C}_0 := \big(g^{s_e} g_2^c, \boxed{\breve{d}.\hat{g}_t}\big), \text{ where } \hat{g}_t \overset{\text{U}}{\longleftarrow} \mathbb{G}_T$$

Semi-functional Type 1 Key. Choose $b, d \overset{\text{U}}{\longleftarrow} \mathbb{Z}_N$. First create a normal key

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

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

$$\mathcal{SK}_A := [A,\ K := g^{\alpha + at} R \boxed{g_2^d},\ L := g^t R'_0 \boxed{g_2^b},\ K_i := T_i{}^t R_i \boxed{g_2^{bz_i}},\ \forall i \in A]$$

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

$$\mathcal{SK}_A := [A,\ K := g^{\alpha + at} R g_2^d,\ L := g^t R'_0,\ K_i := T_i{}^t R_i,\ \forall i \in A]$$

[8] sf-type I (resp. II) signcryption and sf-type 1 (resp. 2) signcryption are different

Normal Unsigncryption Query Key. Let $\Upsilon := (\check{c}, \sigma, \varrho)$ be a unsigncryption query for (Γ_e, Γ_s), where the ciphertext part is $\varrho := (\varrho_0, C_{\ell_e+1})$ and $\sigma := (\vec{S}_0, \{\vec{S}_i\}_{i\in[\ell_s]})$. Let $h_e := H_e(\check{c}, \varrho_0, \sigma)$. Choose $r \xleftarrow{\text{U}} \mathbb{Z}_N$, $R_0 \xleftarrow{\text{U}} \mathbb{G}_{p_3}$. First create a normal key

$$\mathcal{SK}_A \quad := [A,\ K := g^{\alpha+at}R,\ L := g^tR'_0,\ K_i := T_i^{\ t}R_i,\ \forall i \in A]$$

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

$$\mathcal{USK}_A := [A,\ K := g^{\alpha+at}R\boxed{(u_e^{h_e}v_e)^r},\ \boxed{K_0 := g^rR_0},\ L := g^tR'_0,\ K_i := T_i^{\ t}R_i,\ \forall i \in A]$$

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

Semi-functional Type 1 Unsigncryption Query Key. Choose $b, d \xleftarrow{\text{U}} \mathbb{Z}_N$. First create a normal unsigncryption query key as below

$$\mathcal{USK}_A := [A,\ K := g^{\alpha+at}R(u_e^{h_e}v_e)^r,\ K_0 := g^rR_0,\ L := g^tR'_0,\ K_i := T_i^{\ t}R_i,\ \forall i \in A]$$

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

$$\mathcal{USK}_A := [A,\ K := g^{\alpha+at}R(u_e^{h_e}v_e)^r\boxed{g_2^d},\ K_0 := g^rR_0\boxed{g_2^b},\ L := g^tR'_0,\ K_i := T_i^{\ t}R_i\ \forall i \in A]$$

Semi-functional Type 2 Unsigncryption Query Key. This is same as sf-type 1 unsigncryption query key except $b = 0$, i.e.,

$$\mathcal{USK}_A := [A,\ K := g^{\alpha+at}R(u_e^{h_e}v_e)^rg_2^d,\ K_0 := g^rR_0,\ L := g^tR'_0,\ K_i := T_i^{\ t}R_i\ \forall i \in A]$$

6.3 Answering Unsigncryption Query Using uq-key.

Let $\Upsilon := (\check{c}, \sigma, \varrho)$ be a queried signcryption for (Γ_e, Γ_s). Choose a set of attribute A such that $\Gamma_e(A) = \mathsf{True}$. First, create a uq-key $\mathcal{USK}_A := [A, K, K_0, L, K_i\ \forall i \in A]$ of desired type. Then, the query will be handled in a similar manner as in Unsigncrypt algorithm, except the Decrypt routine, viz., the computation Δ_e. Below is the required computation for Δ_e to answer unsigncryption query.

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

Finally it returns the message m if Υ is a valid signcryption else returns $\perp$ (to indicate invalidity of Υ).

A normal key can extract the message from a legitimate normal signcryption as well as sf-type 1 signcryption. But, if a 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 $\vec{v}_e$. A sf-type 1 key is said to be nominally semi-functional if $cd - bv_1 = 0$. In this case, a 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, H_e is a collision resistant hash function and Π_{Commit} has hiding property, then our proposed basic CP-ABSC scheme in section 5 is adaptively secure.*

The proof is described in Appendix C.4.

Proof Sketch of Theorem 6.2

Suppose there are at most ν_1 key queries, ν_2 unsigncryption queries and ν_3 signcryption queries made by an adversary $\mathscr{A}$, then by applying hybrid arguments over the sequence of games $\text{Game}_{Real}, \text{Game}_0,$

$\{\text{Game}_{1-k-1}, \text{Game}_{1-k-2}\}_{k\in[\nu_1]}$, $\{\text{Game}_{2-k-1}, \text{Game}_{2-k-2}\}_{k\in[\nu_2]}$, $\{\text{Game}_{3-k-1}, \text{Game}_{3-k-2}\}_{k\in[\nu_3]}$, Game_4, Game_5 and Game_{Final}, the game Game_{Real} is changed to Game_{Final}.

In Game_0, the challenge signcryption is changed from normal to sf-type 1. In Game_{1-k-1} (for $1 \leq k \leq \nu_1$), the challenge signcryption is of sf-type 1, the unsigncryption queries are answered by normal uq-keys, the replied signcryptions are normal, the first $(k-1)$ keys are sf-type 2, k^{th} key is sf-type 1 and the rest are normal. Game_{1-k-2} (for $1 \leq k \leq \nu_1$) is same as Game_{1-k-1} except that k^{th} key is sf-type 2. In Game_{2-k-1} (for $1 \leq k \leq \nu_2$), the challenge signcryption is of sf-type 1, the replied signcryptions are normal, the keys are sf-type 2, the first $(k-1)$ unsigncryption queries are handled by sf-type 2 uq-keys, k^{th} unsigncryption query is answered by sf-type 1 uq-key and the rest are handled by normal uq-keys. Game_{2-k-2} (for $1 \leq k \leq \nu_2$) is same as Game_{2-k-1} except that k^{th} unsigncryption query is answered by sf-type 2 uq-key. In Game_{3-k-1} (for $1 \leq k \leq \nu_3$), the challenge signcryption is of sf-type 1, the keys are sf-type 2, the unsigncryption queries are handled by sf-type 2 uq-keys, the first $(k-1)$ replied signcryptions are of sf-type II, k^{th} replied signcryption is of sf-type I and the rest replied signcryptions are normal. Game_{3-k-2} (for $1 \leq k \leq \nu_3$) is same as Game_{3-k-1} except that k^{th} replied signcryption is of sf-type II. Game_4 is similar to $\text{Game}_{3-\nu_2-2}$ except, that the challenge signcryption is of sf-type 2. Game_5 is similar to Game_4 except, that the challenge signcryption is of sf-type 3. Game_{Final} is similar to Game_5 except, that the challenge signcryption is of sf-type 4. We prove that the gap advantage between any two consecutive games is at most negligible.

In lemma C.2, we show that the advantage gap between Game_{Real} and Game_0 is equivalent to that of DSG1: we establish a PPT simulator $\mathscr{B}$ for Game_{Real} and Game_0 against a PPT adversary $\mathscr{A}$. The simulator $\mathscr{B}$ takes an instance of DSG1 (with $\beta \xleftarrow{\text{U}} \{0,1\}$) and simulates either Game_{Real} or Game_0 for adversary $\mathscr{A}$. We show that the distribution of challenge signcryption, keys and uq-keys computed by $\mathscr{B}$ is equivalent to Game_{Real} (resp. Game_0) if $\beta = 0$ (resp. $\beta = 1$). Seemingly, this shows that the normal challenge signcryption and sf-type 1 challenge signcryption are indistinguishable under DSG1 assumption.

Similarly, in lemma C.3, we show that the advantage gap between $\text{Game}_{1-(k-1)-2}$ and Game_{1-k-1} (for $1 \leq k \leq \nu_1$) is bounded by the advantage of DSG2 adversary. In other words, it shows that the k^{th} normal key and k^{th} sf-type 1 key are indistinguishable if DSG2 assumption holds.

In a similar manner, we show that the advantage gap between Game_{1-k-1} and Game_{1-k-2} (for $1 \leq k \leq \nu_1$) is bounded by the advantage of DSG2 adversary (lemma C.4). Seemingly, it shows that the k^{th} sf-type 1 key and k^{th} sf-type 2 key are indistinguishable if DSG2 assumption holds.

Similarly, in lemma C.5, we show that the advantage gap between $\text{Game}_{2-(k-1)-2}$ and Game_{2-k-1} (for $1 \leq k \leq \nu_2$) is bounded by the advantage of DSG2 adversary. In other words, it shows that the k^{th} normal uq-key and k^{th} sf-type 1 uq-key are indistinguishable if DSG2 assumption holds.

In a similar manner, we show that the advantage gap between Game_{2-k-1} and Game_{2-k-2} (for $1 \leq k \leq \nu_2$) is bounded by the advantage of DSG2 adversary (lemma C.7). Seemingly, it shows that the k^{th} sf-type 1 uq-key and k^{th} sf-type 2 uq-key are indistinguishable if DSG2 assumption holds.

Similarly, in lemma C.8, we show that the advantage gap between $\text{Game}_{3-(k-1)-2}$ and Game_{3-k-1} (for $1 \leq k \leq \nu_3$) is bounded by the advantage of DSG2 adversary. In other words, it shows that the k^{th} normal signcryption and k^{th} sf-type I signcryption are indistinguishable if DSG2 assumption holds.

In a similar manner, we show that the advantage gap between Game_{3-k-1} and Game_{3-k-2} (for $1 \leq k \leq \nu_3$) is bounded by the advantage of DSG2 adversary (lemma C.9). Seemingly, it shows that the k^{th} sf-type I signcryption and k^{th} sf-type II signcryption are indistinguishable if DSG2 assumption holds.

In lemma C.10, we show that the advantage gap between $\text{Game}_{3-\nu_2-2}$ and Game_4 is bounded by the advantage of DSG2 adversary. In other words, it shows that the sf-type 1 challenge signcryption and the sf-type 2 challenge signcryption are indistinguishable if DSG2 assumption holds.

Similar, in lemma C.11, we show that the advantage gap between Game$_4$ and Game$_5$ is bounded by the advantage of DSG2 adversary. In other words, it shows that the sf-type 2 challenge signcryption and the sf-type 3 challenge signcryption are indistinguishable if DSG2 assumption holds.

In lemma C.12, we show that the advantage gap between Game$_5$ and Game$_{Final}$ is bounded by the advantage of DSG3 adversary. In other words, it shows that the sf-type 3 challenge signcryption and the sf-type 4 challenge signcryption are indistinguishable if DSG3 assumption holds.

Finally in lemma C.13, we show that the adversary has no advantage in Game$_{Final}$ if the primitive commitment scheme has the hiding property.

6.4 Adaptive-Predicates Existential Unforgeability of CP-ABSC

Theorem 6.3. *If DSG1, DSG2 and DSG3 assumptions hold for $\mathcal{J}$, the primitive commitment scheme Π_{Commit} has relaxed-binding property and H_s is a collision resistant hash function, then the proposed basic CP-ABSC scheme in section 5 is adaptive-predicates existential unforgeable.*

Proof. It follows from Theorem 6.4 and Theorem 4.2. □

Theorem 6.4. *The proposed basic CP-ABSC scheme in section 5 is adaptive-predicates existential unforgeable if primitive ABS scheme Π_{ABS} is adaptive-predicate existential unforgeable and the primitive commitment scheme Π_{Commit} has relaxed-binding property.*

The proof is given in Appendix C.5.

7 Strongly Unforgeable and IND-CCA2 Secure CP-ABSC

Here in this section, we describe our strongly unforgeable and IND-CCA2 secure CP-ABSC scheme for access policies represented by the monotone span programs. This scheme follows almost the same structure of weak unforgeable and IND-CCA2 secure CP-ABSC described in section 5. But to protect the signcryption from forging, we bind all the components by a strongly unforgeable OTS scheme which we call "Commit then Encrypt and Sign then Sign" ($\mathcal{C}t\mathcal{E}\&\mathcal{S}t\mathcal{S}$) paradigm. Although the similar type of generic constructions using strongly unforgeable OTS scheme are available in the literature [CHK04, HWZ07] in the context of ABE and ABS, here we do not apply the OTS scheme in straightforward way because of the following reasons: (a) we no more assume the relaxed-binding property of the commitment scheme for strong unforgeability, and (b) to reuse the part of IND-CCA2 security proof of the construction described in section 5 for the current CP-ABSC construction.

We just give a short description of our strongly unforgeable and IND-CCA2 secure CP-ABSC construction, since it follows from the CP-ABSC in section 5 and the idea of strongly unforgeable CP-ABS stated above. Let $\Pi_{\mathsf{Commit}} := (\mathsf{C.Setup}, \mathsf{Commit}, \mathsf{Open})$ be a commitment scheme with hiding property (relaxed-binding property is not required). Let $\Pi_{\mathsf{wABS}} := (\mathsf{wABS.Setup}, \mathsf{wABS.KeyGen}, \mathsf{wABS.Sign}, \mathsf{wABS.Ver})$ and $\Pi_{\mathsf{ABE}} := (\mathsf{ABE.Setup}, \mathsf{ABE.KeyGen}, \mathsf{ABE.Encrypt}, \mathsf{ABE.Decrypt})$ be the CP-ABS scheme and CP-ABE scheme respectively used in section 5. Let $\Pi_{\mathsf{OTS}} := (\mathsf{Gen}, \mathsf{OTS.Sign}, \mathsf{OTS.Ver})$ be a strong unforgeable one-time signature scheme. Demonstrated below are only two routines, Signcrypt and Unsigncrypt as rest are same as in section 5.

– $\mathsf{Signcrypt}(m, \mathcal{SK}_A, \Gamma_s, \Gamma_e) := \left(\begin{array}{c} (\breve{c}, \breve{d}) := \mathsf{Commit}(m) \parallel (\mathsf{verk}, \mathsf{signk}) := \mathsf{Gen}(1^\kappa), \\ \sigma_w := \mathsf{wABS.Sign}(\mathsf{verk}, \mathcal{SK}_A, \Gamma_s) \parallel \varrho_0 := \mathsf{ABE.Encrypt}(\breve{d}, \Gamma_e), \\ \text{where } \sigma_w := (\vec{S}_0, \ldots, \vec{S}_{\ell_s}) \text{ and } \varrho_0 := (\vec{C}_0, \ldots, \vec{C}_{\ell_e}), \\ \text{let } h_e := H_e(\breve{c}, \varrho_0, \sigma_w), C_{\ell_e+1} \longleftarrow \mathsf{fun}(\mathcal{PP}, h_e, s_e), \\ \sigma_o := \mathsf{OTS.Sign}(h_e || C_{\ell_e+1} || \Gamma_e || \Gamma_s, \mathsf{signk}), \\ \text{returns } \Upsilon := (\breve{c}, \sigma_s := (\sigma_w, \sigma_o, \mathsf{verk}), \varrho := (\varrho_0, C_{\ell_e+1})) \end{array}\right)$

– $\mathsf{Unsigncrypt}(\Upsilon, \mathcal{SK}_B, \Gamma_s) := \begin{cases} \mathsf{Open}(\breve{c}, \breve{d}) & \text{if} \left(\begin{array}{c} \mathsf{OTS.Ver}(h_e || C_{\ell_e+1} || \Gamma_e || \Gamma_s, \sigma_o, \mathsf{verk}) = 1 \parallel \\ \mathsf{wABS.Ver}(\mathsf{verk}, \sigma_w, \Gamma_s) = 1 \parallel \\ \text{let } \breve{d} := \mathsf{ABE.Decrypt}(\varrho, \mathcal{SK}_B, \Gamma_e), \text{ where} \\ \Upsilon := (\breve{c}, \sigma_s := (\sigma_w, \sigma_o, \mathsf{verk}), \varrho := (\varrho_0, C_{\ell_e+1})) \end{array}\right) \\ \perp & \text{otherwise.} \end{cases}$

Correctness. It follows from that of section 5.

Remark 7.1. Similar to CP-ABSC scheme in section 5, this construction achieves the dynamic property and non-repudiation.

Theorem 7.1. *The proposed CP-ABSC scheme in section 7 is perfectly private.*

Proof. It is similar to Theorem 6.1. □

Theorem 7.2. *If DSG1, DSG2 and DSG3 assumptions hold, H_e is a collision resistant hash function, Π_{Commit} has hiding property and Π_{OTS} is a strong unforgeable one-time signature scheme, then our proposed CP-ABSC scheme in section 7 is adaptively secure.*

Proof. The proof can be obtained by the similar approach as in proof of Theorem 6.2 and the argument used for proving CCA2 security in [CHK04]. □

Theorem 7.3. *If DSG1, DSG2 and DSG3 assumptions hold for $\mathcal{J}$, Π_{OTS} is a strong unforgeable one-time signature scheme and H_s, H_e are collision resistant hash functions, then the proposed basic CP-ABSC scheme in section 7 is strong existential unforgeable.*

Proof. It is straightforward from Theorem 7.4 and Theorem 4.2. □

Theorem 7.4. *The proposed basic CP-ABSC scheme in section 7 is adaptive-predicates strong existential unforgeable if primitive ABS scheme Π_{wABS} is adaptive-predicate weak existential unforgeable, Π_{OTS} is a strong unforgeable one-time signature scheme and H_e are collision resistant hash function.*

The proof is given in Appendix C.6.

8 Conclusion and Future Work

We have presented an ABSC scheme in $\mathcal{C}t\mathcal{E}\&\mathcal{S}t\mathcal{S}$ paradigm using our proposed adaptive-predicate unforgeable ABS and an adaptive-predicate IND-CCA secure ABE from the literature. The scheme is supported with the combined setup to reduce the size of the public parameter and key. Since the subroutines of ABS and ABE run in parallel in Signcrypt and Unsigncrypt, the execution of the scheme seems to be faster. To best of our knowledge, this is the first ABSC scheme that has been shown secure in strong sense in

Adaptive-predicates model. The scheme also has other features, signer-privacy, non-repudiation etc. In future, we would be interesting to construct the generic ABSC scheme from any ABS and ABE in combined setup.

References

[ADR02] Jee Hea An, Yevgeniy Dodis, and Tal Rabin. On the security of joint signature and encryption. In *EUROCRYPT*, volume 2332 of *LNCS*, pages 83–107. Springer, 2002.

[ALdP11] Nuttapong Attrapadung, Benoît Libert, and Elie de Panafieu. Expressive key-policy attribute-based encryption with constant-size ciphertexts. In *Public Key Cryptography*, volume 6571 of *LNCS*, pages 90–108. Springer, 2011.

[BB04] Dan Boneh and Xavier Boyen. Efficient selective-ID secure identity-based encryption without random oracles. In *EUROCRYPT*, pages 223–238, 2004.

[Bei96] Amos Beimel. *Secure schemes for secret sharing and key distribution.* PhD thesis, Israel Institute of Technology, Technion, Haifa, Israel, 1996.

[BF01] Dan Boneh and Matt Franklin. Identity-based encryption from the weil pairing. In *CRYPTO*, volume 2139 of *LNCS*, pages 213–229. Springer, 2001.

[BK05] Dan Boneh and Jonathan Katz. Improved efficiency for CCA-secure cryptosystems built using identity-based encryption. In Alfred Menezes, editor, *CT-RSA*, volume 3376 of *Lecture Notes in Computer Science*, pages 87–103. Springer, 2005.

[Boy03] Xavier Boyen. Multipurpose identity-based signcryption. In *CRYPTO*, volume 2729 of *LNCS*, pages 383–399. Springer, 2003.

[BSW07] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *IEEE Symposium on Security and Privacy*, pages 321–334. IEEE Press, 2007.

[BSZ02] Joonsang Baek, Ron Steinfeld, and Yuliang Zheng. Formal proofs for the security of signcryption. In *PKC*, volume 2274 of *LNCS*, pages 80–98. Springer, 2002.

[CCL$^+$12] Cheng Chen, Jie Chen, Hoon Wei Lim, Zhenfeng Zhang, and Dengguo Feng. Combined public-key schemes: The case of ABE and ABS. In *PROVSEC*, volume 7496 of *LNCS*, pages 53–69. Springer, 2012.

[CHK04] Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. In *EUROCRYPT*, pages 207–222, 2004.

[CML05] Liqun Chen and John Malone-Lee. Improved identity-based signcryption. In *PKC*, volume 3386 of *LNCS*, pages 362–379. Springer, 2005.

[DF02] Ivan Damgrd and Eiichiro Fujisaki. A statistically-hiding integer commitment scheme based on groups with hidden order. In *ASIACRYPT*, volume 2501 of *LNCS*, pages 125–142. Springer, 2002.

[DFMS10] Alexander W. Dent, Marc Fischlin, Mark Manulis, and Dominique Schrder. Confidential signatures and deterministic signcryption. In *PKC*, volume 6056 of *LNCS*, pages 462–479. Springer, 2010.

[EMR12] Keita Emura, Atsuko Miyaji, and Mohammad Shahriar Rahman. Dynamic attributebased signcryption without random oracles. *International Journal of Applied Cryptography*, 2(11):199–211, 2012.

[GNSN10] Martin Gagné, Shivaramakrishnan Narayan, and Reihaneh Safavi-Naini. Threshold attribute-based signcryption. In *SCN*, volume 6280 of *LNCS*, pages 154–171. Springer, 2010.

[GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *ACM Conference on Computer and Communications Security*, pages 89–98. ACM, 2006.

[Hes02] Florian Hess. Efficient identity based signature schemes based on pairings. In *SAC*, volume 2595 of *LNCS*, pages 310–324. Springer, 2002.

[HM96] Shai Halevi and Silvio Micali. Practical and provably-secure commitment schemes from collision-free hashing. In *CRYPTO*, volume 1109 of *LNCS*, pages 201–215. Springer, 1996.

[HP01] Stuart Haber and Benny Pinkas. Securely combining public-key cryptosystems. In *ACM Conference on Computer and Communications Security*, pages 215–224. ACM, 2001.

[HWZ07] Qiong Huang, Duncan S. Wong, and Yiming Zhao. Generic transformation to strongly unforgeable signatures. In *ACNS*, volume 4521 of *LNCS*, pages 1–17. Springer, 2007.

[LAS$^+$10] Jin Li, Man Ho Au, Willy Susilo, Dongqing Xie, and Kui Ren. Attribute-based signature and its applications. In *ACM Conference on Computer and Communications Security*, pages 60–69. ACM, 2010.

[LOS$^+$10] Allison B. Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In *EUROCRYPT*, volume 6110 of *LNCS*, pages 62–91. Springer, 2010.

[LQ04a] Benoît Libert and Jean-Jacques Quisquater. Efficient signcryption with key privacy from gap diffie-hellman groups. In *PKC*, volume 2947 of *LNCS*, pages 187–200. Springer, 2004.

[LQ04b] Benoît Libert and Jean-Jacques Quisquater. Improved signcryption from q-diffie-hellman problems. In *SCN*, volume 3352 of *LNCS*, pages 220–234. Springer, 2004.

[LW10] Allison B. Lewko and Brent Waters. New techniques for dual system encryption and fully secure HIBE with short ciphertexts. In Daniele Micciancio, editor, *TCC*, volume 5978 of *Lecture Notes in Computer Science*, pages 455–479. Springer, 2010.

[LW12] Allison Lewko and Brent Waters. New proof methods for attribute-based encryption: Achieving full security through selective techniques. In *CRYPTO*, volume 7417 of *LNCS*, pages 180–198. Springer, 2012.

[MLM03] John Malone-Lee and Wenbo Mao. Two birds one stone: Signcryption using RSA. In *CT-RSA*, volume 2612 of *LNCS*, pages 211–226. Springer, 2003.

[MMS09] Takahiro Matsuda, Kanta Matsuura, and Jacob C. N. Schuldt. Efficient constructions of signcryption schemes and signcryption composability. In *INDOCRYPT*, volume 5922 of *LNCS*, pages 321–342. Springer, 2009.

[MPR08] Hemanta Maji, Manoj Prabhakaran, and Mike Rosulek. Attribute-based signatures: Achieving attribute-privacy and collusion-resistance. Cryptology ePrint Archive, Report 2008/328, 2008. `http://eprint.iacr.org/`.

[MPR10] Hemanta Maji, Manoj Prabhakaran, and Mike Rosulek. Attribute-based signatures. Cryptology ePrint Archive, Report 2010/595, 2010. `http://eprint.iacr.org/`.

[NP15] Mridul Nandi and Tapas Pandit. Generic conversions from cpa to cca secure functional encryption. Cryptology ePrint Archive, Report 2015/457, 2015. `http://eprint.iacr.org/`.

[OSW07] Rafail Ostrovsky, Amit Sahai, and Brent Waters. Attribute-based encryption with non-monotonic access structures. In *ACM Conference on Computer and Communications Security*, pages 195–203, 2007.

[OT10] Tatsuaki Okamoto and Katsuyuki Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In *CRYPTO*, volume 6223 of *LNCS*, pages 191–208. Springer, 2010.

[OT11] Tatsuaki Okamoto and Katsuyuki Takashima. Efficient attribute-based signatures for non-monotone predicates in the standard model. In *Public Key Cryptography*, volume 6571 of *LNCS*, pages 35–52. Springer, 2011.

[OT12] Tatsuaki Okamoto and Katsuyuki Takashima. Fully secure unbounded inner-product and attribute-based encryption. In *ASIACRYPT*, volume 7658 of *LNCS*, pages 349–366. Springer, 2012.

[Ped91] Torben Pryds Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *CRYPTO*, volume 576 of *LNCS*, pages 129–140. Springer, 1991.

[PSST11] Kenneth G. Paterson, Jacob C. N. Schuldt, Martijn Stam, and Susan Thomson. On the joint security of encryption and signature, revisited. In *ASIACRYPT*, volume 7073 of *LNCS*, pages 161–178. Springer, 2011.

[RD13] Y. Sreenivasa Rao and Ratna Dutta. Computationally efficient expressive key-policy attribute based encryption schemes with constant-size ciphertext. In *ICICS*, volume 8233 of *LNCS*, pages 346–362. Springer, 2013.

[RD14] Y. Sreenivasa Rao and Ratna Dutta. Expressive bandwidth-efficient attribute based signature and signcryption in standard model. In *ACISP*, volume 8544 of *LNCS*, pages 209–225. Springer, 2014.

[SSN09] Siamak F Shahandashti and Reihaneh Safavi-Naini. Threshold attribute-based signatures and their application to anonymous credential systems. In *AFRICACRYPT*, volume 5580 of *LNCS*, pages 198–216. Springer, 2009.

[SW05] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *EUROCRYPT*, volume 3494 of *LNCS*, pages 457–473. Springer, 2005.

[VHS08] Maria Isabel Gonzalez Vasco, Florian Hess, and Rainer Steinwandt. Combined (identity-based) public key schemes. Cryptology ePrint Archive, Report 2008/466, 2008. `http://eprint.iacr.org/`.

[Wat09] Brent Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In Shai Halevi, editor, *CRYPTO*, volume 5677 of *Lecture Notes in Computer Science*, pages 619–636. Springer, 2009.

[Wat11] Brent Waters. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In *Public Key Cryptography*, volume 6571 of *LNCS*, pages 53–70. Springer, 2011.

[YAHK11] Shota Yamada, Nuttapong Attrapadung, Goichiro Hanaoka, and Noboru Kunihiro. Generic constructions for chosen-ciphertext secure attribute based encryption. In *Public Key Cryptography*, volume 6571 of *LNCS*, pages 71–89. Springer, 2011.

[Zhe97] Yuliang Zheng. Digital signcryption or how to achieve cost(signature & encryption) $\ll$ cost(signature) + cost(encryption). In *CRYPTO*, volume 1294 of *LNCS*, pages 165–179. Springer, 1997.

A Strongly Unforgeable One-Time Signature

A.1 Definition (Signature)

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

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

A.2 Strong Unforgeability of One-Time Signature

Strongly unforgeability one-Time signature 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 $\mathsf{Gen}(1^\kappa) \longrightarrow (\mathsf{verk}, \mathsf{signk})$. Then verk 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 $\mathsf{Ver}(m^*, \sigma^*, \mathsf{verk}) = 1$ and $(m, \sigma) \neq (m^*, \sigma^*)$.

Let $\mathsf{Adv}_{\mathscr{A}}^{\mathrm{OTS}}(\kappa)$ 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* if $\mathsf{Adv}_{\mathscr{A}}^{\mathrm{OTS}}(\kappa)$ is at most negligible function in κ

B Ciphertext-Policy Attribute-Base Signature

B.1 Definition

A ciphertext-policy attribute-base Signature scheme consists of four PPT algorithms - Setup, KeyGen, Sign and Ver.

- Setup: It takes a security parameter κ and a universe of attributes $\mathcal{U}$ as input, outputs the public parameters $\mathcal{PP}$ and the master secret $\mathcal{MSK}$.
- KeyGen: It takes as input a set of attributes A, public parameters $\mathcal{PP}$ and master secret $\mathcal{MSK}$ and outputs a secret key $\mathcal{SK}_A$ corresponding to A.
- Sign: takes a message m, a secret key $\mathcal{SK}_A$, a predicate Γ with $\Gamma(A) = \mathsf{True}$ and public parameters $\mathcal{PP}$ and returns a signature σ which implicitly contains Γ.
- Ver: It receives a message m, a signature σ, a claim predicate Γ and public parameters $\mathcal{PP}$ as input. It returns a boolean value 1 for accept or 0 for reject.

Correctness. For all $(\mathcal{PP}, \mathcal{MSK}) \longleftarrow \mathsf{Setup}$, all messages $m \in \mathcal{M}$, all attribute sets A, all keys $\mathcal{SK}_A \longleftarrow \mathsf{KeyGen}(\mathcal{PP}, \mathcal{MSK}, A)$ and all policies Γ with $\Gamma(A) = \mathsf{True}$, it is required that $\mathsf{Ver}(\mathcal{PP}, m, \mathsf{Sign}(\mathcal{PP}, m, \mathcal{SK}_A, \Gamma), \Gamma) = 1$.

Definition B.1 (Signer Privacy)**.** An ABS scheme is said to be perfectly private if for all $(\mathcal{PP}, \mathcal{MSK}) \longleftarrow \mathsf{Setup}$, all attribute sets A_1, A_2, all keys $\mathcal{SK}_{A_1} \longleftarrow \mathsf{KeyGen}(\mathcal{PP}, \mathcal{MSK}, A_1)$, $\mathcal{SK}_{A_2} \longleftarrow \mathsf{KeyGen}(\mathcal{PP}, \mathcal{MSK}, A_2)$, all messages $m \in \mathcal{M}$, and all claim-predicates $\Gamma_s := (M_s, \rho_s)$ such that $\Gamma_s(A_1) = \Gamma_s(A_2) = \mathsf{True}$, the distributions of $\mathsf{Sign}(\mathcal{PP}, m, \mathcal{SK}_{A_1}, \Gamma_s)$ and $\mathsf{Sign}(\mathcal{PP}, m, \mathcal{SK}_{A_2}, \Gamma_s)$ are identical.

We define an alternative signature oracle, $\mathsf{AltSign}(\mathcal{PP}, m, \mathcal{MSK}, \Gamma)$: it first produces a secret key $\mathcal{SK}_A \longleftarrow \mathsf{KeyGen}(\mathcal{PP}, \mathcal{MSK}, A)$ for a set of attributes A such that $\Gamma(A) = \mathsf{True}$ and then, runs $\sigma \longleftarrow \mathsf{Sign}(\mathcal{PP}, m, \mathcal{SK}_A, \Gamma)$. For an ABS scheme with signer-privacy, $\mathsf{AltSign}(\mathcal{PP}, m, \mathcal{MSK}, \Gamma)$ and $\mathsf{Sign}(\mathcal{PP}, m, \mathcal{SK}_A, \Gamma)$ are identical for all A with $\Gamma(A) = \mathsf{True}$. Therefore, we may replace the Sign oracle by $\mathsf{AltSign}$ oracle for an ABS with signer-privacy whenever it requires.

B.2 Adaptive-Predicate Existential Unforgeability of CP-ABS

The adaptive-predicate existential unforgeable security model is defined as a game between a challenger $\mathscr{B}$ and an adversary $\mathscr{A}$, where the adversary has to forge a signature for a message (consist of plaintext and predicate). It consists of the following phases:

Setup: The challenger $\mathscr{B}$ runs the Setup algorithm to produce the master secret key $\mathcal{MSK}$ and the public parameter $\mathcal{PP}$. Then, $\mathscr{B}$ gives $\mathcal{PP}$ to the adversary $\mathscr{A}$ and keeps $\mathcal{MSK}$ to itself.

Query: The adversary $\mathscr{A}$ is given access to the oracles $\mathsf{KeyGen}(\mathcal{PP}, \mathcal{MSK}, .)$ and $\mathsf{Sign}(\mathcal{PP}, ., .)$

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

We say the adversary succeeds in this game if (m^*, Γ^*) was never queried to Sign oracle, Γ^* does not accept any set of attributes queried to KeyGen oracle and $\mathsf{Ver}(\mathcal{PP}, m^*, \Gamma^*, \sigma^*) = 1$.

Let $\mathsf{Adv}_{\mathscr{A}}^{\mathrm{ABS-EUF}}(\kappa)$ denote the success probability for any adversary $\mathscr{A}$ in the above experiment. An ABS scheme is said to be *adaptive-predicate existential unforgeable* (AP-UF-CMA) if $\mathsf{Adv}_{\mathscr{A}}^{\mathrm{ABS-EUF}}(\kappa)$ is at most negligible function in κ

Remark B.1. The above unforgeability 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* (AP-sUF-CMA), the adversary $\mathscr{A}$ may forge σ^* for a queried message pair (m^*, Γ^*) but the replied signature σ on (m^*, Γ^*) must be different from the forge signature σ^*.

Remark B.2. There is an another variant of unforgeability, called *selective-predicate unforgeability* in both weak and strong sense, where $\mathscr{A}$ submits a challenge policy Γ (later on which it will forge) before obtaining the $\mathcal{PP}$ of ABS.

B.3 Adaptive-Predicate Existential Unforgeability of CP-ABS

Since, the CP-ABS has perfect privacy, we replace through out the proofs the actual Sign algorithm by an alternative algorithm, $\mathsf{AltSign}$ defined in section B. For notational simplicity, we do not use the term "AltSignt" in the proofs. We note that all the Sign oracles appeared in the proofs are basically $\mathsf{AltSign}$ oracles. We prove the adaptive-predicate unforgeability of our basic ABS scheme by the proof technique of Okamoto–Takashima [OT11] and the dual system methodology of Brent Waters [Wat09]. This methodology requires to define 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. Consequently, 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. Also, we use the abbreviation 'sf' and 'vText' for 'semi-functional' and 'verification text' respectively.

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

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

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

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

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

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

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

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

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

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

$$\mathcal{SK}_A := [A,\ K := g^{\alpha+at}Rg_2^d,\ L := g^t R_0',\ K_i := T_i{}^t R_i,\ \forall i \in A]$$

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 shown below in the boxes.

$$\vec{S}_0 := \big(\ g^{\alpha+a\tilde{t}}(u_s^{h_s}v_s)^{r_s}\tilde{R}\boxed{g_2^{\tilde{d}}},\ g^{r_s}\tilde{R}_0\boxed{g_2^{\tilde{b}}}\ \big)$$
$$\vec{S}_i := \big(\ (g^{\tilde{t}})^{\alpha_s^{(i)}}(g^{\tau})^{\beta_i}\tilde{R}_i,\qquad (T_{\rho_s(i)}^{\tilde{t}})^{\alpha_s^{(i)}}(T_{\rho_s(i)}^{\tau})^{\beta_i}\tilde{R}'{}_i\ \big)$$

Semi-functional Type 2 Signature. This is same as sf-type 1 signature except $\tilde{b} = 0$, i.e.,

$$\vec{S}_0 := \big(\ g^{\alpha+a\tilde{t}}(u_s^{h_s}v_s)^{r_s}\tilde{R}g_2^{\tilde{d}},\ g^{r_s}\tilde{R}_0\ \big)$$
$$\vec{S}_i := \big(\ (g^{\tilde{t}})^{\alpha_s^{(i)}}(g^{\tau})^{\beta_i}\tilde{R}_i,\quad (T_{\rho_s(i)}^{\tilde{t}})^{\alpha_s^{(i)}}(T_{\rho_s(i)}^{\tau})^{\beta_i}\tilde{R}'{}_i\ \big)$$

A normal signature can be verified by a normal vText as well as a sf-type 1 vText. But, if a valid sf-type 1 (resp. sf-type 2) signature is verified by a 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 B.1. *The proposed basic CP-ABS scheme is adaptive-predicate existential unforgeable if DSG1, DSG2 and DSG3 assumptions hold 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. The games are defined below:

- Game$_{Real}$:= The original AP-UF-CMA security game of CP-ABS.
- Game$_0$ (= Game$_{1-0-2}$) is just like Game$_{Real}$ except that the vText is of sf-type 1.
- In Game$_{1-k-1}$ (for $1 \le k \le \nu_1$), vText is sf-type 1, all the queried signatures are normal, the first $(k-1)$ keys returned to the adversary are sf-type 2, k^{th} key is sf-type 1 and the rest keys are normal.
- 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.
- In Game$_{2-k-1}$ (for $1 \le k \le \nu_2$), vText is sf-type 1, all the queried keys are sf-type 2, the first $(k-1)$ signatures returned to the adversary are sf-type 2, k^{th} signature is sf-type 1 and the rest signatures are normal. (So, in this sequel Game$_{2-0-2}$ = 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 Game$_{2-\nu_2-2}$ except that the vText is of sf-type 2.

Let $\mathsf{Adv}^{\mathrm{Real}}_{\mathscr{A},\mathrm{ABS}}(\kappa)$, $\mathsf{Adv}^{0}_{\mathscr{A},\mathrm{ABS}}(\kappa)$, $\mathsf{Adv}^{1-k-1}_{\mathscr{A},\mathrm{ABS}}(\kappa)$, $\mathsf{Adv}^{1-k-2}_{\mathscr{A},\mathrm{ABS}}(\kappa)$, $\mathsf{Adv}^{2-i-1}_{\mathscr{A},\mathrm{ABS}}(\kappa)$, $\mathsf{Adv}^{2-i-2}_{\mathscr{A},\mathrm{ABS}}(\kappa)$ and $\mathsf{Adv}^{\mathrm{Final}}_{\mathscr{A},\mathrm{ABS}}(\kappa)$ denote the advantages of an adversary $\mathscr{A}$ in Game_{Real}, Game_0, Game_{1-k-1}, Game_{1-k-2}, Game_{2-i-1}, Game_{2-i-2} and Game_{Final} for $1 \le k \le \nu_1$, $1 \le i \le \nu_2$ respectively. Let $\mathsf{Adv}^{CR-H_s}_{\mathscr{B}}(\kappa)$ be the advantage of $\mathscr{B}$ in breaking collision resistant property of H_s. In Game_{Final}, the part, V_{03} in $\vec{V}_0$ is chosen independently and uniformly random from $\mathbb{G}_T$ implying that the forgery will be invalid with respect to the vText. Therefore, the adversary $\mathscr{A}$ has no advantage in Game_{Final}.

Using lemmas B.2, B.3, B.5, B.6, B.7 and B.8, we have the following inequalities

$$\begin{aligned}
\mathsf{Adv}^{\mathrm{ABS-EUF}}_{\mathscr{A}}(\kappa) &= \mathsf{Adv}^{\mathrm{Real}}_{\mathscr{A},\mathrm{ABS}}(\kappa)\\
&\le |\mathsf{Adv}^{\mathrm{Real}}_{\mathscr{A},\mathrm{ABS}}(\kappa) - \mathsf{Adv}^{0}_{\mathscr{A},\mathrm{ABS}}(\kappa)|\\
&\quad + \sum_{k=1}^{\nu_1}(|\mathsf{Adv}^{1-(k-1)-2}_{\mathscr{A},\mathrm{ABS}}(\kappa) - \mathsf{Adv}^{1-k-1}_{\mathscr{A},\mathrm{ABS}}(\kappa)| + |\mathsf{Adv}^{1-k-1}_{\mathscr{A},\mathrm{ABS}}(\kappa) - \mathsf{Adv}^{1-k-2}_{\mathscr{A},\mathrm{ABS}}(\kappa)|)\\
&\quad + \sum_{i=1}^{\nu_2}(|\mathsf{Adv}^{2-(i-1)-2}_{\mathscr{A},\mathrm{ABS}}(\kappa) - \mathsf{Adv}^{2-i-1}_{\mathscr{A},\mathrm{ABS}}(\kappa)| + |\mathsf{Adv}^{2-i-1}_{\mathscr{A},\mathrm{ABS}}(\kappa) - \mathsf{Adv}^{2-i-2}_{\mathscr{A},\mathrm{ABS}}(\kappa)|)\\
&\quad + |\mathsf{Adv}^{2-\nu_2-2}_{\mathscr{A},\mathrm{ABS}}(\kappa) - \mathsf{Adv}^{\mathrm{Final}}_{\mathscr{A},\mathrm{ABS}}(\kappa)| + \mathsf{Adv}^{\mathrm{Final}}_{\mathscr{A},\mathrm{ABS}}(\kappa)\\
&\le \mathsf{Adv}^{\mathrm{DSG1}}_{\mathscr{B}_0}(\kappa) + \sum_{i=1}^{\nu_1}(\mathsf{Adv}^{\mathrm{DSG2}}_{\mathscr{B}_{1-i-1}}(\kappa) + \mathsf{Adv}^{\mathrm{DSG2}}_{\mathscr{B}_{1-i-2}}(\kappa))\\
&+ \sum_{i=1}^{\nu_2}(\mathsf{Adv}^{\mathrm{DSG2}}_{\mathscr{B}_{2-i-1}}(\kappa) + \mathsf{Adv}^{CR-H_s}_{\mathcal{H}_{2-i-1}}(\kappa) + \mathsf{Adv}^{\mathrm{DSG2}}_{\mathscr{B}_{2-i-2}}(\kappa)) + \mathsf{Adv}^{\mathrm{DSG3}}_{\mathscr{B}_3}(\kappa)
\end{aligned}$$

where $\mathscr{B}_0, \mathscr{B}_{1-i-1}, \mathscr{B}_{1-i-2}, \mathscr{B}_{2-i-1}, \mathcal{H}_{2-i-1}, \mathscr{B}_{2-i-2}$ and $\mathscr{B}_3$ are PPT algorithms whose running times are same as that of $\mathscr{A}$. This concludes the theorem. □

Lemma B.2. 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* $|\mathsf{Adv}^{\mathrm{Real}}_{\mathscr{A},\mathrm{ABS}}(\kappa) - \mathsf{Adv}^{0}_{\mathscr{A},\mathrm{ABS}}(\kappa)| \le \mathsf{Adv}^{\mathrm{DSG1}}_{\mathscr{B}}(\kappa)$.

Proof. We establish a PPT simulator $\mathscr{B}$ who receives an instance of DSG1, $(\mathcal{J}, g, X_3, T_\beta)$ with $\beta \xleftarrow{\mathrm{U}} \{0,1\}$ and depending on the distribution of β, it either simulates 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 $i \in \mathcal{U}$. Then, it sets $u_s := g^{a_s}, v_s := g^{b_s}$ and $T_i := g^{t_i}$ for $i \in \mathcal{U}$. $\mathscr{B}$ selects a hash function $H_s : \{0,1\}^* \longrightarrow \mathbb{Z}_N$. It provides $\mathcal{PP} := (\mathcal{J}, g, g^a, u_s, v_s, g_T^\alpha, \{T_i\}_{i\in\mathcal{U}}, X_3, H_s)$ to $\mathscr{A}$ and keeps $\mathcal{MSK} := (\alpha)$ to itself.

Key Query Answering: It is normal key. $\mathscr{B}$ can handle the key queries of $\mathscr{A}$, since the $\mathcal{MSK}$ is known to him.

Signature Query Answering: It is normal signature. $\mathscr{B}$ can answer the queries of $\mathscr{A}$, since he can construct any key using the $\mathcal{MSK}$ known to him.

Forgery: $\mathscr{A}$ outputs a signature σ^* for $(m^*, \Gamma_s^* := (M_s^*, \rho_s^*))$, where 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 $h_s^* := H_s(m^*||\Gamma^*)$. It selects $\vec{v_s'} := (1, v_2', \ldots, v_{n_s^*}')$, where $v_2', \ldots, v_{n_s^*}' \xleftarrow{\mathrm{U}} \mathbb{Z}_N$. It chooses $r_i' \xleftarrow{\mathrm{U}} \mathbb{Z}_N$ for $i \in [\ell_s^*]$. $\mathscr{B}$ implicitly sets g^s part to be the $\mathbb{G}_{p_1}$ component of T_β.

$$\vec{V}_0 := \big(\ T_\beta, T_\beta^{h_s^* a_s + b_s}, e(g^\alpha, T_\beta)\ \big)$$
$$\vec{V}_i := \big(\ T_\beta^{a(\vec{M}_s^{*(i)}.\vec{v}_s')} T_\beta^{-r_i' t_{\rho_s^*(i)}}, \quad T_\beta^{r_i'}\ \big), \text{ for } i \in [\ell_s^*]$$

The final vText is $\mathcal{V} := (\vec{V}_0, \{\vec{V}_i\}_{i\in[\ell_s^*]})$

$\mathscr{B}$ verifies the signature σ^* using the vText $\mathcal{V}$ and returns 1 if it passes the verification test else returns 0.

Analysis: $\mathscr{B}$ implicitly sets $\vec{u}_s := s\vec{v}'_s = (s, sv'_2, \ldots, sv'_{n_s^*})$ 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 $\vec{u}_s$ is a random vector over $\mathbb{Z}_{p_1}$. Similarly, since $r'_1, \ldots, r'_{\ell_s^*}$ are uniformly and independently distributed over $\mathbb{Z}_N$, so are $r_s^{(1)}, \ldots, r_s^{(\ell_s^*)}$ over $\mathbb{Z}_{p_1}$. Therefore, if $\beta = 0$, i.e., $T_\beta = g^s$, $\mathcal{V}$ is a properly distributed normal verification text.

Suppose $\beta = 1$, i.e., $T_\beta = g^s g_2^c$ for some $c \in \mathbb{Z}_N$. It implicitly sets $\vec{v}_s := ca\vec{v}'_s = (ca, cav'_2, \ldots, cav'_{n_s^*})$, $\iota := c(h_s^* a_s + b_s)$, $\gamma_s^{(i)} := -cr'_i$ 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 $i \in [\ell_s^*]$ modulo p_1 are uncorrelated from these values modulo p_2. Therefore, if $\beta = 1$, $\mathcal{V}$ is a properly distributed sf-type 1 verification text. □

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

Proof. $\mathscr{B}$ is given an instance of DSG2, $(\mathcal{J}, g, X_1X_2, Y_2Y_3, X_3, T_\beta)$ with $\beta \xleftarrow{\mathrm{U}} \{0,1\}$ and depending on the distribution of β, $\mathscr{B}$ either simulates Game$_{1-(k-1)-2}$ or Game$_{1-k-1}$.

Setup: Similar to Lemma B.2.

Key Query Answering: 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 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 $\mathcal{SK}_{A_j}$ 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 $t \xleftarrow{\mathrm{U}} \mathbb{Z}_N$, $R'_0, R_i \xleftarrow{\mathrm{U}} \mathbb{G}_{p3}$ for $i \in A_j$ and returns the following to $\mathscr{A}$.

  $\mathcal{SK}_{A_j} := [A_j,\ K := g^{\alpha+at}(Y_2Y_3)^t,\ L := g^t R'_0,\ K_i := T_i{}^t R_i,\ \forall i \in A_j]$

  Since, t modulo p_2 and t modulo p_3 are uncorrelated, so the key $\mathcal{SK}_{A_j}$ is properly distributed sf-type 2 key.
- If $j = k$ then it is either normal or sf-type 1 key. $\mathscr{B}$ generates $\mathcal{SK}_{A_k}$ using T_β of the instance of DSG2. $\mathscr{B}$ implicitly sets g^t part 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 $i \in A_k$

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

Signature Query Answering: It is normal signature. $\mathscr{B}$ can handle the queries of $\mathscr{A}$ using $\mathcal{MSK}$.

Forgery: $\mathscr{A}$ outputs a signature σ^* for $(m^*, \Gamma_s^* := (M_s^*, \rho_s^*))$, where 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 $h_s^* := H_s(m^*||\Gamma_s^*)$. It selects $\vec{v}'_s := (1, v'_2, \ldots, v'_{n_s^*})$, where $v'_2, \ldots, v'_{n_s^*} \xleftarrow{\mathrm{U}} \mathbb{Z}_N$.

$\vec{V}_0 := \big(\ X_1X_2,\ (X_1X_2)^{h_s^* a_s + b_s},\ e(g^\alpha, X_1X_2)\ \big)$

$\vec{V}_i := \big(\ (X_1X_2)^{a(\vec{M}_s^{*(i)}.\vec{v}'_s)}(X_1X_2)^{-r'_i t_{\rho_s^*(i)}},\quad (X_1X_2)^{r'_i}\ \big)$, for $i \in [\ell_s^*]$

The final vText is $\mathcal{V} := (\vec{V}_0, \{\vec{V}_i\}_{i\in[\ell_s^*]})$

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

Analysis: Let $X_1X_2 = g^s g_2^c$. $\mathscr{B}$ implicitly sets $\vec{u}_s := s\vec{v}'_s = (s, sv'_2, \ldots, sv'_{n_s^*})$ 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 $\vec{u}_s$ is a random vector over $\mathbb{Z}_{p_1}$. Similarly, since $r'_1, \ldots, r'_{\ell_s^*}$ are uniformly and independently distributed over $\mathbb{Z}_N$, so are $r_s^{(1)}, \ldots, r_s^{(\ell_s^*)}$ over $\mathbb{Z}_{p_1}$. It implicitly sets $\vec{v}_s := ca\vec{v}'_s = (ca, cav'_2, \ldots, cav'_{n_s^*})$, $\iota := c(h_s^* a_s + b_s)$, $\gamma_s^{(i)} := -cr'_i$ and $z_{\rho_s^*(i)} := t_{\rho_s^*(i)}$ for $i \in [\ell_s^*]$. By CRT, the values $v'_2, \ldots, v'_{n_s^*}$ and $r'_i, t_{\rho_s^*(i)}$ for $i \in [\ell_s^*]$ modulo p_1 are uncorrelated from these values modulo p_2. Hence, $\mathcal{V}$ is a properly distributed sf-type 1 verification text. Therefore, if $\beta = 0$, i.e., $T_\beta = g^t g_3^\varsigma$, then the joint distribution of keys, signatures and vText are identical to that of Game$_{1-(k-1)-2}$. 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, t_i modulo p_1 and a, t_i modulo p_2 are uncorrelated, $\mathcal{SK}_{A_k}$ 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 $\vec{v}_s$). If we show either $d = ba$ or ac is independent from the adversary's point of view, then we are done.

Claim B.4. *The shared value in $\mathbb{G}_{p_2}$, i.e., 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 $\mathcal{SK}_A$ such that $\Gamma_s^*(A) = \mathsf{False}$. Proof of the claim is found in proof of the lemma 8 in [LOS+10].

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

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

Proof. It is similar to that of Lemma B.3 except, the k^{th} key query answering. An instance of DSG2, $(\mathcal{J}, g, X_1X_2, Y_2Y_3, X_3, T_\beta)$ with $\beta \xleftarrow{\mathrm{U}} \{0,1\}$ is given to the simulator $\mathscr{B}$ and depending on the distribution of β, it either simulates 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 $\mathcal{SK}_{A_k}$ using T_β of the instance of DSG2. $\mathscr{B}$ implicitly sets g^t part 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 $i \in A_k$

  $$\mathcal{SK}_{A_k} := [A_k,\ K := g^\alpha T_\beta^a (Y_2Y_3)^\zeta,\ L := T_\beta R'_0,\ K_i := T_\beta^{t_i} R_i,\ \forall i \in A_k]$$

In k^{th} key construction an additional term, $(Y_2Y_3)^\zeta$ is added to K. This prevents the requirement of the claim B.4.

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 are 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 $Y_2 = g_2^y$ $z_i := t_i$ for $i \in A_k$. Due to the distribution of $d = ba + y\zeta$, the above correlation between key and ciphertext is not possible in this lemma.

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

Lemma B.6. Game$_{2-(k-1)-2}$ *and* Game$_{2-k-1}$ *are indistinguishable under the DSG2 assumption and collision resistant property of H_s. That is, for every adversary $\mathscr{A}$, there exists PPT algorithms, $\mathscr{B}$ and $\mathcal{H}$ such that* $|\mathsf{Adv}_{\mathscr{A},\mathrm{ABS}}^{2-(k-1)-2}(\kappa) - \mathsf{Adv}_{\mathscr{A},\mathrm{ABS}}^{2-k-1}(\kappa)| \leq \mathsf{Adv}_{\mathscr{B}}^{\mathrm{DSG2}}(\kappa) + \mathsf{Adv}_{\mathcal{H}}^{CR-H_s}(\kappa)$ *for* $1 \leq k \leq \nu_2$.

Proof. Similar to previous lemma, $\mathscr{B}$ receives an instance of DSG2, $(\mathcal{J}, g, X_1X_2, Y_2Y_3, X_3, T_\beta)$ with $\beta \xleftarrow{\text{U}} \{0,1\}$ and depending on the distribution of β, $\mathscr{B}$ either simulates $\text{Game}_{2-(k-1)-2}$ or Game_{2-k-1}.

Setup: Similar to Lemma B.2

Key Query Answering: It is sf-type 2 key. Let A_j be the j^{th} query set of attributes. 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 to $\mathscr{A}$.

$$\mathcal{SK}_{A_j} := [A_j,\ K := g^{\alpha+at}(Y_2Y_3)^t,\ L := g^tR'_0,\ K_i := {T_i}^tR_i,\ \forall i \in A_j]$$

Since, t modulo p_2 and t modulo p_3 are uncorrelated, so the key $\mathcal{SK}_{A_j}$ is properly distributed sf-type 2 key.

Signature Query Answering: 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}. Let $(m^{(j)}, \Gamma_s^{(j)} := (M_s, \rho_s))$ be the j^{th} query tuple, where M_s is an $\ell_s \times n_s$ matrix. It chooses $r_s, \tilde{t}, \tau \xleftarrow{\text{U}} \mathbb{Z}_N$, $\tilde{R}_0, \tilde{R}_i, \tilde{R}'_i \xleftarrow{\text{U}} \mathbb{G}_{p_3}$ for $i \in [\ell_s]$. $\mathscr{B}$ chooses a set of attributes A such that $\Gamma_s^{(j)}(A) = \mathsf{True}$. It then, chooses sets $\mathcal{I}_A \subset [\ell_s]$ and $\{\alpha_s^{(i)}\}_{i \in \mathcal{I}_A}$ such that $\sum_{i \in \mathcal{I}_A} \alpha_s^{(i)} \vec{M}_s^{(i)} = \vec{1}$. $\mathscr{B}$ sets $\alpha_s^{(i)} := 0$ for $i \notin \mathcal{I}_A$. It selects $\vec{\beta} \xleftarrow{\text{U}} \{\vec{\beta} = (\vec{\beta}_1, \ldots, \vec{\beta}_{\ell_s}) \in \mathbb{Z}_N^{\ell_s} \mid \sum_{i \in [\ell_s]} \beta_i \vec{M}_s^{(i)} = \vec{0}\}$. It sets $h_s^{(j)} := H_s(m^{(j)} || \Gamma_s^{(j)})$. Then, $\mathscr{B}$ answers the signature σ_j for $(m^{(j)}, \Gamma_s^{(j)})$ as follows:

- If $j > k$, it is normal. $\mathscr{B}$ can handle it using $\mathcal{MSK}$.
- If $j < k$, it is sf-type 2. $\mathscr{B}$ computes the components of σ_j as below.

  $$\begin{aligned} \vec{S}_0 &:= \big(\ g^{\alpha+a\tilde{t}}(u_s^{h_s^{(j)}}v_s)^{r_s}(Y_2Y_3)^{\tilde{t}},\ g^{r_s}\tilde{R}_0\ \big) \\ \vec{S}_i &:= \big(\ (g^{\tilde{t}})^{\alpha_s^{(i)}}(g^{\tau})^{\beta_i}\tilde{R}_i, \qquad (T_{\rho_s(i)}^{\tilde{t}})^{\alpha_s^{(i)}}(T_{\rho_s(i)}^{\tau})^{\beta_i}\tilde{R}'_i\ \big) \text{ for } i \in [\ell_s] \end{aligned}$$

  Since, $\tilde{t}$ modulo p_2 and $\tilde{t}$ modulo p_3 are uncorrelated, so the key σ_j is properly distributed sf-type 2 signature.
- If $j = k$ then it is either normal or sf-type 1 signature. $\mathscr{B}$ generates σ_k using T_β of the instance of DSG2. $\mathscr{B}$ implicitly sets g^{r_s} part to be the $\mathbb{G}_{p_1}$ component of T_β.

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

Forgery: $\mathscr{A}$ outputs a signature σ^* for $(m^*, \Gamma_s^* := (M_s^*, \rho_s^*))$, where M_s^* is a matrix of order $\ell_s^* \times n_s^*$. Then, $\mathscr{B}$ prepares a vText $\mathcal{V}$ for (m^*, Γ_s^*) as follows: It computes $h_s^* := H_s(m^* || \Gamma_s^*)$. It selects $\vec{v}'_s := (1, v'_2, \ldots, v'_{n_s^*})$, where $v'_2, \ldots, v'_{n_s^*} \xleftarrow{\text{U}} \mathbb{Z}_N$.

$$\begin{aligned} \vec{V}_0 &:= \big(\ X_1X_2,\ (X_1X_2)^{h_s^*a_s+b_s},\ e(g^\alpha, X_1X_2)\ \big) \\ \vec{V}_i &:= \big(\ (X_1X_2)^{a(\vec{M}_s^{*(i)}.\vec{v}'_s)}(X_1X_2)^{-r'_it_{\rho^*(i)}}, \quad (X_1X_2)^{r'_i}\ \big), \text{ for } i \in [\ell_s^*] \end{aligned}$$

$\mathscr{B}$ verifies the signature σ^* using the vText $\mathcal{V}$ and returns 1 if it is valid else returns 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 are identical to that of $\text{Game}_{2-(k-1)-2}$. Now, suppose $\beta = 1$, i.e., $T_\beta = g^{r_s}g_2^bg_3^{\varsigma}$. Let $X_1X_2 = g^sg_2^c$. $\mathscr{B}$ implicitly sets $\tilde{d} := b(h_s^*a_s + b_s), \tilde{b} := b$ and $\iota := c(h_s^*a_s + b_s)$ in k^{th} key and vText respectively. Since, in the security definition B.2, the adversary must not forge σ^* for a pair (m^*, Γ_s^*), for which $\mathscr{A}$ has made a signature query implying that $(m^*, \Gamma_s^*) \neq (m^{(k)}, \Gamma_s^{(k)})$. Since, H_s is a collision resistant hash function, we have $h_s^* \neq h_s^{(k)}$. Hence, $h_s^*a_s + b_s$ and $h_s^{(k)}a_s + b_s$ are uniformly and independently distributed over $\mathbb{Z}_{p_2}$. Therefore, if $\beta = 1$, i.e., $T_\beta = g^{r_s} g_2^b g_3^c$, then the joint distribution of keys, signatures and vText are identical to that of Game_{2-k-1}. □

Lemma B.7. Game_{2-k-1} *and* Game_{2-k-2} *are indistinguishable under the DSG2 assumption. That is, for every adversary* $\mathscr{A}$*, there exists a PPT algorithm* $\mathscr{B}$ *such that* $|\mathsf{Adv}^{2-k-1}_{\mathscr{A},\text{ABS}}(\kappa) - \mathsf{Adv}^{2-k-2}_{\mathscr{A},\text{ABS}}(\kappa)| \leq \mathsf{Adv}^{\text{DSG2}}_{\mathscr{B}}(\kappa)$ *for* $1 \leq k \leq \nu_2$.

Proof. It is similar to the proof of Lemma B.6, except, answering the k^{th} signature query. $\mathscr{B}$ is given an instance of DSG2, $(\mathcal{J}, g, X_1X_2, Y_2Y_3, X_3, T_\beta)$ with $\beta \xleftarrow{\text{U}} \{0,1\}$ and depending on the distribution of β, $\mathscr{B}$ either simulates 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 σ_k using T_β of the instance of DSG2. $\mathscr{B}$ implicitly sets g^{r_s} part to be the $\mathbb{G}_{p_1}$ component of T_β.

$$\begin{aligned} \vec{S}_0 &:= \big(\ g^{\alpha + a\tilde{t}} (T_\beta)^{h_s^{(k)} a_s + b_s} (Y_2Y_3)^{\tilde{t}},\ T_\beta\ \big) \\ \vec{S}_i &:= \big(\ (g^{\tilde{t}})^{\alpha_s^{(i)}} (g^\tau)^{\beta_i} \tilde{R}_i, \qquad (T^{\tilde{t}}_{\rho_s(i)})^{\alpha_s^{(i)}} (T^{\tau}_{\rho_s(i)})^{\beta_i} \tilde{R'}_i\ \big) \text{ for } i \in [\ell_s] \end{aligned}$$

Note that an extra term, $(Y_2Y_3)^{\tilde{t}}$ is added to the first component of $\vec{S}_0$. Unlike to lemma B.6, we do not require the restriction argument between the forge and the signatures. It is straightforward that if $\beta = 0$ (resp. $\beta = 1$), the joint distribution of keys, signatures and vText are identical to that of Game_{2-k-1} (resp. Game_{2-k-2}). □

Lemma B.8. $\text{Game}_{2-\nu_2-2}$ *and* Game_{Final} *are indistinguishable under the DSG3 assumption. That is, for every adversary* $\mathscr{A}$*, there exists a PPT algorithm* $\mathscr{B}$ *such that* $|\mathsf{Adv}^{\text{Final}}_{\mathscr{A},\text{ABS}}(\kappa) - \mathsf{Adv}^{2-\nu_2-2}_{\mathscr{A},\text{ABS}}(\kappa)| \leq \mathsf{Adv}^{\text{DSG3}}_{\mathscr{B}}(\kappa)$

Proof. The simulator $\mathscr{B}$ receives an instance of DSG3, $(\mathcal{J}, g, g^\alpha X_2, g^s Y_2, Z_2, X_3, T_\beta)$ with $\beta \xleftarrow{\text{U}} \{0,1\}$ and depending on the distribution of β, it either simulates $\text{Game}_{2-\nu_2-2}$ 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 $i \in \mathcal{U}$. Then, it sets $u_s := g^{a_s}, v := g^{b_s}$ and $T_i := g^{t_i}$ for $i \in \mathcal{U}$. $\mathscr{B}$ picks a hash function $H_s : \{0,1\}^* \longrightarrow \mathbb{Z}_N$. It provides $\mathcal{PP} := (\mathcal{J}, g, g^a, u_s, v_s, g_T^\alpha := e(g, g^\alpha X_2), \{T_i\}_{i \in \mathcal{U}}, X_3, H_s)$ to $\mathscr{A}$. In this case, $\mathscr{B}$ does not know the master secret $\mathcal{MSK}$.

Key Query Answering: It is sf-type 2 key. Let A_j be the j^{th} query set of attributes. It picks $t \xleftarrow{\text{U}} \mathbb{Z}_N$, $R_0, R'_0, R_i \xleftarrow{\text{U}} \mathbb{G}_{p_3}$ for $i \in A_j$ and returns the following to $\mathscr{A}$.

$$\mathcal{SK}_{A_j} := [A_j,\ K := (g^\alpha X_2)(g^a Z_2)^t R_0,\ L := g^t R'_0,\ K_i := T_i{}^t R_i,\ \forall i \in A_j]$$

Since, t modulo p_1 and t modulo p_2 are uncorrelated, so the key $\mathcal{SK}_{A_j}$ is properly distributed sf-type 2 key.

Signature Query Answering: It is sf-type 2. Let $(m^{(j)}, \Gamma_s^{(j)} := (M_s, \rho_s))$ be the j^{th} query tuple, where 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 $i \in [\ell_s]$. $\mathscr{B}$ chooses a set of attributes A such that $\Gamma_s^{(j)}(A) = \mathsf{True}$, It then, chooses sets $\mathcal{I}_A \subset [\ell_s]$ and $\{\alpha_s^{(i)}\}_{i \in \mathcal{I}_A}$ such that $\sum_{i \in \mathcal{I}_A} \alpha_s^{(i)} \vec{M}_s^{(s)} = \vec{1}$. $\mathscr{B}$ sets $\alpha_s^{(i)} := 0$ for $i \notin \mathcal{I}_A$. It selects $\vec{\beta} \xleftarrow{\text{U}} \{\vec{\beta} = (\vec{\beta}_1, \ldots, \vec{\beta}_{\ell_s}) \in \mathbb{Z}_N^{\ell_s} \mid \sum_{i \in [\ell_s]} \beta_i \vec{M}_s^{(i)} = \vec{0}\}$. It sets $h_s^{(j)} := H_s(m^{(j)} || \Gamma_s^{(j)})$. Then, $\mathscr{B}$ answers the signature σ_j for $(m^{(j)}, \Gamma_s^{(j)})$ as follows:

$$\begin{aligned} \vec{S}_0 &:= \big(\ (g^\alpha X_2)(g^a Z_2)^{\tilde{t}} (u_s^{h_s^{(j)}} v_s)^{r_s} \tilde{R},\ g^{r_s} \tilde{R}_0\ \big) \\ \vec{S}_i &:= \big(\ (g^{\tilde{t}})^{\alpha_s^{(i)}} (g^\tau)^{\beta_i} \tilde{R}_i, \qquad (T^{\tilde{t}}_{\rho_s(i)})^{\alpha_s^{(i)}} (T^{\tau}_{\rho_s(i)})^{\beta_i} \tilde{R'}_i\ \big) \text{ for } i \in [\ell_s] \end{aligned}$$

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

Forgery: $\mathscr{A}$ outputs a signature σ^* for $(m^*, \Gamma_s^* := (M_s^*, \rho_s^*))$, where M_s^* is a matrix of order $\ell_s^* \times n_s^*$. Then $\mathscr{B}$ prepares a vText $\mathcal{V}$ for (m^*, Γ_s^*) as follows: It computes $h_s^* := H_s(m^*||\Gamma_s^*)$. It selects $\vec{v}'_s := (1, v'_2, \ldots, v'_{n_s^*})$, where $v'_2, \ldots, v'_{n_s^*} \xleftarrow{\text{U}} \mathbb{Z}_N$.

$\vec{V}_0 := \big(\ g^sY_2,\ (g^sY_2)^{h_s^* a_s + b_s},\ T_\beta\ \big)$

$\vec{V}_i := \big(\ (g^sY_2)^{a(\vec{M}_s^{*(i)}.\vec{v}'_s)}(g^sY_2)^{-r'_i t_{\rho_s^*(i)}},\quad (g^sY_2)^{r'_i}\ \big)$, for $i \in [\ell_s^*]$

$\mathscr{B}$ verifies the signature σ^* using the vText $\mathcal{V}$ and returns 1 if it is valid else returns 0.
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 are identical to that of Game$_{2-\nu_2-2}$ (resp. Game$_{Final}$). □

C Ciphertext-Policy Attribute-Base Signcryption

C.1 Definition

A ciphertext-policy Attribute-Base Signcryption(CP-ABSC) scheme consists of four PPT algorithms - Setup, KeyGen, Signcrypt and Unsigncrypt.

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

Correctness. For all $(\mathcal{PP}, \mathcal{MSK}) \longleftarrow$ Setup, all $m \in \mathcal{M}$, all attribute sets A, all keys $\mathcal{SK}_A \longleftarrow$ KeyGen$(\mathcal{PP}, \mathcal{MSK}, A)$, all signer policies Γ_s with $\Gamma_s(A) =$ True, all receiver policies Γ_e, all signcryptions $\Upsilon \longleftarrow$ Signcrypt$(\mathcal{PP}, m, \mathcal{SK}_A, \Gamma_s, \Gamma_e)$ and all attribute sets B with $\Gamma_e(B) =$ True, all keys $\mathcal{SK}_B \longleftarrow$ KeyGen$(\mathcal{PP}, \mathcal{MSK}, B)$, it is required that Unsigncrypt$(\mathcal{PP}, \Upsilon, \mathcal{SK}_A, \Gamma_s) = m$.

Definition C.1 (Signer Privacy of CP-ABSC)**.** A CP-ABSC scheme is said to be perfectly private if for all $(\mathcal{PP}, \mathcal{MSK}) \longleftarrow$ Setup, all attribute sets A_1, A_2, all keys $\mathcal{SK}_{A_1} \longleftarrow$ KeyGen$(\mathcal{PP}, \mathcal{MSK}, A_1)$, $\mathcal{SK}_{A_2} \longleftarrow$ KeyGen$(\mathcal{PP}, \mathcal{MSK}, A_2)$, all messages m, all claim-predicates $\Gamma_s := (M_s, \rho_s)$ for signer such that $\Gamma_s(A_1) = \Gamma_s(A_2) =$ True, and all claim-predicates $\Gamma_e := (M_e, \rho_e)$ for receiver, the distributions of Signcrypt$(\mathcal{PP}, m, \mathcal{SK}_{A_1}, \Gamma_s, \Gamma_e)$ and Signcrypt$(\mathcal{PP}, m, \mathcal{SK}_{A_2}, \Gamma_s, \Gamma_e)$ are identical.

Similar to AltSign defined in section B, for the CP-ABSC scheme having signer-privacy, one may replace Signcrypt$(\mathcal{PP}, m, \mathcal{SK}_A, \Gamma_s, \Gamma_e)$ oracle by an alternative signcryption oracle, AltSigncrypt$(\mathcal{PP}, m, \mathcal{MSK}, \Gamma_s, \Gamma_e)$ in the following two definitions.

C.2 Adaptive-Predicates IND-CCA2 Security of CP-ABSC

The adaptive security model is defined as an indistinguishability game between a challenger $\mathscr{B}$ and an adversary $\mathscr{A}$, where the adversary has to distinguish the signcryptions under adaptive chosen ciphertext attack (CCA). The game consists of the following phases:

Setup: The challenger $\mathscr{B}$ runs the Setup algorithm to produce the master secret key $\mathcal{MSK}$ and the public parameter $\mathcal{PP}$. Then, $\mathscr{B}$ gives $\mathcal{PP}$ to the adversary $\mathscr{A}$ and keeps $\mathcal{MSK}$ to itself.

Query: The adversary $\mathscr{A}$ is given access to the oracles $\mathsf{KeyGen}(\mathcal{PP}, \mathcal{MSK}, .)$, $\mathsf{Signcrypt}(\mathcal{PP}, ., .)$ and $\mathsf{Unsigncrypt}(\mathcal{PP}, ., .)$

Challenge: $\mathscr{A}$ provides two equal length messages m_0, m_1 and the challenge access policies Γ_s^*, Γ_e^* with the restriction that for each set of attributes A queried to $\mathsf{KeyGen}(\mathcal{PP}, \mathcal{MSK}, .)$ oracle, $\Gamma_e^*(A) = \mathsf{False}$. $\mathscr{B}$ picks $b \xleftarrow{\text{U}} \{0, 1\}$. Then, it signcrypts the challenge message m_b using the challenge policies Γ_s^* and Γ_e^* and gives the challenge signcryption Υ_b to $\mathscr{A}$.

Query: Again, $\mathscr{A}$ is given access to the oracles $\mathsf{KeyGen}(\mathcal{PP}, \mathcal{MSK}, .)$, $\mathsf{Signcrypt}(\mathcal{PP}, ., .)$ and $\mathsf{Unsigncrypt}(\mathcal{PP}, ., .)$ but with the restrictions that for each set of attributes, A queried to $\mathsf{KeyGen}(\mathcal{PP}, \mathcal{MSK}, .)$ implies $\Gamma_e^*(A) = \mathsf{False}$ and $(\Upsilon_b, B, \Gamma_s)$ with $\Gamma_e^*(B) = \mathsf{True}$ was never queried to $\mathsf{Unsigncrypt}(\mathcal{PP}, ., .)$ oracle.

Guess: $\mathscr{A}$ sends a guess b' to $\mathscr{B}$.

The advantage of $\mathscr{A}$ in above game is defined by

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

The CP-ABSC scheme is said to be *adaptively secure* (adaptive-predicates IND-CCA2 secure or in short APs-IND-CCA2) if for all PPT adversary $\mathscr{A}$, the advantage $\mathsf{Adv}_{\mathscr{A}}^{\text{ABSC-CCA}}(\kappa)$ is at most a negligible function in security parameter κ.

Remark C.1. Likewise in Selective-Predicates IND-CCA2 security, the adversary $\mathscr{A}$ submits receiver's policy Γ_e^* before receiving $\mathcal{PP}$ of ABSC.

C.3 Adaptive-Predicates Existential Unforgeability of CP-ABSC

The adaptive-predicates existential unforgeable security model is defined as a computational game between a challenger $\mathscr{B}$ and an adversary $\mathscr{A}$, where the adversary has to forge a signcryption for a message (consist of plaintext, signer's predicate and receiver's predicate). The game consists of the following phases:

Setup: The challenger $\mathscr{B}$ runs the Setup algorithm to produce the master secret key $\mathcal{MSK}$ and the public parameter $\mathcal{PP}$. Then, $\mathscr{B}$ gives $\mathcal{PP}$ to the adversary $\mathscr{A}$ and keeps $\mathcal{MSK}$ to itself.

Query: The adversary $\mathscr{A}$ is given access to the oracles $\mathsf{KeyGen}(\mathcal{PP}, \mathcal{MSK}, .)$, $\mathsf{Signcrypt}(\mathcal{PP}, ., .)$ and $\mathsf{Unsigncrypt}(\mathcal{PP}, ., .)$.

Forgery: The adversary outputs a tuple, $(\Upsilon^*, \Gamma_s^*, \Gamma_e^*)$.

We say the adversary succeeds in this game if $\mathsf{Unsigncrypt}(\mathcal{PP}, \Upsilon^*, \mathcal{SK}_B, \Gamma_s^*, \Gamma_e^*) = m^* \neq \perp$, where $\Gamma_e^*(B) = \mathsf{True}$ and $(m^*, \Gamma_s^*, \Gamma_e^*)$ was never queried to $\mathsf{Signcrypt}$ oracle and Γ_s^* does not accept any set of attributes queried to KeyGen oracle.

Let $\mathsf{Adv}_{\mathscr{A}}^{\text{ABSC-EUF}}(\kappa)$ denote the success probability for any adversary $\mathscr{A}$ in the above experiment. An CP-ABSC scheme is said to be *adaptive-predicates existential unforgeable* (APs-UF-CMA) if $\mathsf{Adv}_{\mathscr{A}}^{\text{ABSC-EUF}}(\kappa)$ is at most negligible function in κ

Remark C.2. Similar to section B.2, the above unforgeability 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* (APs-sUF-CMA), the adversary $\mathscr{A}$ may forge Υ^* for a queried message pair $(m^*, \Gamma_s^*, \Gamma_e^*)$ but the replied signcryption Υ on $(m^*, \Gamma_s^*, \gamma_e^*)$ must be different from the forge signcryption Υ^*.

Remark C.3. Similar to above, there is an another variant of unforgeability, called *selective-predicates unforgeability* in both weak and strong sense, where $\mathscr{A}$ submits signer's policy Γ_s^* before receiving $\mathcal{PP}$ of ABSC.

C.4 Proof of Theorem 6.2

Theorem C.1. *If DSG1, DSG2 and DSG3 assumptions hold, H_e is a collision resistant hash function and Π_{Commit} has hiding property, then our proposed basic CP-ABSC scheme in section 5 is adaptively secure.*

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

- Game$_{Real}$:= The original APs-IND-CCA2 security game of CP-ABSC.
- Game$_0$ (= Game$_{1-0-2}$) is just like Game$_{Real}$ except that the challenge signcryption is of sf-type 1.
- In Game$_{1-k-1}$ (for $1 \leq k \leq \nu_1$), challenge signcryption is sf-type 1, all the unsigncryption queries are answered by normal uq-Key, all the replied signcryptions are normal, the first $(k-1)$ keys returned to the adversary are sf-type 2, k^{th} key is sf-type 1 and the rest keys are normal.
- 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.
- In Game$_{2-k-1}$ (for $1 \leq k \leq \nu_2$), challenge signcryption is sf-type 1, all the replied signcryptions are normal, all the keys are sf-type 2, the first $(k-1)$ unsigncryption queries are answered by sf-type 2 uq-keys, k^{th} unsigncryption query is answered by sf-type 1 uq-key and the rest are answered by normal uq-keys. (So, in this sequel Game$_{2-0-2}$ = 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} unsigncryption query is answered by sf-type 2 uq-key.
- In Game$_{3-k-1}$ (for $1 \leq k \leq \nu_3$), challenge signcryption is sf-type 1, all the keys are sf-type 2, all the unsigncryption queries are answered by sf-type 2 uq-keys, the first $(k-1)$ replied signcryptions are of sf-type II, the k^{th} replied signcryption is sf-type I and the rest are normal signcryption. (So, in this sequel Game$_{3-0-2}$ = 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 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.

Let $\mathsf{Adv}^{\text{Real}}_{\mathscr{A},\text{ABSC}}(\kappa)$, $\mathsf{Adv}^{0}_{\mathscr{A},\text{ABSC}}(\kappa)$, $\mathsf{Adv}^{1-k-1}_{\mathscr{A},\text{ABSC}}(\kappa)$, $\mathsf{Adv}^{1-k-2}_{\mathscr{A},\text{ABSC}}(\kappa)$, $\mathsf{Adv}^{2-i-1}_{\mathscr{A},\text{ABSC}}(\kappa)$, $\mathsf{Adv}^{2-i-2}_{\mathscr{A},\text{ABSC}}(\kappa)$, $\mathsf{Adv}^{3-j-1}_{\mathscr{A},\text{ABSC}}(\kappa)$, $\mathsf{Adv}^{3-j-2}_{\mathscr{A},\text{ABSC}}(\kappa)$, $\mathsf{Adv}^{4}_{\mathscr{A},\text{ABSC}}(\kappa)$, $\mathsf{Adv}^{5}_{\mathscr{A},\text{ABSC}}(\kappa)$ and $\mathsf{Adv}^{\text{Final}}_{\mathscr{A},\text{ABSC}}(\kappa)$ denote the advantages of an adversary $\mathscr{A}$ in Game$_{Real}$, Game$_0$, Game$_{1-k-1}$, Game$_{1-k-2}$, Game$_{2-i-1}$, Game$_{2-i-2}$, Game$_{3-j-1}$, Game$_{3-j-2}$, Game$_4$, Game$_5$ and Game$_{Final}$ for $1 \leq k \leq \nu_1$, $1 \leq i \leq \nu_2$, $1 \leq j \leq \nu_3$ respectively.

Using lemmas C.2, C.3, C.4, C.5, C.7, C.8, C.9, C.10, C.11, C.12 and C.13, we have the following reduction

$$
\begin{aligned}
\mathsf{Adv}_{\mathscr{A}}^{\mathrm{ABSC-CCA}}(\kappa) &= \mathsf{Adv}_{\mathscr{A},\mathrm{ABSC}}^{\mathrm{Real}}(\kappa) \\
&\leq |\mathsf{Adv}_{\mathscr{A},\mathrm{ABSC}}^{\mathrm{Real}}(\kappa) - \mathsf{Adv}_{\mathscr{A},\mathrm{ABSC}}^{0}(\kappa)| \\
&\quad + \sum_{k=1}^{\nu_1}(|\mathsf{Adv}_{\mathscr{A},\mathrm{ABSC}}^{1-(k-1)-2}(\kappa) - \mathsf{Adv}_{\mathscr{A},\mathrm{ABSC}}^{1-k-1}(\kappa)| + |\mathsf{Adv}_{\mathscr{A},\mathrm{ABSC}}^{1-k-1}(\kappa) - \mathsf{Adv}_{\mathscr{A},\mathrm{ABSC}}^{1-k-2}(\kappa)|) \\
&\quad + \sum_{k=1}^{\nu_2}(|\mathsf{Adv}_{\mathscr{A},\mathrm{ABSC}}^{2-(k-1)-2}(\kappa) - \mathsf{Adv}_{\mathscr{A},\mathrm{ABSC}}^{2-k-1}(\kappa)| + |\mathsf{Adv}_{\mathscr{A},\mathrm{ABSC}}^{2-k-1}(\kappa) - \mathsf{Adv}_{\mathscr{A},\mathrm{ABSC}}^{2-k-2}(\kappa)|) \\
&\quad + \sum_{k=1}^{\nu_3}(|\mathsf{Adv}_{\mathscr{A},\mathrm{ABSC}}^{3-(k-1)-2}(\kappa) - \mathsf{Adv}_{\mathscr{A},\mathrm{ABSC}}^{3-k-1}(\kappa)| + |\mathsf{Adv}_{\mathscr{A},\mathrm{ABSC}}^{3-k-1}(\kappa) - \mathsf{Adv}_{\mathscr{A},\mathrm{ABSC}}^{3-k-2}(\kappa)|) \\
&\quad + |\mathsf{Adv}_{\mathscr{A},\mathrm{ABSC}}^{3-\nu_3-2}(\kappa) - \mathsf{Adv}_{\mathscr{A},\mathrm{ABSC}}^{4}(\kappa)| + |\mathsf{Adv}_{\mathscr{A},\mathrm{ABSC}}^{4}(\kappa) - \mathsf{Adv}_{\mathscr{A},\mathrm{ABSC}}^{5}(\kappa)| \\
&\quad + |\mathsf{Adv}_{\mathscr{A},\mathrm{ABSC}}^{5}(\kappa) - \mathsf{Adv}_{\mathscr{A},\mathrm{ABSC}}^{\mathrm{Final}}(\kappa)| + \mathsf{Adv}_{\mathscr{A},\mathrm{ABSC}}^{\mathrm{Final}}(\kappa) \\
&\leq \mathsf{Adv}_{\mathscr{B}_0}^{\mathrm{DSG1}}(\kappa) + \sum_{k=1}^{\nu_1}(\mathsf{Adv}_{\mathscr{B}_{1-k-1}}^{\mathrm{DSG2}}(\kappa) + \mathsf{Adv}_{\mathscr{B}_{1-k-2}}^{\mathrm{DSG2}}(\kappa)) \\
&+ \sum_{k=1}^{\nu_2}(\mathsf{Adv}_{\mathscr{B}_{2-k-1}}^{\mathrm{DSG2}}(\kappa) + \mathsf{Adv}_{\mathcal{H}_{2-k-1}}^{CR-H_e}(\kappa) + \mathsf{Adv}_{\mathscr{B}_{2-k-2}}^{\mathrm{DSG2}}(\kappa)) \\
&+ \sum_{k=1}^{\nu_3}(\mathsf{Adv}_{\mathscr{B}_{3-k-1}}^{\mathrm{DSG2}}(\kappa) + \mathsf{Adv}_{\mathscr{B}_{3-k-2}}^{\mathrm{DSG2}}(\kappa)) + \mathsf{Adv}_{\mathscr{B}_4}^{\mathrm{DSG2}}(\kappa) \ + \mathsf{Adv}_{\mathscr{B}_5}^{\mathrm{DSG2}}(\kappa) \\
&+ \mathsf{Adv}_{\mathscr{B}_6}^{\mathrm{DSG3}}(\kappa) \ + \mathsf{Adv}_{\mathscr{B}_7}^{\mathrm{Hiding}}(\kappa)
\end{aligned}
$$

where $\mathscr{B}_0, \mathscr{B}_{1-k-1}, \mathscr{B}_{1-k-2}, \mathscr{B}_{2-k-1}, \mathcal{H}_{2-k-1}, \mathscr{B}_{2-k-2}, \mathscr{B}_{3-k-1}, \mathscr{B}_{3-k-2}, \mathscr{B}_4, \mathscr{B}_5, \mathscr{B}_6$ and $\mathscr{B}_7$ are PPT algorithms whose running times are same as that of $\mathscr{A}$. This concludes the theorem. □

Lemma C.2. 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* $|\mathsf{Adv}_{\mathscr{A},\mathrm{ABSC}}^{\mathrm{Real}}(\kappa) - \mathsf{Adv}_{\mathscr{A},\mathrm{ABSC}}^{0}(\kappa)| \leq \mathsf{Adv}_{\mathscr{B}}^{\mathrm{DSG1}}(\kappa)$.

Proof. We construct a PPT algorithm $\mathscr{B}$ (called simulator) who receives an instance of DSG1, $(\mathcal{J}, g, X_3, T_\beta)$ with $\beta \xleftarrow{\mathrm{U}} \{0,1\}$ and depending on the distribution of β, it either simulates Game$_{Real}$ or Game$_0$.

Setup: $\mathscr{B}$ runs $\mathsf{C.Setup}(1^\kappa)$ to obtain the public commitment key $\mathcal{CK}$. $\mathscr{B}$ 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 $i \in \mathcal{U}$. 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 $i \in \mathcal{U}$. $\mathscr{B}$ selects hash functions $H_s, H_e : \{0,1\}^* \longrightarrow \mathbb{Z}_N$. It provides $\mathcal{PP} := (\mathcal{J}, g, g^a, u_s, u_e, v_s, v_e, g_T^\alpha, \{T_i\}_{i \in \mathcal{U}}, X_3, H_s, H_e, \mathcal{CK})$ to $\mathscr{A}$ and keeps $\mathcal{MSK} := (\alpha)$ to itself.

Key Query Answering: It is normal key. $\mathscr{B}$ can handle this, since the $\mathcal{MSK}$ is known to him.

Signcryption Query Answering: It is normal signcryption. Let $\mathscr{A}$ makes a signcryption query for the message (m, Γ_s, Γ_e). $\mathscr{B}$ first computes $\mathcal{SK}_B$ such that $\Gamma_s(B) = \mathsf{True}$ using $\mathcal{MSK}$, then the normal signcryption is generated using $\mathcal{SK}_B$.

Unsigncryption Query Answering: $\mathscr{B}$ unsigncrypts it by normal uq-key as described in section 6.3. Since, the $\mathcal{MSK}$ is known to $\mathscr{B}$, he can construct the normal uq-key and then returns the message if it is valid else returns $\perp$. (See footnote [9])

Challenge: $\mathscr{A}$ provides two equal length messages m_0, m_1 and the challenge access policies $\Gamma_s^* := (M_s^*, \rho_s^*), \Gamma_e^* := (M_e^*, \rho_e^*)$, where M_s^* (resp. 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 $(\breve{c}_b, \breve{d}_b) \longleftarrow \mathsf{Commit}(m_b)$. It chooses a set of attributes, A such that $\Gamma_s^*(A) = \mathsf{True}$ and then computes a normal key $\mathcal{SK}_A$ for A. Now, it executes $\mathsf{ABS.Sign}(\mathcal{ABS.PP}, (\breve{c}_b || \Gamma_e^*), \mathcal{SK}_A, \Gamma_s^*)$ for the message $(\breve{c}_b || \Gamma_e^*)$ to have $\sigma_b := (\vec{S}_0, \{\vec{S}_i\}_{i \in [\ell_s^*]})$, where the components are given by

$$\begin{aligned} \vec{S}_0 &:= \big(\ g^{\alpha + a\tilde{t}}(u_s^{h_s^*} v_s)^{r_s} \tilde{R},\ g^{r_s} \tilde{R}_0\ \big), \text{ where } h_s^* := H_s(\breve{c}_b || \Gamma_e^* || \Gamma_s^*) \\ \vec{S}_i &:= \big(\ (g^{\tilde{t}})^{\alpha_s^{(i)}} (g^{\tau})^{\beta_i} \tilde{R}_i,\ \ (T_{\rho_s(i)}^{\tilde{t}})^{\alpha_s^{(i)}} (T_{\rho_s(i)}^{\tau})^{\beta_i} \tilde{R}'_i\ \big) \end{aligned}$$

It selects $\vec{v}'_e := (1, v'_2, \ldots, v'_{n_e^*})$, where $v'_2, \ldots, v'_{n_e^*} \xleftarrow{\text{U}} \mathbb{Z}_N$. 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

$$\begin{aligned} \vec{C}_0 &:= \big(\ T_\beta,\ \breve{d}_b . e(g^\alpha, T_\beta)\ \big) \\ \vec{C}_i &:= \big(\ T_\beta^{a(\vec{M}_e^{*(i)} . \vec{v}'_s)} T_\beta^{-r'_i t_{\rho_e^*(i)}},\quad T_\beta^{r'_i}\ \big) \text{ for } i \in [\ell_e^*] \end{aligned}$$

Now, it sets $\varrho_{b0} := (\vec{C}_0, \ldots, \vec{C}_{\ell_e^*})$ and computes $h_e^* := H_e(\breve{c}_b, \varrho_{b0}, \sigma_b)$. Then it computes another ciphertext component as

$$C_{\ell_e^*+1} := T_\beta^{a_e h_e^* + b_e}$$

So, the ciphertext part of the signcryption is $\varrho_b := (\varrho_{b0}, C_{\ell_e^*+1})$. $\mathscr{B}$ returns the challenge signcryption $\Upsilon_b := (\breve{c}_b, \sigma_b, \varrho_b)$ to $\mathscr{A}$.

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

Analysis: First of all, note that the signature part σ_b of the challenge signcryption Υ_b is identical to that of the normal signcryption as well as sf-type 1 signcryption. Let's concentrate on the distribution of ciphertext part ϱ_b of Υ_b. $\mathscr{B}$ implicitly sets $\vec{u}_e := s_e \vec{v}'_e = (s_e, s_e v'_2, \ldots, s_e v'_{n_e^*})$ 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 $\vec{u}_e$ is a random vector over $\mathbb{Z}_{p_1}$. Similarly, since $r'_1, \ldots, r'_{\ell_e^*}$ are uniformly and independently distributed over $\mathbb{Z}_N$, so are $r_e^{(1)}, \ldots, r_e^{(\ell_e^*)}$ over $\mathbb{Z}_{p_1}$. Therefore, if $\beta = 0$, i.e., $T_\beta = g^{s_e}$, Υ_b is a properly distributed normal signcryption.

Suppose $\beta = 1$, i.e., $T_\beta = g^{s_e} g_2^c$ for some $c \in \mathbb{Z}_N$. It implicitly sets $\vec{v}_e := ca\vec{v}'_e = (ca, cav'_2, \ldots, cav'_{n_e^*})$, $\iota := c(h_e^* a_e + b_e)$, $\gamma_e^{(i)} := -cr'_i$ 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 $i \in [\ell_e^*]$ modulo p_1 are uncorrelated from these values modulo p_2. Therefore, if $\beta = 1$, Υ_b is a properly distributed sf-type 1 signcryption. □

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

Proof. $\mathscr{B}$ is given an instance of DSG2, $(\mathcal{J}, g, X_1X_2, Y_2Y_3, X_3, T_\beta)$ with $\beta \xleftarrow{\text{U}} \{0,1\}$ and depending on the distribution of β, $\mathscr{B}$ either simulates Game$_{1-(k-1)-2}$ or Game$_{1-k-1}$.

Setup: Similar to Lemma C.2.

Key Query Answering: 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 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 $\mathcal{SK}_{A_j}$ for A_j as follows:

[9] The queries (key, signcryption and unsigncryption) can be adaptive, i.e., before and after the challenge phase but with the natural restriction as in definition C.2. In the proof of the lemmas, we write the queries before challenge phase, but it covers the queries before challenge and after challenge.

- 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 to $\mathscr{A}$.

  $\mathcal{SK}_{A_j} := [A_j,\ K := g^{\alpha+at}(Y_2Y_3)^t,\ L := g^tR'_0,\ K_i := T_i{}^tR_i,\ \forall i \in A_j]$

  Since, t modulo p_2 and t modulo p_3 are uncorrelated, so the key $\mathcal{SK}_{A_j}$ is properly distributed sf-type 2 key.
- If $j = k$ then it is either normal or sf-type 1 key. $\mathscr{B}$ generates $\mathcal{SK}_{A_k}$ using T_β of the instance of DSG2. $\mathscr{B}$ implicitly sets g^t part to be the $\mathbb{G}_{p_1}$ component of T_β. It chooses $R, R'_0, R_i \xleftarrow{\text{U}} \mathbb{G}_{p_3}$ for $i \in A_k$

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

Signcryption Query Answering: Similar to lemma C.2.

Unsigncryption Query Answering: Similar to lemma C.2.

Challenge: It is similar to that of lemma C.2 except the ciphertext part ϱ_b of the challenge signcryption Υ_b as given below: It selects $\vec{v}'_e := (1, v'_2, \ldots, v'_{n^*_e})$, where $v'_2, \ldots, v'_{n^*_e} \xleftarrow{\text{U}} \mathbb{Z}_N$. It chooses $r'_i \xleftarrow{\text{U}} \mathbb{Z}_N$ for $i \in [\ell^*_e]$. The ciphertext components of the signcryption are given by

$$\begin{aligned} \vec{C}_0 &:= \big(\ X_1X_2,\ \breve{d}_b.e(g^\alpha, X_1X_2)\ \big) \\ \vec{C}_i &:= \big(\ (X_1X_2)^{a(\vec{M}^{*(i)}_e.\vec{v}'_s)}(X_1X_2)^{-r'_i t_{\rho^*_e(i)}},\quad (X_1X_2)^{r'_i}\ \big) \text{ for } i \in [\ell^*_e] \end{aligned}$$

Now, it sets $\varrho_{b0} := (\vec{C}_0, \ldots, \vec{C}_{\ell^*_e})$ and then computes $h^*_e := H_e(\breve{c}_b, \varrho_{b0}, \sigma_b)$. Then it computes another ciphertext component as

$$C_{\ell^*_e+1} := (X_1X_2)^{a_eh^*_e+b_e}$$

So, the ciphertext part of the signcryption is $\varrho_b := (\varrho_{b0}, C_{\ell^*_e+1})$.

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

Analysis: Let $X_1X_2 = g^{s_e}g_2^c$. $\mathscr{B}$ implicitly sets $\vec{u}_e := s_e\vec{v}'_e = (s_e, s_ev'_2, \ldots, s_ev'_{n^*_e})$ 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 $\mathbb{Z}_N$, so the vector $\vec{u}_e$ is a random vector over $\mathbb{Z}_{p_1}$. Similarly, since $r'_1, \ldots, r'_{\ell^*_e}$ are uniformly and independently distributed over $\mathbb{Z}_N$, so are $r_e^{(1)}, \ldots, r_e^{(\ell^*_e)}$ over $\mathbb{Z}_{p_1}$. It implicitly sets $\vec{v}_e := ca\vec{v}'_e = (ca, cav'_2, \ldots, cav'_{n^*_e})$, $\iota := c(h^*_ea_e + b_e)$, $\gamma_e^{(i)} := -cr'_i$ 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 $i \in [\ell^*_e]$ modulo p_1 are uncorrelated from these values modulo p_2. Hence, Υ_b is a properly distributed sf-type 1 signcryption. Therefore, if $\beta = 0$, i.e., $T_\beta = g^tg_3^\varsigma$, then the joint distribution of keys, signcryptions, uq-keys and challenge signcryption are identical to that of $\text{Game}_{1-(k-1)-2}$. Now, suppose $\beta = 1$, i.e., $T_\beta = g^tg_2^bg_3^\varsigma$. $\mathscr{B}$ implicitly sets $d := ba$, $z_i := t_i$ for $i \in A_k$. Since, a, t_i modulo p_1 and a, t_i modulo p_2 are uncorrelated, $\mathcal{SK}_{A_k}$ 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 $\vec{v}_e$). Since, the adversary $\mathscr{A}$ is forbidden to ask for a key $\mathcal{SK}_A$ such that $\Gamma^*_e(A) = \mathsf{True}$ and ρ^*_e is injective, by claim B.4 the above correlation can be shown to be hidden to $\mathscr{A}$.

Therefore, if $\beta = 1$, i.e., $T_\beta = g^tg_2^bg_3^\varsigma$, then the joint distribution of keys, signcryptions, uq-keys and challenge signcryption are identical to that of Game_{1-k-1}. □

Lemma C.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* $|\mathsf{Adv}^{1-k-1}_{\mathscr{A},\text{ABSC}}(\kappa) - \mathsf{Adv}^{1-k-2}_{\mathscr{A},\text{ABSC}}(\kappa)| \le \mathsf{Adv}^{\text{DSG2}}_{\mathscr{B}}(\kappa)$ *for* $1 \le k \le \nu_1$.

Proof. It is similar to that of Lemma C.3 except, the k^{th} key query answering. An instance of DSG2, $(\mathcal{J}, g, X_1X_2, Y_2Y_3, X_3, T_\beta)$ with $\beta \xleftarrow{\text{U}} \{0,1\}$ is given to the simulator $\mathscr{B}$ and depending on the distribution of β, it either simulates 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 $\mathcal{SK}_{A_k}$ using T_β of the instance of DSG2. $\mathscr{B}$ implicitly sets g^t part to be the $\mathbb{G}_{p_1}$ component of T_β. It chooses $\zeta \xleftarrow{\text{U}} \mathbb{Z}_N,\ R'_0, R_i \xleftarrow{\text{U}} \mathbb{G}_{p_3}$ for $i \in A_k$

  $$\mathcal{SK}_{A_k} := [A_k,\ K := g^\alpha T_\beta^a (Y_2Y_3)^\zeta,\ L := T_\beta R'_0,\ K_i := T_\beta^{t_i} R_i,\ \forall i \in A_k]$$

In above computation, an extra term $(Y_2Y_3)^\zeta$ 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 the claim B.4. Rest of the proof is very straightforward. □

Lemma C.5. $\text{Game}_{2-(k-1)-2}$ *and* Game_{2-k-1} *are indistinguishable under the DSG2 assumption and collision resistant property of* H_e. *That is, for every adversary* $\mathscr{A}$ *there exists PPT algorithms,* $\mathscr{B}$ *and* $\mathcal{H}$ *such that* $|\mathsf{Adv}^{2-(k-1)-2}_{\mathscr{A},\text{ABSC}}(\kappa) - \mathsf{Adv}^{2-k-1}_{\mathscr{A},\text{ABSC}}(\kappa)| \leq \mathsf{Adv}^{\text{DSG2}}_{\mathscr{B}}(\kappa) + \mathsf{Adv}^{CR-H_e}_{\mathcal{H}}(\kappa)$ *for* $1 \leq k \leq \nu_2$.

Proof. Similar to previous lemma, $\mathscr{B}$ receives an instance of DSG2, $(\mathcal{J}, g, X_1X_2, Y_2Y_3, X_3, T_\beta)$ with $\beta \xleftarrow{\text{U}} \{0,1\}$ and depending on the distribution of β, $\mathscr{B}$ either simulates $\text{Game}_{2-(k-1)-2}$ or Game_{2-k-1}.

Setup: Similar to Lemma C.2

Key Query Answering: It is sf-type 2 key. Let A_j be the j^{th} query set of attributes. 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 to $\mathscr{A}$.

$$\mathcal{SK}_{A_j} := [A_j,\ K := g^{\alpha+at}(Y_2Y_3)^t,\ L := g^t R'_0,\ K_i := T_i{}^t R_i,\ \forall i \in A_j]$$

Since, t modulo p_2 and t modulo p_3 are uncorrelated, so the key $\mathcal{SK}_{A_j}$ is properly distributed sf-type 2 key.

Signcryption Query Answering: Similar to lemma C.2.

Unsigncryption Query Answering: The first $(k-1)$ unsigncryption queries are answered by sf-type 2 uq-keys and the last (ν_2-k) are unsigncrypted by normal uq-keys. The k^{th} unsigncryption 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 $(\Upsilon_j, B_j, \Gamma_s^{(j)})$ be the j^{th} unsigncryption query, where $\Upsilon_j := (\breve{c}_j, \sigma_j, \varrho^{(j)})$, $\sigma_j := (\vec{S}_0, \{\vec{S}_i\}_{i\in[\ell_s^{(j)}]})$, $\varrho^{(j)} := (\varrho_0^{(j)}, C_{\ell_e^{(j)}+1})$, $\varrho_0^{(j)} := (\vec{C}_0, \{\vec{C}_i\}_{i\in[\ell_e^{(j)}]})$ and $\Gamma_e^{(j)}$ is the policy implicitly contained in Υ_j. $\mathscr{B}$ computes $h_e^{(j)} := H_e(\breve{c}_j, \varrho_0^{(j)}, \sigma_j)$. It picks $r, t \xleftarrow{\text{U}} \mathbb{Z}_N,\ R_0, R'_0, R_i \xleftarrow{\text{U}} \mathbb{G}_{p_3}$ for $i \in B_j$. Demonstrated below are the different types of uq-keys to be used to answer the unsigncryption queries (as described in section 6.3). To each query, $\mathscr{B}$ answers using the key $\mathcal{USK}_{B_j}$ if the unsigncryption query is valid[10] else returns $\perp$.

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

  $$\mathcal{USK}_{B_j} := [B_j,\ K := g^{\alpha+at}(u_e^{h_e^{(j)}} v_e)^r (Y_2Y_3)^t,\ K_0 := g^r R_0,\ L := g^t R'_0,\ K_i := T_i{}^t R_i\ \forall i \in B_j]$$

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

[10] Here valid query means it does not violet the rules of the security game and $\Gamma_e^{(j)}(B_j) = \mathsf{True}$, i.e., if $\Gamma^{(j)}(B_j) = \mathsf{False}$ (invalid), $\mathscr{B}$ returns $\perp$ else if $\Upsilon_j = \Upsilon_b$ (invalid), returns $\perp$ else (valid), answers with $\mathcal{USK}_j$

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

$$\mathcal{USK}_{B_k} := [B_k,\ K := g^{\alpha+at}(T_\beta)^{h_e^{(k)} a_e + b_e},\ K_0 := T_\beta,\ L := g^t R_0',\ K_i := T_i{}^t R_i\ \forall i \in B_k]$$

Challenge: Similar to Lemma C.3, but still the ciphertext part ϱ_b of the challenge signcryption Υ_b are illustrated. It selects $\vec{v}_e' := (1, v_2', \ldots, v_{n_e^*}')$, where $v_2', \ldots, v_{n_e^*}' \xleftarrow{\text{U}} \mathbb{Z}_N$. It chooses $r_i' \xleftarrow{\text{U}} \mathbb{Z}_N$ for $i \in [\ell_e^*]$. The ciphertext components of the signcryption are given by

$$\begin{array}{ll} \vec{C}_0 & := \big(\ X_1X_2,\ \breve{d}_b.e(g^\alpha, X_1X_2)\ \big) \\ \vec{C}_i & := \big(\ (X_1X_2)^{a(\vec{M}_e^{*(i)}.\vec{v}_s')}(X_1X_2)^{-r_i' t_{\rho_e^*(i)}}, \quad (X_1X_2)^{r_i'}\ \big) \text{ for } i \in [\ell_e^*] \end{array}$$

Now, it sets $\varrho_{b0} := (\vec{C}_0, \ldots, \vec{C}_{\ell_e^*})$ and then computes $h_e^* := H_e(\breve{c}_b, \varrho_{b0}, \sigma_b)$. Then it computes the final component as

$$C_{\ell_e^*+1} := (X_1X_2)^{a_e h_e^* + b_e}$$

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

Analysis: Let $X_1X_2 = g^{s_e} g_2^c$. $\mathscr{B}$ implicitly sets $\vec{u}_e := s_e \vec{v}_e' = (s_e, s_e v_2', \ldots, s_e v_{n_e^*}')$ 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 $\vec{u}_e$ is a random vector over $\mathbb{Z}_{p_1}$. Similarly, since $r_1', \ldots, r_{\ell_e^*}'$ are uniformly and independently distributed over $\mathbb{Z}_N$, so are $r_e^{(1)}, \ldots, r_e^{(\ell_e^*)}$ over $\mathbb{Z}_{p_1}$. It implicitly sets $\vec{v}_e := ca\vec{v}_e' = (ca, cav_2', \ldots, cav_{n_e^*}')$, $\iota := c(h_e^* a_e + b_e)$, $\gamma_e^{(i)} := -cr_i'$ 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 $i \in [\ell_e^*]$ modulo p_1 are uncorrelated from these values modulo 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 $\text{Game}_{2-(k-1)-2}$.

Suppose $\beta = 1$, i.e., $T_\beta := g^r g_2^b g_3^\varsigma$. Lets take a look on the distribution of the exponents of g_2 in C_{01} and $C_{\ell_e^*+1}$, i.e., c and $\iota = c(h_e^* a_e + b_e)$. This type of correlation does not hamper our task unless the same correlation is found in other components. Unluckily, 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 $d = b(h_e^{(k)} a_e + b_e)$.

Claim C.6. *If H_e is a collision resistant hash function, then either $h_e^* \neq h_e^{(k)}$ or $\mathscr{B}$ returns $\perp$ (indicates invalid signcryption) to $\mathscr{A}$.*

From this claim, it is immediate that for a valid k^{th} unsigncryption query, the variables $(h_e^* a_e + b_e)$ and $(h_e^{(k)} a_e + b_e)$ are uniformly and independently distributed over $\mathbb{Z}_{p_2}$. Therefore, if $\beta = 1$ then the joint distribution of keys, signcryptions, uq-keys and challenge signcryption are identical to that of Game_{2-k-1}.

Proof of the Claim C.6. If the queried signcryption is invalid, then $\mathscr{B}$ returns $\perp$. Suppose the queried signcryption is a valid signcryption. Then, we must have the relation (from Unsigncrypt)

$$e(g, C_{\ell_e^{(k)}+1}) = e(u_e^{h_e^{(k)}} v_e, C_{01}) \tag{3}$$

To complete the proof of this claim, we will consider two cases, viz, k^{th} unsigncryption query is made *before challenge phase* and *after challenge phase.*

Before Challenge Phase : Since the components of the challenge signcryption Υ_b are computed by randomized way and h_e^* is calculated by taking hash (H_e) of all the components except, $C_{\ell_e^*+1}$, hence the pre-computed value $h_e^{(k)}$ can not be equal to h_e^*.

After Challenge Phase : Suppose $h_e^* = h_e^{(k)}$. Since H_e is a collision resistant hash function and $\Upsilon_b \neq \Upsilon_k$ (by security definition C.2), we must have[11] $C_{\ell_e^*+1} \neq C_{\ell_e^{(k)}+1}$, which implies

$$e(g, C_{\ell_e^*+1}) \neq e(g, C_{\ell_e^{(k)}+1}) \tag{4}$$

$$\begin{aligned} e(g, C_{\ell_e^*+1}) &= e(u_e^{h_e^*} v_e, C_{01}) \text{ (by construction)} && (5)\\ &= e(u_e^{h_e^{(k)}} v_e, C_{01}) \text{ (by assumption, i.e., } h_e^* = h_e^{(k)}) && (6) \end{aligned}$$

From relations 4, 5 and 6, we have following relation

$$e(g, C_{\ell_e^{(k)}+1}) \neq e(u_e^{h_e^{(k)}} v_e, C_{01}) \tag{7}$$

The relations 3 and 7 are contradictory to each other. Therefore, we have $h_e^* \neq h_e^{(k)}$ as our requirement.

□

□

Lemma C.7. Game$_{2-k-1}$ *and* Game$_{2-k-2}$ *are indistinguishable under the DSG2 assumption. That is, for every adversary* $\mathscr{A}$ *there exists a PPT algorithm* $\mathscr{B}$ *such that* $|\mathsf{Adv}^{2-k-1}_{\mathscr{A},\text{ABSC}}(\kappa) - \mathsf{Adv}^{2-k-2}_{\mathscr{A},\text{ABSC}}(\kappa)| \leq \mathsf{Adv}^{\text{DSG2}}_{\mathscr{B}}(\kappa)$ *for* $1 \leq k \leq \nu_2$.

Proof. It is similar to the proof of Lemma C.5, except, answering the k^{th} unsigncryption query (i.e., k^{th} uq-key). $\mathscr{B}$ is given an instance of DSG2, $(\mathcal{J}, g, X_1X_2, Y_2Y_3, X_3, T_\beta)$ with $\beta \xleftarrow{\text{U}} \{0,1\}$ and depending on the distribution of β, $\mathscr{B}$ either simulates Game$_{2-k-1}$ or Game$_{2-k-2}$. Below, we only provide the simulation of k^{th} unsigncryption query answering.

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

$$\mathcal{USK}_{B_k} := [B_k,\ K := g^{\alpha+at}(T_\beta)^{h_e^{(k)}a_e+b_e}(Y_2Y_3)^t,\ K_0 := T_\beta,\ L := g^tR'_0,\ K_i := T_i^{\,t}R_i\ \forall i \in B_k]$$

Note that an extra term, $(Y_2Y_3)^t$ 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. Therefore, Unlike to lemma C.5, we do not require the above claim. It is straightforward that if $\beta = 0$ (resp. $\beta = 1$), the joint distribution of keys, signcryptions, uq-keys and challenge signcryption are identical to that of Game$_{2-k-1}$ (resp. Game$_{2-k-2}$). □

Lemma C.8. Game$_{3-(k-1)-2}$ *and* Game$_{3-k-1}$ *are indistinguishable under the DSG2 assumption. That is, for every adversary* $\mathscr{A}$ *there exists a PPT algorithm* $\mathscr{B}$ *such that* $|\mathsf{Adv}^{3-(k-1)-2}_{\mathscr{A},\text{ABSC}}(\kappa) - \mathsf{Adv}^{3-k-1}_{\mathscr{A},\text{ABSC}}(\kappa)| \leq \mathsf{Adv}^{\text{DSG2}}_{\mathscr{B}}(\kappa)$ *for* $1 \leq k \leq \nu_3$.

[11] i.e., the corresponding components of Υ_b and Υ_k are equal except, $C_{\ell_e^*+1} \neq C_{\ell_e^{(k)}+1}$

Proof. An instance $(\mathcal{J}, g, X_1X_2, Y_2Y_3, X_3, T_\beta)$ with $\beta \xleftarrow{\text{U}} \{0,1\}$ of DSG2 is given to $\mathscr{B}$ and depending on the distribution of β, $\mathscr{B}$ either simulates $\text{Game}_{3-(k-1)-2}$ or Game_{3-k-1}

Setup: Similar to Lemma C.2.

Key Query Answering: Similar to lemma C.5.

Signcryption Query Answering: 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 $(m^{(j)}, \Gamma_s^{(j)}, \Gamma_e^{(j)})$ be the j^{th} signcryption query made by $\mathscr{A}$. $\mathscr{B}$ chooses a set of attributes A_j such that $\Gamma_s^{(j)}(A_j) = \mathsf{True}$ and answers the queries as follows:

- If $j > k$, then $\mathscr{B}$ first constructs normal key $\mathcal{SK}_{A_j}$ by running KeyGen algorithm and then it computes j^{th} signcryption Υ_j using the key $\mathcal{SK}_{A_j}$ (as in $\mathsf{Signcrypt}$ algorithm). The simulator $\mathscr{B}$ replies the normal signcryption Υ_j to $\mathscr{A}$.

- If $j < k$, then it is sf-type II signcryption. It first computes the sf-type 2 key $\mathcal{SK}_{A_j}$, then using this key it produces sf-type II signcryption Υ_j (similar to above case [12]).

- If $j = k$ then it is either normal or sf-type I signcryption. $\mathscr{B}$ generates signature part σ_j of the signcryption Υ_j using T_β of the instance of DSG2. $\mathscr{B}$ implicitly sets g^{r_s} part to be the $\mathbb{G}_{p_1}$ component of T_β.

  $$\begin{aligned} \vec{S}_0 \quad &:= \big(\ g^{\alpha + a\tilde{t}} (T_\beta)^{h_s^{(j)} a_s + b_s},\ T_\beta\ \big) \\ \vec{S}_i \quad &:= \big(\ (g^{\tilde{t}})^{\alpha_s^{(i)}} (g^\tau)^{\beta_i} \tilde{R}_i, \quad (T_{\rho_s(i)}^{\tilde{t}})^{\alpha_s^{(i)}} (T_{\rho_s(i)}^{\tau})^{\beta_i} \tilde{R}'_i\ \big) \text{ for } i \in [\ell_s^{(j)}] \end{aligned}$$

  The rest part of the signcryption Υ_j is computed as described in $\mathsf{Signcrypt}$ algorithm.

Unsigncryption Query Answering: It is unsigncrypted by sf-type 2 uq-key. Let $(\Upsilon_j, B_j, \Gamma_s^{(j)})$ be the j^{th} unsigncryption query, where $\Upsilon_j := (\breve{c}_j, \sigma_j, \varrho^{(j)})$, $\sigma_j := (\vec{S}_0, \{\vec{S}_i\}_{i \in [\ell_s^{(j)}]})$, $\varrho^{(j)} := (\varrho_0^{(j)}, C_{\ell_e^{(j)}+1})$, $\varrho_0^{(j)} := (\vec{C}_0, \{\vec{C}_i\}_{i \in [\ell_e^{(j)}]})$ and $\Gamma_e^{(j)}$ is the policy implicitly contained in Υ_j. $\mathscr{B}$ constructs the sf-type 2 uq-key $\mathcal{USK}_{B_j}$ (given below), then unsigncrypts Υ_j by $\mathcal{USK}_{B_j}$ as described in section 6.3. It then returns the message if it is valid else returns $\perp$.

$$\mathcal{USK}_{B_j} := [B_j,\ K := g^{\alpha + at} (u_e^{h_e^{(j)}} v_e)^r (Y_2Y_3)^t,\ K_0 := g^r R_0,\ L := g^t R'_0,\ K_i := T_i{}^t R_i\ \forall i \in B_j]$$

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

Challenge: Similar to Lemma C.5

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 are identical to that of $\text{Game}_{3-(k-1)-2}$ (resp. Game_{3-k-1}) if $\beta = 0$ (resp. $\beta = 1$). □

Lemma C.9. 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* $|\mathsf{Adv}_{\mathscr{A},\text{ABSC}}^{3-k-1}(\kappa) - \mathsf{Adv}_{\mathscr{A},\text{ABSC}}^{3-k-2}(\kappa)| \leq \mathsf{Adv}_{\mathscr{B}}^{\text{DSG2}}(\kappa)$ *for* $1 \leq k \leq \nu_3$.

Proof. Similar to the proof of Lemma C.8, except the S_{01} component of the signature part σ_k of the k^{th} signcryption query. $\mathscr{B}$ is given an instance of DSG2, $(\mathcal{J}, g, X_1X_2, Y_2Y_3, X_3, T_\beta)$ with $\beta \xleftarrow{\text{U}} \{0,1\}$ and

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

depending on the distribution of β, $\mathscr{B}$ either simulates Game$_{3-k-1}$ or Game$_{3-k-2}$. Described here is only the $\vec{S}_0$ part of the signature part σ_k.

$$\vec{S}_0 \quad := \big(\ g^{\alpha+a\tilde{t}}(T_\beta)^{h_s^* a_s+b_s}(Y_2Y_3)^{\tilde{t}},\ T_\beta\ \big)$$

Due to the additional term $(Y_2Y_3)^{\tilde{t}}$, the exponent of g_2 in S_{01} becomes uniformly and independently distributed random variable over $\mathbb{Z}_{p_2}$. Rest of the proof are easily handled similarly to the previous lemma. □

Lemma C.10. 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, $(\mathcal{J}, g, X_1X_2, Y_2Y_3, X_3, T_\beta)$ with $\beta \xleftarrow{\mathrm{U}} \{0,1\}$ and depending on the distribution of β, $\mathscr{B}$ either simulates Game$_{3-\nu_3-2}$ or Game$_4$.

Setup: Similar to Lemma C.2.

Key Query Answering: Similar to lemma C.5.

Signcryption Query Answering: It is sf-type II signcryption. It is generated in similar manner as sf-type II signcryptions construction in signcryption query answering of lemma C.8.

Unsigncryption Query Answering: Similar to lemma C.8.

Challenge: It is similar to lemma C.3 except the signature part σ_b of the challenge signature Υ_b. $\mathscr{B}$ generates σ_k using T_β of the instance of DSG2. $\mathscr{B}$ implicitly sets g^{r_s} part to be the $\mathbb{G}_{p_1}$ component of T_β.

$$\vec{S}_0 \quad := \big(\ g^{\alpha+a\tilde{t}}(T_\beta)^{h_s^* a_s+b_s},\quad T_\beta\ \big)$$
$$\vec{S}_i \quad := \big(\ (g^{\tilde{t}})^{\alpha_s^{(i)}}(g^{\tau})^{\beta_i}\tilde{R}_i, \qquad (T^{\tilde{t}}_{\rho_s(i)})^{\alpha_s^{(i)}}(T^{\tau}_{\rho_s(i)})^{\beta_i}\tilde{R}'_i\ \big) \text{ for } i \in [\ell_s^*]$$

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

Analysis: Similar to Lemma C.3, the distribution of the ciphertext part ϱ_b of the challenge signcryption Υ_b is identical to that of sf-type 1 and 2 signcryptions. Now, the rest of analysis depend upon the distribution of the signature part σ_b of Υ_b. It is easy to check that if $\beta = 0$ (resp. $\beta = 1$), the distribution of the signature part σ_b of Υ_b 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 Υ_b 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 are identical to that of Game$_{3-\nu_3-2}$ (resp. Game$_4$) if $\beta = 0$ (resp. $\beta = 1$). □

Lemma C.11. 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 C.10, except the signature part σ_b of the challenge signcryption Υ_b. $\mathscr{B}$ is given an instance of DSG2, $(\mathcal{J}, g, X_1X_2, Y_2Y_3, X_3, T_\beta)$ with $\beta \xleftarrow{\mathrm{U}} \{0,1\}$ and depending on the distribution of β, $\mathscr{B}$ either simulates Game$_4$ or Game$_5$. Demonstrated here is only the signature part σ_b.

$$\vec{S}_0 \quad := \big(\ g^{\alpha+a\tilde{t}}(T_\beta)^{h_s^* a_s+b_s}(Y_2Y_3)^{\tilde{t}},\ T_\beta\ \big)$$
$$\vec{S}_i \quad := \big(\ (g^{\tilde{t}})^{\alpha_s^{(i)}}(g^{\tau})^{\beta_i}\tilde{R}_i, \qquad (T^{\tilde{t}}_{\rho_s(i)})^{\alpha_s^{(i)}}(T^{\tau}_{\rho_s(i)})^{\beta_i}\tilde{R}'_i\ \big) \text{ for } i \in [\ell_s^*]$$

Rest of the proof are easily handled similarly to the previous lemma. □

Lemma C.12. 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, $(\mathcal{J}, g, g^{\alpha}X_2, g^sY_2, Z_2, X_3, T_\beta)$ with $\beta \xleftarrow{\text{U}} \{0,1\}$ and depending on the distribution of β, it either simulates Game$_5$ or Game$_{Final}$.

Setup: $\mathscr{B}$ runs $\mathsf{C.Setup}(1^\kappa)$ to obtain the public commitment key $\mathcal{CK}$. $\mathscr{B}$ 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 $i \in \mathcal{U}$. 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 $i \in \mathcal{U}$. $\mathscr{B}$ selects hash functions $H_s, H_e : \{0,1\}^* \longrightarrow \mathbb{Z}_N$. It provides $\mathcal{PP} := (\mathcal{J}, g, g^a, u_s, u_e, v_s, v_e, g_T^{\alpha} := e(g, g^{\alpha}X_2), \{T_i\}_{i \in \mathcal{U}}, X_3, H_s, H_e, \mathcal{CK})$ to $\mathscr{A}$. But $\mathscr{B}$ does not know the master secret $\mathcal{MSK}$.

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

$$\mathcal{SK}_{A_j} := [A_j,\ K := (g^{\alpha}X_2)(g^aZ_2)^tR,\ L := g^tR'_0,\ K_i := T_i{}^tR_i,\ \forall i \in A_j]$$

Since, t modulo p_1 and t modulo p_2 are uncorrelated, so the key $\mathcal{SK}_{A_j}$ is properly distributed sf-type 2 key.

Signcryption Query Answering: It is sf-type II signcryption. Let $\mathscr{A}$ makes a signcryption query for the message (m, Γ_s, Γ_e). $\mathscr{B}$ first computes a sf-type 2 key $\mathcal{SK}_A$ (as above) such that $\Gamma_s(A) = \mathsf{True}$, then using this key it produces the sf-type II signcryption Υ and returns it to $\mathscr{A}$.

Unsigncryption Query Answering: It is unsigncrypted by sf-type 2 uq-key. Let $(\Upsilon_j, B_j, \Gamma_s^{(j)})$ be the j^{th} unsigncryption query, where $\Gamma_e^{(j)}$ is the policy implicitly contained in Υ_j. $\mathscr{B}$ constructs the sf-type 2 uq-key $\mathcal{USK}_{B_j}$ (given below), then unsigncrypts it by $\mathcal{USK}_{B_j}$. It then returns the message if it is valid else returns $\perp$.

$$\mathcal{USK}_{B_j} := [B_j,\ K := (g^{\alpha}X_2)(g^aZ_2)^t(u_e^{h_e^{(j)}}v_e)^rR,\ K_0 := g^rR_0,\ L := g^tR'_0,\ K_i := T_i{}^tR_i\ \forall i \in B_j]$$

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

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

$$\begin{aligned} \vec{S}_0 &:= \big(\ (g^{\alpha}X_2)(g^aZ_2)^{\tilde{t}}(u_s^{h_s^*}v_s)^{r_s}\tilde{R}_0,\quad g^{r_s}\tilde{R}'_0\ \big) \\ \vec{S}_i &:= \big(\ (g^{\tilde{t}})^{\alpha_s^{(i)}}(g^{\tau})^{\beta_i}\tilde{R}_i,\qquad (T_{\rho_s(i)}^{\tilde{t}})^{\alpha_s^{(i)}}(T_{\rho_s(i)}^{\tau})^{\beta_i}\tilde{R}'{}_i\ \big) \text{ for } i \in [\ell_s^*] \end{aligned}$$

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

$$\begin{aligned} \vec{C}_0 &:= \big(\ g^sY_2,\ \breve{d}_b.T_\beta\ \big) \\ \vec{C}_i &:= \big(\ (g^sY_2)^{a(\vec{M}_s^{*(i)}.\vec{v}'_s)}(g^sY_2)^{-r'_it_{\rho_s^*(i)}},\quad (g^sY_2)^{r'_i}\ \big), \text{ for } i \in [\ell_e^*] \end{aligned}$$

Now, it sets $\varrho_{b0} := (\vec{C}_0, \ldots, \vec{C}_{\ell_e^*})$ and then computes $h_e^* := H_e(\breve{c}_b, \varrho_{b0}, \sigma_b)$. Then it computes the final component as

$$C_{\ell_e^*+1} := (g^sY_2)^{a_eh_e^*+b_e}$$

So, the ciphertext part of the signcryption is $\varrho_b := (\varrho_{b0}, C_{\ell_e^*+1})$. $\mathscr{B}$ returns the challenge signcryption $\Upsilon_b := (\breve{c}_b, \sigma_b, \varrho_b)$ to $\mathscr{A}$.

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

Analysis: It is obvious that if $\beta = 0$, i.e., $T_\beta := g_T^{\alpha s}$ (resp. if $\beta = 1$, i.e., $T_\beta \xleftarrow{\text{U}} \mathbb{G}_T$) form of the challenge signcryption Υ_b is identical to that of sf-type 2 (resp. sf-type 3) signcryption. Therefore, if $\beta = 0$ (resp. $\beta = 1$) then, the joint distribution of keys, signcryptions, uq-keys and challenge signcryption are identical to that of Game$_5$ (resp. Game$_{Final}$)

□

Lemma C.13. *For every adversary $\mathscr{A}$ there exists a PPT algorithm $\mathscr{B}$ such that* $\mathsf{Adv}^{\text{Final}}_{\mathscr{A},\text{ABSC}}(\kappa) \leq \mathsf{Adv}^{\text{Hiding}}_{\mathscr{B}}(\kappa)$

Proof. In Game$_{Final}$, the decommitment part $\breve{d}_b$ is masked with random element from $\mathbb{G}_T$ or in other word 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 $\breve{c}_b$ only may have the information about m_b. Now, we show that if the primitive commitment scheme Π_{Commit} 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 Π_{Commit}. Let $\mathcal{CH}$ be the challenger for the commitment scheme Π_{Commit}.

Setup: $\mathcal{CH}$ runs $\mathsf{C.Setup}(1^\kappa)$ and gives the public commitment key $\mathcal{CK}$ to $\mathscr{B}$. Now $\mathscr{B}$ executes $\mathcal{G}(1^\kappa)$ to have a composite order bilinear groups descriptor $\mathcal{J} := (N := p_1p_2p_3, \mathbb{G}, \mathbb{G}_T, e)$ with known factorization p_1, p_2 and p_3 of N. Then, $\mathscr{B}$ sets the public parameter $\mathcal{PP} := (\mathcal{J}, g, g^a, u_s, u_e, v_s, v_e, g_T^\alpha, \{T_i\}_{i\in\mathcal{U}}, X_3, H_s, H_e, \mathcal{CK})$ as per rule of the original setup and it gives $\mathcal{PP}$ to $\mathscr{A}$. Note that the simulator $\mathscr{B}$ knows all the secrets even including the generator of $\mathbb{G}_{p_2}$.

Key Query Answering: It is of sf-type 2 key. $\mathscr{B}$ can generate the key as it knows all the secrets of signcryption.

Signcryption Query Answering: It is sf-type II signcryption. By similar argument above, it is easily computable.

Unsigncryption Query Answering: It is unsigncrypted by sf-type 2 uq-key. Let $(\Upsilon_j, B_j, \Gamma_s^{(j)})$ be the j^{th} unsigncryption query, where $\Gamma_e^{(j)}$ is the policy implicitly contained in Υ_j. $\mathscr{B}$ generates the sf-type 2 uq-key $\mathcal{USK}_{B_j}$ and then unsigncrypts it by $\mathcal{USK}_{B_j}$. It then returns the message if it is valid else returns $\perp$.

Challenge: $\mathscr{A}$ provides two equal length messages m_0, m_1 and the challenge access policies $\Gamma_s^* := (M_s^*, \rho_s^*), \Gamma_e^* := (M_e^*, \rho_e^*)$, where M_s (resp. 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 $\mathcal{CH}$. Now $\mathcal{CH}$ chooses $m_b \xleftarrow{\text{U}} \{m_0, m_1\}$, runs $(\breve{c}_b, \breve{d}_b) \longleftarrow \mathsf{Commit}(m_b)$ and returns challenge commitment part $\breve{c}_b$ to $\mathscr{B}$. Note that $\mathscr{B}$ does not know the decommitment part $\breve{d}_b$ 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 $\mathsf{Signcrypt}$ algorithm on $(\breve{c}_b, d_r)$ using the policies Γ_s^* and Γ_e^* to produce the challenge signcryption[13] Υ_b 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 $\mathcal{CH}$.

Analysis: In sf-type 4 signcryption, the decommitment part $\breve{d}_b$ is masked with an uniformly and independently chosen element form $\mathbb{G}_T$ which is same as masking an 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.

□

C.5 Proof of Theorem 6.4

Suppose an adversary $\mathscr{A}$ can break the adaptive-predicates existential unforgeability of the proposed CP-ABSC scheme with non-negligible advantage ϵ. Lets assume that $\mathscr{A}$ has made ν number of signcryption

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

query to the signcryption oracle. Let $(m^{(i)}, \Gamma_s^{(i)}, \Gamma_e^{(i)})$ be the i^{th} query and $\Upsilon_i := (\breve{c}_i, \sigma_i, \varrho_i)$ be the corresponding replied signcryption. Let $\Upsilon := (\breve{c}, \sigma, \varrho)$ be the forge by $\mathscr{A}$ for the message (m, Γ_s, Γ_e). Let Forged be the event that $\breve{c}||\Gamma_e||\Gamma_s \notin \{\breve{c}_i||\Gamma_e^{(i)}||\Gamma_s^{(i)} \mid i \in [\nu]\}$. Then, we have

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

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

Case Forged : We establish a PPT algorithm $\mathscr{B}_{\mathsf{ABS}}$ (simulator) for forging to the primitive ABS scheme Π_{ABS} with advantage at least $\epsilon/2$. Basically in this simulation, the algorithm $\mathscr{B}_{\mathsf{ABS}}$ will make use of the adversary $\mathscr{A}$. Let $\mathcal{CH}$ be the challenger for the primitive ABS scheme Π_{ABS}.

Setup: First, the challenger $\mathcal{CH}$ publishes the public parameters $\mathcal{ABS.PP}$ of Π_{ABS}. Then, simulator $\mathscr{B}_{\mathsf{ABS}}$ runs $\mathsf{C.Setup}(1^\kappa)$ to produce $\mathcal{CK}$. $\mathscr{B}_{\mathsf{ABS}}$ chooses $a_e, b_e \xleftarrow{\text{U}} \mathbb{Z}_N$ and sets $u_e := g^{a_e}, v_e := g^{b_e}$. $\mathscr{B}_{\mathsf{ABS}}$ selects a hash function $H_e : \{0,1\}^* \longrightarrow \mathbb{Z}_N$. It provides $\mathcal{PP} := (\mathcal{J}, g, g^a, u_s, u_e, v_s, v_e, g_T^\alpha, \{T_i\}_{i \in \mathcal{U}}, X_3, H_s, H_e, \mathcal{CK})$ to $\mathscr{A}$. Interpretations of the variables described here are same as in the ABSC scheme in section 5.

Key Query Answering: Since the ABSC scheme and the primitive ABS scheme have the identical key distribution for a set of attributes, the key queries from $\mathscr{A}$ will be forwarded to the challenger $\mathcal{CH}$. Similarly, the answers (keys) will be reversed back to $\mathscr{A}$.

Signcryption Query Answering: Lets see how $\mathscr{B}_{\mathsf{ABS}}$ will answer the signcryption queries of $\mathscr{A}$. Let $(m^{(i)}, \Gamma_s^{(i)}, \Gamma_e^{(i)})$ be the i^{th} signcryption query to $\mathscr{B}_{\mathsf{ABS}}$ by $\mathscr{A}$. $\mathscr{B}_{\mathsf{ABS}}$ runs $(\breve{c}_i, \breve{d}_i) \longleftarrow \mathsf{Commit}(m^{(i)})$. Then, $\mathscr{B}_{\mathsf{ABS}}$ makes a signature query for $(\breve{c}_i||\Gamma_e^{(i)}, \Gamma_s^{(i)})$ to $\mathcal{CH}$ and gets the replied signature σ_i from $\mathcal{CH}$. Then, $\mathscr{B}_{\mathsf{ABS}}$ runs encryption routine of the Signcrypt algorithm to produce the ciphertext part ϱ_i and it returns the i^{th} signcryption $\Upsilon_i := (\breve{c}_i, \sigma_i, \varrho_i)$ to $\mathscr{A}$.

Forgery: $\mathscr{A}$ outputs a tuple $(\Upsilon, \Gamma_s, \Gamma_e)$. Then, $\mathscr{B}_{\mathsf{ABS}}$ forges the signature σ for $(\breve{c}||\Gamma_e, \Gamma_s)$ to the primitive ABS scheme Π_{ABS}.

Analysis: From the event Forged, it implies that $(\breve{c}||\Gamma_e, \Gamma_s)$ has not been queried for signature to $\mathcal{CH}$.

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

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

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

Signcryption Query Answering: Let $(m^{(j)}, \Gamma_s^{(j)}, \Gamma_e^{(j)})$ be the j^{th} signcryption query to $\mathscr{B}_{\mathsf{Commit}}$ by $\mathscr{A}$. If $j = i$, then $\mathscr{B}_{\mathsf{Commit}}$ makes a commitment query for the message $m^{(j)}$ to $\mathcal{CH}$ to have a pair $(\breve{c}_j, \breve{d}_j)$ else $\mathscr{B}_{\mathsf{Commit}}$ itself computes $(\breve{c}_j, \breve{d}_j)$. After that, $\mathscr{B}_{\mathsf{Commit}}$ follows the rest of Signcrypt algorithm in section 5 to produce the signcryption $\Upsilon_j := (\breve{c}_j, \sigma_j, \varrho_j)$ for $(m^{(j)}, \Gamma_s^{(j)}, \Gamma_e^{(j)})$ and returns it to $\mathscr{A}$.

Forgery: $\mathscr{A}$ outputs a tuple $(\Upsilon, \Gamma_s, \Gamma_e)$. By the event $\neg$Forged, we have $\breve{c}||\Gamma_e||\Gamma_s = \breve{c}_i||\Gamma_e^{(i)}||\Gamma_s^{(i)}$. Then, $\mathscr{B}_{\mathsf{Commit}}$ submits $\breve{d}$ to $\mathcal{CH}$ as a witness of breaking relaxed-binding property (for $(\breve{c}_i(= \breve{c}), \breve{d}_i)$) of Π_{Commit}. Note that $\mathscr{B}_{\mathsf{Commit}}$ obtains $\breve{d}$ by running the Unsigncrypt algorithm on input Υ as it knows the $\mathcal{MSK}$.

Analysis: With probability $1/\nu$, $\mathscr{B}_{\mathsf{Commit}}$ correctly guesses i such that $\check{c}||\Gamma_e||\Gamma_s = \check{c}_i||\Gamma_e^{(i)}||\Gamma_s^{(i)}$. It is easy to see that $m = \mathsf{Open}(\check{c}, \check{d})$ and $m^{(i)} = \mathsf{Open}(\check{c}_i, \check{d}_i)$. To draw the conclusion, we have to show that $m \neq m^{(i)}$. Indeed, if $m = m^{(i)}$ and we already have $\check{c}||\Gamma_e||\Gamma_s = \check{c}_i||\Gamma_e^{(i)}||\Gamma_s^{(i)}$ implying $(m^{(i)}, \Gamma_s^{(i)}, \Gamma_e^{(i)}) = (m, \Gamma_s, \Gamma_e)$. Hence, it shows that $\Upsilon := (\check{c}, \sigma, \varrho)$ is a forge for $(m^{(i)}, \Gamma_s^{(i)}, \Gamma_e^{(i)})$ (queried message), which is a contradiction to the definition of existential unforgeability of ABSC scheme.

C.6 Proof of Theorem 7.4

Let $\mathscr{A}$ be an adversary that breaks the adaptive-predicates strong existential unforgeability of the proposed CP-ABSC scheme with non-negligible advantage ϵ. Suppose $\mathscr{A}$ has made ν number of signcryption query to the signcryption oracle. Let $\Upsilon_i := (\check{c}_i, \sigma_s^{(i)}, \varrho^{(i)})$ be the replied signcryption to the i^{th} query message $(m^{(i)}, \Gamma_s^{(i)}, \Gamma_e^{(i)})$ for $i \in [\nu]$. Let $\Upsilon := (\check{c}, \sigma_s, \varrho)$ be the forge by $\mathscr{A}$ for the message (m, Γ_s, Γ_e). We define an event as

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

Then, we have

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

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

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

Setup: First, the challenger $\mathcal{CH}$ publishes the public parameters $\mathcal{ABS.PP}$ of Π_{wABS}. Then, the simulator $\mathscr{B}_{\mathsf{wABS}}$ runs $\mathsf{C.Setup}(1^\kappa)$ to produce $\mathcal{CK}$. $\mathscr{B}_{\mathsf{wABS}}$ 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 $\mathcal{PP} := (\mathcal{J}, g, g^a, u_s, u_e, v_s, v_e, g_T^\alpha, \{T_i\}_{i \in \mathcal{U}}, X_3, H_s, H_e, \mathcal{CK})$ to $\mathscr{A}$.

Key Query Answering: Since the ABSC scheme and the primitive ABS scheme have the identical key distribution for a set of attributes, the key queries from $\mathscr{A}$ will be forwarded to the challenger $\mathcal{CH}$. Similarly, the answers (keys) will be reversed back to $\mathscr{A}$.

Signcryption Query Answering: Let $(m^{(i)}, \Gamma_s^{(i)}, \Gamma_e^{(i)})$ be the i^{th} signcryption query to $\mathscr{B}_{\mathsf{wABS}}$ by $\mathscr{A}$. $\mathscr{B}_{\mathsf{wABS}}$ executes $(\check{c}_i, \check{d}_i) \longleftarrow \mathsf{Commit}(m^{(i)})$ and $(\mathsf{verk}^{(i)}, \mathsf{signk}^{(i)}) \longleftarrow \mathsf{Gen}(1^\kappa)$. Then, $\mathscr{B}_{\mathsf{wABS}}$ makes a signature query for $(\mathsf{verk}^{(i)}, \Gamma_s^{(i)})$ to $\mathcal{CH}$ and gets the replied signature $\sigma_w^{(i)}$ from $\mathcal{CH}$. Then, $\mathscr{B}_{\mathsf{wABS}}$ runs $\varrho_0^{(i)} \longleftarrow \mathsf{ABE.Encrypt}(\mathcal{PP}, \check{d}_i, \Gamma_e^{(i)})$. It computes $h_e^{(i)} := H_e(\check{c}_i, \varrho_0^{(i)}, \sigma_w^{(i)})$ and $C_{\ell_e^{(i)}+1} \longleftarrow \mathsf{fun}(\mathcal{PP}, h_e^{(i)}, s_e^{(i)})$. Then it executes $\sigma_o^{(i)} \longleftarrow \mathsf{OTS.Sign}(h_e^{(i)}||C_{\ell_e^{(i)}+1}||\Gamma_e^{(i)}||\Gamma_s^{(i)}, \mathsf{signk}^{(i)})$ and sets $\Upsilon_i := (\check{c}, \sigma_s^{(i)}, \varrho^{(i)})$, where $\sigma_s^{(i)} := (\sigma_w^{(i)}, \sigma_o^{(i)}, \mathsf{verk}^{(i)})$ and $\varrho^{(i)} := (\varrho_0^{(i)}, C_{\ell_e^{(i)}+1})$. It returns the i^{th} signcryption Υ_i to $\mathscr{A}$.

Forgery: $\mathscr{A}$ outputs a tuple $(\Upsilon, \Gamma_s, \Gamma_e)$. Then, $\mathscr{B}_{\mathsf{wABS}}$ forges the signature σ_w for $(\mathsf{verk}, \Gamma_s)$ to the primitive ABS scheme Π_{wABS}.

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

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

Setup: Similar to that of section 5.

Key Query Answering: $\mathscr{B}_{\mathsf{OTS}}$ can handle the key queries as it knows the $\mathcal{MSK}$.

Signcryption Query Answering: Let $(m^{(j)}, \Gamma_s^{(j)}, \Gamma_e^{(j)})$ be the j^{th} signcryption query to $\mathscr{B}_{\mathsf{OTS}}$ by $\mathscr{A}$.

▷ $(j \neq i)$:

$\mathscr{B}_{\mathsf{OTS}}$ executes $(\breve{c}_j, \breve{d}_j) \longleftarrow \mathsf{Commit}(m^{(j)})$, $(\mathsf{verk}^{(j)}, \mathsf{signk}^{(j)}) \longleftarrow \mathsf{Gen}(1^\kappa)$. Then, it runs $\sigma_w^{(j)} \longleftarrow \mathsf{wABS.Sign}(\mathcal{PP}, \mathsf{verk}^{(j)}, \mathcal{SK}_A, \Gamma_s^{(j)})$, $\varrho_0^{(j)} \longleftarrow \mathsf{ABE.Encrypt}(\mathcal{PP}, \breve{d}_j, \Gamma_e^{(j)})$, $C_{\ell_e^{(j)}+1} \longleftarrow \mathsf{fun}(\mathcal{PP}, h_e^{(j)}, s_e^{(j)})$, $\sigma_o^{(j)} \longleftarrow \mathsf{OTS.Sign}(h_e^{(j)}||C_{\ell_e^{(j)}+1}||\Gamma_e^{(j)}||\Gamma_s^{(j)}, \mathsf{signk}^{(j)})$. It sets $\sigma_s^{(j)} := (\sigma_w^{(j)}, \sigma_o^{(j)}, \mathsf{verk}^{(j)})$, $\varrho^{(j)} := (\varrho_0^{(j)}, C_{\ell_e^{(j)}+1})$ and returns the j^{th} signcryption $\Upsilon_j := (\breve{c}_j, \sigma_s^{(j)}, \varrho^{(j)})$ to $\mathscr{A}$.

▷ $(j = i)$:

Same as above except $\mathscr{B}_{\mathsf{OTS}}$ does not run $\mathsf{Gen}(1^\kappa)$ but it sets $\mathsf{verk}^{(j)} := \mathsf{verk}$ and it makes a query to $\mathcal{CH}$ for the message $h_e^{(j)}||C_{\ell_e^{(j)}+1}||\Gamma_e^{(j)}||\Gamma_s^{(j)}$ and gets the replied signature $\sigma_o^{(j)}$.

Forgery: $\mathscr{A}$ outputs a tuple $(\Upsilon, \Gamma_s, \Gamma_e)$. Then, $\mathscr{B}_{\mathsf{OTS}}$ forges the signature σ_o for $h_e||C_{\ell_e+1}||\Gamma_e||\Gamma_s$ to the primitive one-time signature scheme Π_{OTS}.

Analysis: With probability $1/\nu$, $\mathscr{B}_{\mathsf{OTS}}$ correctly guesses i such that this case is happened. Now we only have to show that $(h_e||C_{\ell_e+1}||\Gamma_e||\Gamma_s, \sigma_o) \neq (h_e^{(i)}||C_{\ell_e^{(i)}+1}||\Gamma_e^{(i)}||\Gamma_s^{(i)}, \sigma_o^{(i)})$. Indeed, if $(h_e||C_{\ell_e+1}||\Gamma_e||\Gamma_s, \sigma_o) = (h_e^{(i)}||C_{\ell_e^{(i)}+1}||\Gamma_e^{(i)}||\Gamma_s^{(i)}, \sigma_o^{(i)})$, we have $h_e = h_e^{(i)}, C_{\ell_e+1} = C_{\ell_e^{(i)}+1}, \sigma_o = \sigma_o^{(i)}$. Since H_e is collision resistant, we have $\breve{c} = \breve{c}_i, \sigma_w = \sigma_w^{(i)}$, $\varrho_0 = \varrho_0^{(i)}$ and which implies $\breve{d} = \breve{d}_i$ and using $\breve{c} = \breve{c}_i$, we have $m = m^{(i)}$. All together, we have $(\Upsilon, m, \Gamma_s, \Gamma_e) = (\Upsilon_i, m^{(i)}, \Gamma_s^{(i)}, \Gamma_e^{(i)})$, which is contradiction to the definition of strong existential unforgeability of CP-ABSC scheme.

D Mechanism for Full Construction

Although the technique is available in [LOS+10] but for self-containment, in this section we briefly demonstrate it. The mechanism described here is for both CP-ABS and CP-ABSC supporting MSPs. For full 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 $\mathfrak{M}$ time and the size of the universe $\mathcal{U}$ is n, then the size of new universe $\mathcal{U}'$ for the full construction will be $n\mathfrak{M}$. Basically in this full construction, for each attribute $\chi \in \mathcal{U}$, we consider $\mathfrak{M}$ copies of χ in $\mathcal{U}'$. To enumerate each copy, we assign a label say j to the attribute say χ, i.e., $\mathcal{U}' := \{(\chi, j) | \chi \in \mathcal{U}, j \in [\mathfrak{M}]\}$. Similarly, for any access policy $\Gamma := (M, \rho)$ 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 $\mathcal{U}$, then $A' := \{(\chi, j) | \chi \in A, j \in [\mathfrak{M}]\}$ is the set of attributes for $\mathcal{U}'$. Then, we have that the set of attributes A satisfies the policy (M, ρ) if and only if A' satisfies (M, ρ'). Due to this technique, the sizes of public parameters and key increase by a factor linear to $\mathfrak{M}$, but the sizes of signature (resp. signcryption) and the cost of sign and ver (resp. signcrypt and unsigncrypt) for CP-ABS (resp. CP-ABSC) remain unchanged.

E Strong Unforgeability in presence of Unsigncryption Oracle

The strong existential unforgeability of the published version (in ProvSec, 2014) of this paper was proven without giving the unsigncryption oracle access to the adversary $\mathscr{A}$. In this full version, we include the proof of strong unforgeability, where $\mathscr{A}$ is provided the access to unsigncryption oracle.

The proof style is just an extension of the unforgeability proof of CP-ABS (section 4), where a verification text (for verifying the forgery) is changed to a sf-type 2 with respect to which the forgery will be invalid. The oracle queries are handled in similar manner as in confidentiality proof of CP-ABSC (section 6.2). We will be considering two forms of semi-functional key, viz, sf-type I and sf-type II. Each of the following stuffs has two forms, sf-type 1 and sf-type 2: verification text (involved for checking signature part of the forge), unsigncryption-query key and signcryption. Below, we only define the sf-type 1 and sf-type 2 vText as rest are similar to that of section 6.2.

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

$$\vec{V}_0 := \left(g^s \boxed{g_2^c}, (u_s^{h_s} v_s)^s \boxed{g_2^\iota}, \ g_T^{\alpha s} \right)$$

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

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

$$\vec{V}_0 := \left(g^s g_2^c, (u_s^{h_s} v_s)^s g_2^\iota, \boxed{\hat{g}_t} \right), \text{ where } \hat{g}_t \xleftarrow{\text{U}} \mathbb{G}_T$$

Theorem E.1. *If DSG1, DSG2 and DSG3 assumptions hold for $\mathcal{J}$, Π_{OTS} is a strong unforgeable one-time signature scheme and H_s, H_e are collision resistant hash functions, then the proposed basic CP-ABSC scheme in section 7 is strong existential unforgeable.*

Proof. Suppose an adversary $\mathscr{A}$ can break the adaptive-predicates strong existential unforgeability of the proposed CP-ABSC scheme with non-negligible advantage ϵ. Lets assume that $\mathscr{A}$ has made ν_3 number of signcryption query to the signcryption oracle. Let $(m^{(i)}, \Gamma_s^{(i)}, \Gamma_e^{(i)})$ be the i^{th} query and $\Upsilon_i := (\breve{c}_i, \sigma_s^{(i)} := (\sigma_w^{(i)}, \sigma_o^{(i)}, \mathsf{verk}^{(i)}), \varrho^{(i)})$ be the corresponding replied signcryption. Let $\Upsilon := (\breve{c}, \sigma_s := (\sigma_w, \sigma_o, \mathsf{verk}), \varrho)$ be the forge by $\mathscr{A}$ for the message (m, Γ_s, Γ_e). We define an event as

$$\mathsf{Forged} := \mathsf{verk} \notin \{\mathsf{verk}^{(i)} | \ i \in [\nu_3]\}$$

Then, we have

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

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

Case Forged : Suppose there are at most ν_1 key queries and ν_2 unsigncryption queries, then the security proof consists of hybrid argument over a sequence of $2(\nu_1+\nu_2+\nu_3)+4$ games. The games are defined below:

- Game$_{Real}$:= The original APs-sUF-CMA security game of CP-ABSC.
- Game$_{Real'}$:= This is same as game Game$_{Real}$ except the event Forged is always happened.

- Game$_0$ (= Game$_{1-0-2}$) is just like Game$_{Real'}$ except that the vText is of sf-type 1.
- In Game$_{1-k-1}$ (for $1 \leq k \leq \nu_1$), vText is sf-type 1, all the unsigncryption queries are answered by normal uq-Key, all the replied signcryptions are normal, the first $(k-1)$ keys returned to the adversary are sf-type 2, k^{th} key is sf-type 1 and the rest keys are normal.
- 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 $1 \leq k \leq \nu_2$), vText is sf-type 1, all the replied signcryptions are normal, all the keys are sf-type 2, the first $(k-1)$ unsigncryption queries are answered by sf-type 2 uq-keys, k^{th} unsigncryption query is answered by sf-type 1 uq-key and the rest are answered by normal uq-keys. (So, in this sequel Game$_{2-0-2}$ = 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} unsigncryption query is answered by sf-type 2 uq-key.
- In Game$_{3-k-1}$ (for $1 \leq k \leq \nu_3$), vText is sf-type 1, all the keys are sf-type 2, all the unsigncryption queries are answered by sf-type 2 uq-keys, the first $(k-1)$ replied signcryptions are of sf-type II, the k^{th} replied signcryption is sf-type I and the rest are normal signcryption. (So, in this sequel Game$_{3-0-2}$ = 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 sf-type II.
- Game$_{Final}$ is similar to Game$_{3-\nu_3-2}$ except that the vText is of sf-type 2.

Let $\mathsf{Adv}^{\text{Real}}_{\mathscr{A},\text{ABSC}}(\kappa)$, $\mathsf{Adv}^{\text{Real}'}_{\mathscr{A},\text{ABSC}}(\kappa)$, $\mathsf{Adv}^{0}_{\mathscr{A},\text{ABSC}}(\kappa)$, $\mathsf{Adv}^{1-k-1}_{\mathscr{A},\text{ABSC}}(\kappa)$, $\mathsf{Adv}^{1-k-2}_{\mathscr{A},\text{ABSC}}(\kappa)$, $\mathsf{Adv}^{2-i-1}_{\mathscr{A},\text{ABSC}}(\kappa)$, $\mathsf{Adv}^{2-i-2}_{\mathscr{A},\text{ABSC}}(\kappa)$, $\mathsf{Adv}^{3-j-1}_{\mathscr{A},\text{ABSC}}(\kappa)$, $\mathsf{Adv}^{3-j-2}_{\mathscr{A},\text{ABSC}}(\kappa)$ and $\mathsf{Adv}^{\text{Final}}_{\mathscr{A},\text{ABSC}}(\kappa)$ denote the advantages of an adversary $\mathscr{A}$ in Game$_{Real}$, Game$_{Real'}$, Game$_0$, Game$_{1-k-1}$, Game$_{1-k-2}$, Game$_{2-i-1}$, Game$_{2-i-2}$, Game$_{3-j-1}$, Game$_{3-j-2}$ and Game$_{Final}$ for $1 \leq k \leq \nu_1$, $1 \leq i \leq \nu_2$, $1 \leq j \leq \nu_3$ respectively. In Game$_{Final}$, the part, V_{03} of $\vec{V}_0$ is chosen independently and uniformly random from $\mathbb{G}_T$ implying that each forge will be invalid with respect to the vText. Therefore, the adversary $\mathscr{A}$ has no advantage in Game$_{Final}$.

Using lemmas E.2, E.3, E.4, E.5, E.6, E.7, E.8 and E.9, we have the following reduction

$$\begin{aligned}
\frac{1}{2}.\mathsf{Adv}_{\mathscr{A}}^{\mathrm{ABSC-EUF}}(\kappa) = \frac{1}{2}.\mathsf{Adv}_{\mathscr{A},\mathrm{ABSC}}^{\mathrm{Real}}(\kappa) &\leq \mathsf{Adv}_{\mathscr{A},\mathrm{ABSC}}^{\mathrm{Real}'}(\kappa) \\
&\leq |\mathsf{Adv}_{\mathscr{A},\mathrm{ABSC}}^{\mathrm{Real}'}(\kappa) - \mathsf{Adv}_{\mathscr{A},\mathrm{ABSC}}^{0}(\kappa)| \\
&\quad + \sum_{k=1}^{\nu_1}(|\mathsf{Adv}_{\mathscr{A},\mathrm{ABSC}}^{1-(k-1)-2}(\kappa) - \mathsf{Adv}_{\mathscr{A},\mathrm{ABSC}}^{1-k-1}(\kappa)| + |\mathsf{Adv}_{\mathscr{A},\mathrm{ABSC}}^{1-k-1}(\kappa) - \mathsf{Adv}_{\mathscr{A},\mathrm{ABSC}}^{1-k-2}(\kappa)|) \\
&\quad + \sum_{k=1}^{\nu_2}(|\mathsf{Adv}_{\mathscr{A},\mathrm{ABSC}}^{2-(k-1)-2}(\kappa) - \mathsf{Adv}_{\mathscr{A},\mathrm{ABSC}}^{2-k-1}(\kappa)| + |\mathsf{Adv}_{\mathscr{A},\mathrm{ABSC}}^{2-k-1}(\kappa) - \mathsf{Adv}_{\mathscr{A},\mathrm{ABSC}}^{2-k-2}(\kappa)|) \\
&\quad + \sum_{k=1}^{\nu_3}(|\mathsf{Adv}_{\mathscr{A},\mathrm{ABSC}}^{3-(k-1)-2}(\kappa) - \mathsf{Adv}_{\mathscr{A},\mathrm{ABSC}}^{3-k-1}(\kappa)| + |\mathsf{Adv}_{\mathscr{A},\mathrm{ABSC}}^{3-k-1}(\kappa) - \mathsf{Adv}_{\mathscr{A},\mathrm{ABSC}}^{3-k-2}(\kappa)|) \\
&\quad + |\mathsf{Adv}_{\mathscr{A},\mathrm{ABSC}}^{3-\nu_3-2}(\kappa) - \mathsf{Adv}_{\mathscr{A},\mathrm{ABSC}}^{\mathrm{Final}}(\kappa)| \\
&\leq \mathsf{Adv}_{\mathscr{B}_0}^{\mathrm{DSG1}}(\kappa) + \sum_{k=1}^{\nu_1}(\mathsf{Adv}_{\mathscr{B}_{1-k-1}}^{\mathrm{DSG2}}(\kappa) + \mathsf{Adv}_{\mathscr{B}_{1-k-2}}^{\mathrm{DSG2}}(\kappa)) \\
&+ \sum_{k=1}^{\nu_2}(\mathsf{Adv}_{\mathscr{B}_{2-k-1}}^{\mathrm{DSG2}}(\kappa) + \mathsf{Adv}_{\mathscr{B}_{2-k-2}}^{\mathrm{DSG2}}(\kappa)) \\
&+ \sum_{k=1}^{\nu_3}(\mathsf{Adv}_{\mathscr{B}_{3-k-1}}^{\mathrm{DSG2}}(\kappa) + \mathsf{Adv}_{\mathcal{H}_{3-k-1}}^{CR-H_s}(\kappa) + \mathsf{Adv}_{\mathscr{B}_{3-k-2}}^{\mathrm{DSG2}}(\kappa)) + \mathsf{Adv}_{\mathscr{B}_4}^{\mathrm{DSG3}}(\kappa)
\end{aligned}$$

where $\mathscr{B}_0, \mathscr{B}_{1-k-1}, \mathscr{B}_{1-k-2}, \mathscr{B}_{2-k-1}, \mathscr{B}_{2-k-2}, \mathscr{B}_{3-k-1}, \mathcal{H}_{3-k-1}, \mathscr{B}_{3-k-2}$ and $\mathscr{B}_4$ are PPT algorithms whose running times are same as that of $\mathscr{A}$.

Case $\neg\mathsf{Forged}$: This case is similar to that of Theorem 7.4. We note that in this case the simulator can handle the unsigncryption queries as it knows the $\mathcal{MSK}$.

□

The following lemmas can be proven similarly to that of CP-ABS and CP-ABSC.

Lemma E.2. $\mathrm{Game}_{Real'}$ *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* $|\mathsf{Adv}_{\mathscr{A},\mathrm{ABSC}}^{\mathrm{Real}'}(\kappa) - \mathsf{Adv}_{\mathscr{A},\mathrm{ABSC}}^{0}(\kappa)| \leq \mathsf{Adv}_{\mathscr{B}}^{\mathrm{DSG1}}(\kappa)$.

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

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

Lemma E.5. $\mathrm{Game}_{2-(k-1)-2}$ *and* Game_{2-k-1} *are indistinguishable under the 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* $|\mathsf{Adv}_{\mathscr{A},\mathrm{ABSC}}^{2-(k-1)-2}(\kappa) - \mathsf{Adv}_{\mathscr{A},\mathrm{ABSC}}^{2-k-1}(\kappa)| \leq \mathsf{Adv}_{\mathscr{B}}^{\mathrm{DSG2}}(\kappa)$ *for* $1 \leq k \leq \nu_2$.

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

Lemma E.7. $\mathrm{Game}_{3-(k-1)-2}$ *and* Game_{3-k-1} *are indistinguishable under the DSG2 assumption and collision resistant property of H_s. That is, for every adversary $\mathscr{A}$ there exists PPT algorithms, $\mathscr{B}$ and $\mathcal{H}$ such that* $|\mathsf{Adv}^{3-(k-1)-2}_{\mathscr{A},\mathrm{ABSC}}(\kappa) - \mathsf{Adv}^{3-k-1}_{\mathscr{A},\mathrm{ABSC}}(\kappa)| \leq \mathsf{Adv}^{\mathrm{DSG2}}_{\mathscr{B}}(\kappa) + \mathsf{Adv}^{CR-H_s}_{\mathcal{H}}(\kappa)$ *for* $1 \leq k \leq \nu_3$.

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

Lemma E.9. $\mathrm{Game}_{3-\nu_3-2}$ *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* $|\mathsf{Adv}^{3-\nu_3-2}_{\mathscr{A},\mathrm{ABSC}}(\kappa) - \mathsf{Adv}^{\mathrm{Final}}_{\mathscr{A},\mathrm{ABSC}}(\kappa)| \leq \mathsf{Adv}^{\mathrm{DSG}}_{\mathscr{B}}(\kappa)$