Compact Attribute-Based Encryption and Signcryption for General Circuits from Multilinear Maps[*]

Pratish Datta, Ratna Dutta, and Sourav Mukhopadhyay

Department of Mathematics
Indian Institute of Technology Kharagpur
Kharagpur-721302, India
{pratishdatta,ratna,sourav}@maths.iitkgp.ernet.in

Abstract. Designing attribute-based systems supporting highly expressive access policies has been one of the principal focus of research in attribute-based cryptography. While *attribute-based encryption* (ABE) enables fine-grained access control over encrypted data in a multi-user environment, *attribute-based signature* (ABS) provides a powerful tool for preserving signer anonymity. *Attribute-based signcryption* (ABSC), on the other hand, is a combination of ABE and ABS into a unified cost-effective primitive. In this paper, we start by presenting a *key-policy* ABE supporting *general polynomial-size circuit* realizable decryption policies and featuring *compactness*. More specifically, our ABE construction exhibits *short* ciphertexts and *shorter* decryption keys compared to existing similar works. We then proceed to design a *key-policy* ABSC scheme which enjoys several interesting properties that were *never* achievable *before*. It supports *arbitrary polynomial-size circuits*, thereby handles highly sophisticated control over signing and decryption rights. Besides, it generates *short* ciphertext as well. Our ABE construction employs multilinear map of level $n + l + 1$, while that used for our ABSC scheme has level $n + n' + l + 1$, where n, n', and l represent respectively the input length of decryption policy circuits, input size of signing policy circuits, and depth of both kinds of circuits. *Selective* security of our constructions are proven in the *standard model* under the *Multilinear Decisional Diffie-Hellman* and *Multilinear Computational Diffie-Hellman* assumptions which are *standard* complexity assumptions in the multilinear map setting. Our key-policy constructions can be converted to the corresponding *ciphertext-policy* variants achieving *short* ciphertext by utilizing the technique of *universal circuits*.

Keywords: ABE for circuits, ABSC for circuits, polynomial-size circuits, multilinear map.

1 Introduction

ABE: The recent advancements in online social networks and cloud technology have triggered an emerging trend among individuals and organizations to outsource potentially sensitive private data to external servers. This necessitates enforcing sophisticated access control while sharing the outsourced data with other individuals or organizations. *Attribute-based encryption* (ABE), introduced by Sahai and Waters [SW05], offers a natural solution to the above scenario by enabling fine-grained management of decryption rights to encrypted data. ABE comes in two flavors, namely, *key-policy* and *ciphertext-policy*. In the key-policy version of ABE, a ciphertext encrypts a message M with respect to a public vector, called an encryption input string, x of Boolean variables representing a set of descriptive attributes. An entity can obtain a private decryption key $\mathsf{SK}_f^{(\mathsf{DEC})}$ from a trusted key generation center only if the center deems that the requesting entity is entitled to possess the key. Here, f is some Boolean function belonging to some class of allowable Boolean functions expressing decryption policies. $\mathsf{SK}_f^{(\mathsf{DEC})}$ can be utilized to recover the message M from the ciphertext if and only if $f(x) = 1$. In a ciphertext-policy ABE system, the roles of f and x are reversed: They are associated to ciphertexts and decryption keys respectively.

[*] This is the full version of the paper that appeared in Proceedings of the 16th Progress in Cryptology Conference (INDOCRYPT 2015), LNCS 9462, pp. 3–24, Springer.

A central theme of research in the field of ABE has been to expand the class of admissible decryption policies in view of implementing ABE in more complicated settings. However, until recently candidate constructions of ABE were limited to restricted class of decryption policies, namely, *polynomial-size Boolean formulas* or *circuits with fan-out one* [GPSW06], [Wat11]. In 2013, the independent breakthrough works due to Garg et al. [GGH^{+}13b] and Gorbunov et al. [GVW13] on ABE systems were able to realize decryption policies representable as *polynomial-size Boolean circuits with arbitrary fan-out.* Besides, tackling the issue of complex access control in diverse multi-user digital communication and storage systems, ABE for general circuits has found countless applications in cryptography, most notably for publicly verifiable two message delegation scheme with a preprocessing phase [PRV12], succinct one-query functional encryption [GKP^{+}13a], reuse garbled circuits [BGG^{+}14], token-based obfuscation [GKP^{+}13a], and homomorphic encryption for Turing machines [GKP^{+}13b].

Following Garg et al. [GGH^{+}13b] and Gorbunov et al. [GVW13], a series of distinguished works have contributed in making ABE for general circuits more practical in terms of both efficiency and security [BGG^{+}14], [GGHZ14], [Att14]. The ABE schemes of [GVW13], [BGG^{+}14] are based on lattices, while others [GGH^{+}13b], [GGHZ14], [Att14] employ multilinear map [GGH13a], [CLT13], [CLT15]. However, in all the existing ABE constructions supporting arbitrary circuits (except the multilinear map-based scheme of [BGG^{+}14]), the number of ciphertext components proliferates with (I) the number of associated Boolean variables [GGH^{+}13b], [Att14], (II) depth of the decryption policy circuits [GVW13], or (III) size of the decryption policy circuits [GGHZ14]. The ABE scheme from multilinear map developed in [BGG^{+}14] achieves short ciphertext but the number of decryption key components grows quadratically with the input length of the decryption policy circuits. This can be a serious bottleneck if the allowable decryption policy circuits have large input size. Moreover, the security of this construction is derived from the *Multilinear Diffie-Hellman Exponent* assumption which holds in *generic multilinear map framework.*

ABSC: *Attribute-based signcryption* (ABSC) is a logical mixture of attribute-based encryption (ABE) and *attribute-based signature* (ABS) into an unified cost-effective primitive. ABS aims to allow signers to preserve their anonymity while signing digital documents. As for ABE, ABS is also classified into *key-policy* and *ciphertext-policy* variants. In a key-policy ABS, a message M is signed relative to a public vector, known as a signature input string, y of Boolean variables representing certain authorization credentials using a signing key $\mathsf{SK}_g^{(\mathsf{SIG})}$ obtained from a trusted key generation center. Here, g is some Boolean function from a designated class of permissable Boolean functions characterizing signing capabilities such that $g(y) = 1$. A verifier is only assured of the validity of such an endorsement. The cipertext-policy variant interchanges the roles of g and y in an analogous fashion as for ABE. Being a combination of ABE and ABS, ABSC also inherits a categorization similar to that of ABE and ABS.

ABSC resolves the issue of managing sophisticated authentication and decryption rights simultaneously in large distributed networks with better efficiency compared to a sequential implementation of ABE and ABS. For instance, in cloud-based data sharing systems, storing sensitive information securely to the cloud may not be sufficient. The data owner should also be able to prove its genuineness at the cloud as well as to the data recipients to avoid illegal data storage by the cloud server.

A desirable property of an ABSC scheme is *public verifiability* meaning that any party can verify the authenticity of a ciphertext even without the knowledge of the signcrypted message or a valid decryption key. This feature is especially appealing in real-life applications such as filtering out the spams in secure email systems. Here, a spam filter can check whether a signcrypted email is generated from a source with claimed credentials or not before sending to inbox, without knowing the original message. If an email does not satisfy the public verifiability mechanism, it can be treated as spam and can be sent to the spam folder.

Designing efficient ABSC schemes for highly expressive signing and decryption policies is a challenging task and have received considerable attention to the recent research community [GNSN10], [EMR12], [WHL14], [RD14a], [RD14b], [WH11]. Further, the works of [GNSN10], [RD14a], [RD14b] achieve public verifiability. However, the most significant drawback of all the aforementioned ABSC schemes is that the classes of admissible signing and decryption policies have been restricted to *circuits of fan-out one*. As explained in [GGH$^+$13b] for ABE, "backtracking" attack can be mounted on the signing and decryption policy circuits respectively to forge a signcryption and break the confidentiality of a ciphertext if the current ABSC constructions are applied for circuits of fan-out greater than one.

Our Contribution: In this paper, we propose two attribute-based cryptographic constructions:

– A key-policy ABE scheme supporting *arbitrary polynomial-size circuits* with *short ciphertext* and *shorter decryption keys* compared to existing similar works under *standard* complexity assumption.
– The *first* key-policy ABSC scheme for *general circuits of polynomial-size* achieving *public verifiability* and featuring *compact ciphertext* as well.

More precisely, similar to [GGH$^+$13b], [GVW13], [BGG$^+$14], [Att14], our ABE construction permits circuits of arbitrary polynomial-size and unbounded fan-out with bounded depth and input sizes that are fixed at the setup. We develop our ABE scheme in current multilinear map setting [GGH13a], [CLT13], [CLT15] with multilinearity level $n + l + 1$, where n and l denote respectively the input length and depth of the decryption policy circuits. To realize short ciphertext, we adopt the technique of [HSW13] in developing a full domain hash from multilinear map. Specifically, our encryption procedure computes a "hash value" of the encryption input string using multilinear map and includes that hash value within the factor used to mask the message, thereby compressing the ciphertext to involve only two group elements (or encodings). The structure of our decryption keys is similar to that of [GGH$^+$13b], [BGG$^+$14] except that the key components corresponding to the input wires of the decryption policy circuits are suitably modified. This enables us to apply the "move forward and shift" mechanism of [GGH$^+$13b], [BGG$^+$14] in decrypting our newly structured ciphertext. More interestingly, this modification is in favor of shortening of the decryption key size. Only a single group element is sufficient to be provided for each input wire of the decryption policy circuits. This is less than all previous multilinear map-based constructions.

We prove *selective* security of our ABE construction against *chosen plaintext attack* (CPA) under the *Multilinear Decisional Diffie-Hellman* assumption. This is a standard complexity assumption in the sense that its validity in current multilinear map candidates has been justified by means of rigorous cryptanalysis rather than using the folklore generic multilinear map framework. A standard complexity leveraging argument, as in [BB11], can be applied to our selectively secure construction in order to realize adaptive security. Note that the recent improved multilinear map candidate proposed in [CLT15] claims to fix all known attacks against the previous candidates [GGH13a], [CLT13], especially the "zeroizing attack" due to Cheon et al. [CHL$^+$14], and, to the best of our knowledge, no further attack against this new candidate has been reported till date. Thus, one can securely instantiate schemes based on Multilinear Decisional Diffie-Hellman assumption using this multilinear map candidate.

The second and more significant contribution of this paper is an ABSC scheme of the key-policy category. This scheme also supports signing and decryption policies realizable by polynomial-size circuits of arbitrary fan-out having bounded depths and input lengths. This scheme is developed by augmenting our ABE construction with an attribute-based authentication functionality. We utilize a multilinear map of multilinearity level $n + n' + l + 1$, where n, n', and l represent respectively the input length of decryption policy circuits, input size of signing policy circuits, and depth of both types of circuits. Roughly speaking, our ABSC construction works as follows: We break the master secret exponent into two parts and provide an encoding of one part in appropriate multilinear group while embed the other part within the signing keys of the signcrypters. In order to pool these two segments to reconstruct the master secret exponent and compute a valid signcryption of some message, the signcrypter should possess a signing key associated with some signing policy circuit that evaluates to 1 on the claimed signature input string, against which the authenticity of the ciphertext would be verified. The unsigncryption procedure is similar to the decryption algorithm of our ABE construction except with an added check step for authenticity verification. We emphasize that the validity confirming step can be performed utilizing only publicly available information empowering our scheme with the public verifiability feature.

Our ABSC construction is proven *selectively message confidential* against *chosen-plaintext attack* (CPA) and *selectively ciphertext unforgeable* against *chosen message attack* (CMA) under the *Multilinear Decisional Diffie-Hellman* and *Multilinear Computational Diffie-Hellman* assumption respectively. The number of group elements comprising our ABSC ciphertext is also constant– 3 to be exact.

Finally, we demonstrate in Appendix C that using the technique of universal circuits, as in [GGH$^+$13b], [Att14], both of our constructions can be utilized to realize their corresponding ciphertext-policy variants for arbitrary bounded-size circuits featuring short ciphertext as well.

2 Preliminaries

2.1 Circuit Notation

We adopt the same notations for circuits as in [GGH$^+$13b], [Att14]. First note that without loss of generality we can consider only those circuits which are *monotone*, where gates are either OR or AND having fan-in two, and *layered* since, as mentioned in [GGH$^+$13b] using De Morgan's law one can build a general circuit from a monotone circuit with negation only appearing at the input wires and an arbitrary circuit can be transformed into a layered one for the same function with a small amount of overhead. Our circuits will have a single output gate. A circuit will be represented as a six-tuple $f = (n, q, l, \mathbb{A}, \mathbb{B}, \mathsf{GateType})$. Here, n and q respectively denote the length of the input and the number of gates, while l represents the depth of the circuit which is one plus the length of the shortest path from the output wire to any input wire. We designate the set of input wires as $\mathsf{Input} = \{1, \ldots, n\}$, the set of gates as $\mathsf{Gates} = \{n+1, \ldots, n+q\}$, the total set of wires in the circuit as $W = \mathsf{Input} \cup \mathsf{Gates} = \{1, \ldots, n+q\}$, and the wire $n+q$ to be the output wire. Let $\mathbb{A}, \mathbb{B} : \mathsf{Gates} \to W \backslash \{n+q\}$ be functions where for all $w \in \mathsf{Gates}$, $\mathbb{A}(w)$ and $\mathbb{B}(w)$ respectively identify w's first and second incoming wires. Finally, $\mathsf{GateType} : \mathsf{Gates} \to \{\mathsf{AND}, \mathsf{OR}\}$ defines a functions that identifies a gate as either an AND or an OR gate. We follow the convention that $w > \mathbb{B}(w) > \mathbb{A}(w)$ for any $w \in \mathsf{Gates}$.

We also define a function $\mathsf{depth} : W \to \{1, \ldots, l\}$ such that if $w \in \mathsf{Inputs}$, $\mathsf{depth}(w) = 1$, and in general $\mathsf{depth}(w)$ of wire w is equal to one plus the length of the shortest path from w to an input wire. Since our circuit is layered, we have, for all $w \in \mathsf{Gates}$, if $\mathsf{depth}(w) = j$ then $\mathsf{depth}(\mathbb{A}(w)) = \mathsf{depth}(\mathbb{B}(w)) = j - 1$.

We will abuse notation and let $f(x)$ be the evaluation of the circuit f on input $x \in \{0,1\}^n$, and $f_w(x)$ be the value of wire w of the circuit f on input x.

2.2 The Notion of ABE for General Circuits

- **Syntax of ABE for circuits**: Consider a circuit family $\mathbb{F}_{n,l}$ that consists of all circuits f with input length n and depth l characterizing decryption rights. A key-policy attribute-based encryption (ABE) scheme for circuits in $\mathbb{F}_{n,l}$ with message space $\mathbb{M}$ consists of the following algorithms:

$\mathsf{ABE.Setup}(1^\lambda, n, l)$: The trusted key generation center takes as input a security parameter 1^λ, the length n of Boolean inputs to decryption policy circuits, and the allowed depth l of the decryption policy circuits. It publishes the public parameters PP, while keeps the master secret key MK to itself.

$\mathsf{ABE.KeyGen}(\mathsf{PP}, \mathsf{MK}, f)$: On input the public parameters PP, the master secret key MK, and the description of a decryption policy circuit $f \in \mathbb{F}_{n,l}$ from a decrypter, the key generation center provides a decryption key $\mathsf{SK}_f^{(\mathsf{DEC})}$ to the decrypter.

$\mathsf{ABE.Encrypt}(\mathsf{PP}, x, M)$: Taking as input the public parameters PP, an encryption input string $x \in \{0,1\}^n$, and a message $M \in \mathbb{M}$, the encrypter prepares a ciphertext CT_x.

$\mathsf{ABE.Decrypt}(\mathsf{PP}, \mathsf{CT}_x, \mathsf{SK}_f^{(\mathsf{DEC})})$: A decrypter takes as input the public parameters PP, a ciphertext CT_x encrypted for x, and its decryption key $\mathsf{SK}_f^{(\mathsf{DEC})}$ corresponding to circuit $f \in \mathbb{F}_{n,l}$. It attempts to decrypt the ciphertext and outputs the message $M \in \mathbb{M}$ if successful; otherwise, it outputs the distinguished symbol $\perp$.

- **Correctness**: Consider all messages $M \in \mathbb{M}$, encryption input strings $x \in \{0,1\}^n$, and decryption policy circuit $f \in \mathbb{F}_{n,l}$ such that $f(x) = 1$, i.e., f evaluated on input x outputs 1. If $\mathsf{ABE.Encrypt}(\mathsf{PP}, x, M)$ outputs CT_x and $\mathsf{ABE.KeyGen}(\mathsf{PP}, \mathsf{MK}, f)$ generates $\mathsf{SK}_f^{(\mathsf{DEC})}$, where PP, MK are created by $\mathsf{ABE.Setup}(1^\lambda, n, l)$, then $\mathsf{ABE.Decrypt}(\mathsf{PP}, \mathsf{CT}_x, \mathsf{SK}_f^{(\mathsf{DEC})})$ outputs M.
- **Security Definition**: The security notion for ABE supporting general circuits is presented in Appendix A.1.

2.3 The Notion of ABSC for General Circuits

- **Syntax of ABSC for circuits**: Consider a circuit family $\mathbb{F}_{n,l}^{(\mathsf{DEC})}$ consisting of all circuits f with input length n and depth l expressing decryption access structures along with a circuit class $\mathbb{F}_{n',l}^{(\mathsf{SIG})}$ containing all circuits g of input length n' and depth l characterizing signing rights. A key-policy attribute-based signcryption (ABSC) scheme for circuits in $\mathbb{F}_{n,l}^{(\mathsf{DEC})}$ and $\mathbb{F}_{n',l}^{(\mathsf{SIG})}$ with message space $\mathbb{M}$ consists of the following algorithms:

$\mathsf{ABSC.Setup}(1^\lambda, n, n', l)$: The trusted key generation center takes as input a security parameter 1^λ, the length n of Boolean inputs to decryption policy circuits, the length n' of Boolean inputs to signing policy circuits, and the common allowed depth l of both types of circuits. It publishes the public parameters PP and keeps the master secret key MK to itself.

$\mathsf{ABSC.SKeyGen}(\mathsf{PP}, \mathsf{MK}, g)$: On input the public parameters PP, the master secret key MK, and the description of a signing policy circuit $g \in \mathbb{F}_{n',l}^{(\mathsf{SIG})}$ from a signcrypter, the key generation center provides a signing key $\mathsf{SK}_g^{(\mathsf{SIG})}$ to the signcrypter.

$\mathsf{ABSC.DKeyGen}(\mathsf{PP}, \mathsf{MK}, f)$: Taking as input the public parameters PP, the master secret key MK, and the description a decryption policy circuit $f \in \mathbb{F}_{n,l}^{(\mathsf{DEC})}$ from a decrypter, the key generation center hands a decryption key $\mathsf{SK}_f^{(\mathsf{DEC})}$ to the decrypter.

$\mathsf{ABSC.Signcrypt}(\mathsf{PP}, \mathsf{SK}_g^{(\mathsf{SIG})}, x, y, M)$: A signcrypter takes as input the public parameters PP, its signing key $\mathsf{SK}_g^{(\mathsf{SIG})}$ corresponding to some circuit $g \in \mathbb{F}_{n',l}^{(\mathsf{SIG})}$, an encryption input string $x \in \{0,1\}^n$ describing a set of legitimate decrypter, a signature input string $y \in \{0,1\}^{n'}$ such that $g(y) = 1$, and a message $M \in \mathbb{M}$. It outputs a ciphertext $\mathsf{CT}_{x,y}$.

$\mathsf{ABSC.Unsigncrypt}(\mathsf{PP}, \mathsf{CT}_{x,y}, \mathsf{SK}_f^{(\mathsf{DEC})})$: A decrypter takes as input the public parameters PP, a ciphertext $\mathsf{CT}_{x,y}$ signcrypted with x, y, and its decryption key $\mathsf{SK}_f^{(\mathsf{DEC})}$ corresponding to circuit $f \in \mathbb{F}_{n,l}^{(\mathsf{DEC})}$. It attempts to unsigncrypt the ciphertext and outputs the message $M \in \mathbb{M}$ if successful; otherwise, it outputs $\perp$ indicating that either the ciphertext is invalid or the ciphertext cannot be decrypted.

- **Correctness**: Consider all messages $M \in \mathbb{M}$, encryption input string $x \in \{0,1\}^n$, signature input string $y \in \{0,1\}^{n'}$, and decryption policy circuit $f \in \mathbb{F}_{n,l}^{(\mathsf{DEC})}$ satisfying $f(x) = 1$. If $\mathsf{ABE.Signcrypt}(\mathsf{PP}, \mathsf{SK}_g^{(\mathsf{SIG})}, x, y, M)$ with $g(y) = 1$ for some $g \in \mathbb{F}_{n',l}^{(\mathsf{SIG})}$ outputs $\mathsf{CT}_{x,y}$ while $\mathsf{ABSC.DKeyGen}(\mathsf{PP}, \mathsf{MK}, f)$ forms $\mathsf{SK}_f^{(\mathsf{DEC})}$, where $\mathsf{SK}_g^{(\mathsf{SIG})}$ is generated by $\mathsf{ABSC.SKeyGen}(\mathsf{PP}, \mathsf{MK}, g)$ and PP, MK are outputted by $\mathsf{ABSC.Setup}(1^\lambda, n, n', l)$, then $\mathsf{ABSC.Unsigncrypt}(\mathsf{PP}, \mathsf{CT}_{x,y}, \mathsf{SK}_f^{(\mathsf{DEC})})$ outputs M.

- **Security Definitions**: An ABSC scheme has two security requirements, namely, (I) *message confidentiality* and (II) *ciphertext unforgeability* which are described in Appendix B.1.

2.4 Multilinear Maps and Complexity Assumption

Here we review multilinear maps [BS03], [CLT13], [GGH13a]. A (leveled) multilinear map consists of the following two algorithms:

(i) $\mathcal{G}(1^\lambda, k)$: It takes as input a security parameter 1^λ and a positive integer k indicating the number of allowed pairing operations. It outputs a sequence of groups $\vec{\mathbb{G}} = (\mathbb{G}_1, \ldots, \mathbb{G}_k)$ each of large prime order $p > 2^\lambda$ together with the canonical generators g_i of $\mathbb{G}_i$. We call $\mathbb{G}_1$ the source group, $\mathbb{G}_k$ the target group, and $\mathbb{G}_2, \ldots, \mathbb{G}_{k-1}$ intermediate groups.

(ii) $e_{i,j}(g, h)$ (for $i, j \in \{1, \ldots, k\}$ with $i + j \leq k$): On input two elements $g \in \mathbb{G}_i$ and $h \in \mathbb{G}_j$ with $i + j \leq k$, it outputs an element of $\mathbb{G}_{i+j}$ such that $e_{i,j}(g_i^a, g_j^b) = g_{i+j}^{ab}$ for $a, b \in \mathbb{Z}_p$. We often omit the subscripts and just write e. We can also generalize e to multiple inputs as $e(h^{(1)}, \ldots, h^{(t)}) = e(h^{(1)}, e(h^{(2)}, \ldots, h^{(t)}))$.

We refer g_i^a as a level-i encoding of $a \in \mathbb{Z}_p$. The scalar a itself is referred to as a level-0 encoding of a. Then the map e combines a level-i encoding of an element $a \in \mathbb{Z}_p$ and a level-j encoding of another element $b \in \mathbb{Z}_p$, and produces level-$(i + j)$ encoding of the product ab.

We note that current candidates of multilinear maps, also known as graded encoding systems (GES) [GGH13a], [CLT13], [CLT15], depart from the ideal notions of multilinear maps described above. In particular, in these candidates representations of group elements are not unique and contain a noise term that can cause errors during group and multilinear operations. Although we present our ABE and ABSC constructions using ideal multilinear maps for simplicity, our constructions can be instantiated using current non-ideal candidates of multilinear map in a manner analogous to [GGH$^+$13b], [BGG$^+$14].

Assumption 1 (k-Multilinear Decisional Diffie-Hellman: k-MDDH [GGH13a]). *The k-Multilinear Decisional Diffie-Hellman (k-MDDH) problem is to guess $\bar{b} \in \{0,1\}$ given $\varrho_{\bar{b}} =$*

$(\vec{\mathbb{G}}, g_1, \overline{S}, C_1, \ldots, C_k, T_{\overline{b}})$ *generated by* $\mathcal{G}_{\overline{b}}^{k\text{-MDDH}}(1^\lambda)$, *where*

$$\begin{aligned}\mathcal{G}_{\overline{b}}^{k\text{-MDDH}}(1^\lambda): &\ \textit{It runs } \mathcal{G}(1^\lambda, k) \textit{ to generate } \vec{\mathbb{G}} = (\mathbb{G}_1, \ldots, \mathbb{G}_k) \textit{ with } g_1, \ldots, g_k \textit{ of order } p;\\ &\ \textit{picks random } \overline{s}, c_1, \ldots, c_k \in \mathbb{Z}_p \textit{ and computes } \overline{S} = g_1^{\overline{s}}, C_1 = g_1^{c_1}, \ldots, C_k = g_1^{c_k};\\ &\ \textit{sets } T_0 = g_k^{\overline{s}\prod_{j=1}^k c_j},\ T_1 = \textit{some random element in } \mathbb{G}_k;\\ &\ \textit{returns } \varrho_{\overline{b}} = (\vec{\mathbb{G}}, g_1, \overline{S}, C_1, \ldots, C_k, T_{\overline{b}}).\end{aligned}$$

The advantage of a probabilistic algorithm $\mathcal{B}$ *in solving the* k-MDDH *problem is defined as*

$$\mathsf{Adv}_{\mathcal{B}}^{k\text{-MDDH}}(\lambda) = |\Pr[\mathcal{B}(1^\lambda, \varrho_0) \to 1] - \Pr[\mathcal{B}(1^\lambda, \varrho_1) \to 1]|.$$

The k-MDDH *assumption is that for all* PPT *algorithms* $\mathcal{B}$, $\mathsf{Adv}_{\mathcal{B}}^{k\text{-MDDH}}(\lambda)$ *is at most negligible.*

Assumption 2 (*k*-Multilinear Computational Diffie-Hellman: *k*-MCDH [HSW13]). *The* k*-multilinear computational Diffie-Hellman* (k-MCDH) *problem is to output* T *given* $\varrho = (\vec{\mathbb{G}}, g_1, C_1, \ldots, C_k)$ *generated by* $\mathcal{G}^{k\text{-MCDH}}(1^\lambda)$, *where*

$$\begin{aligned}\mathcal{G}^{k\text{-MCDH}}(1^\lambda): &\ \textit{It runs } \mathcal{G}(1^\lambda, k) \textit{ to generate } \vec{\mathbb{G}} = (\mathbb{G}_1, \ldots, \mathbb{G}_k) \textit{ with } g_1, \ldots, g_k \textit{ of order } p;\\ &\ \textit{picks random } c_1, \ldots, c_k \in \mathbb{Z}_p \textit{ and computes } C_1 = g_1^{c_1}, \ldots, C_k = g_1^{c_k};\\ &\ \textit{returns } \varrho = (\vec{\mathbb{G}}, g_1, C_1, \ldots, C_k);\end{aligned}$$

and $T = g_{k-1}^{\prod_{i=1}^k c_i}$. *The advantage of a probabilistic algorithm* $\mathcal{B}$ *in solving the* k-MCDH *problem is defined as*

$$\mathsf{Adv}_{\mathcal{B}}^{k\text{-MCDH}}(\lambda) = \Pr[\mathcal{B}(1^\lambda, \varrho) \to T].$$

The k-MCDH *assumption is that for all* PPT *algorithms* $\mathcal{B}$, $\mathsf{Adv}_{\mathcal{B}}^{k\text{-MCDH}}(\lambda)$ *is at most negligible.*

3 Our ABE Scheme

The Construction

$\mathsf{ABE.Setup}(1^\lambda, n, l)$: The trusted key generation center takes as input a security parameter 1^λ, the length of Boolean inputs n to the decryption policy circuits, and the allowed depth l of decryption policy circuits. It proceeds as follows:

1. It runs $\mathcal{G}(1^\lambda, k = n + l + 1)$ to produce group sequence $\vec{\mathbb{G}} = (\mathbb{G}_1, \ldots, \mathbb{G}_k)$ of prime order $p > 2^\lambda$ with canonical generators $g_1, \ldots, g_k$.
2. It selects random $\alpha \in \mathbb{Z}_p$ together with random $(a_{1,0}, a_{1,1}), \ldots, (a_{n,0}, a_{n,1}) \in \mathbb{Z}_p^2$, and computes
$$H = g_{l+1}^{\alpha},\ A_{i,\beta} = g_1^{a_{i,\beta}} \text{ for } i = 1, \ldots, n;\ \beta \in \{0, 1\}.$$
3. It publishes the public parameters PP consisting of the group sequence description along with H and $\{A_{i,\beta}\}_{i=1,\ldots,n;\ \beta\in\{0,1\}}$. The master secret key $\mathsf{MK} = g_l^{\alpha}$ is kept to itself.

$\mathsf{ABE.KeyGen}(\mathsf{PP}, \mathsf{MK}, f)$: The key generation center takes as input the public parameters PP, the master secret key MK, and the description $f = (n, q, l, \mathbb{A}, \mathbb{B}, \mathsf{GateType})$ of a decryption policy circuit from a decrypter. Our circuit has $n + q$ wires $\{1, \ldots, n + q\}$ where $\{1, \ldots, n\}$ are n input wires, $\{n + 1, \ldots, n + q\}$ are q gates (OR or AND gates), and the wire $n + q$ designated as the output wire. It performs the following steps:

1. It chooses random $r_1, \ldots, r_{n+q} \in \mathbb{Z}_p$ where we think of randomness r_w as being associated with wire $w \in \{1, \ldots, n + q\}$. It produces the "header" component
$$\mathcal{K} = g_l^{\alpha} g_l^{-r_{n+q}} = g_l^{\alpha - r_{n+q}},$$
where g_l^{α} is obtained from MK.

2. Next, it generates key components for every wire w. The structure of the key component depends upon the category of w, i.e., whether w is an Input wire, an OR gate, or an AND gate. We describe how it generates the key components in each case.

   - Input wire: If $w \in \{1, \ldots, n\}$ then it corresponds to the w-th input. It computes the key component
   $$\mathcal{K}_w = e(A_{w,1}, g_1)^{r_w} = g_2^{r_w a_{w,1}}.$$
   - OR gate: Suppose that wire $w \in \mathsf{Gates}$, $\mathsf{GateType}(w) = \mathsf{OR}$, and $j = \mathsf{depth}(w)$. It picks random $b_w, d_w \in \mathbb{Z}_p$ and creates the key component
   $$\mathcal{K}_w = \Big(K_{w,1} = g_1^{b_w}, K_{w,2} = g_1^{d_w}, K_{w,3} = g_j^{r_w - b_w r_{\mathbb{A}(w)}}, K_{w,4} = g_j^{r_w - d_w r_{\mathbb{B}(w)}}\Big).$$
   - AND gate: Let wire $w \in \mathsf{Gates}$, $\mathsf{GateType}(w) = \mathsf{AND}$, and $j = \mathsf{depth}(w)$. It selects random $b_w, d_w \in \mathbb{Z}_p$ and forms the key component
   $$\mathcal{K}_w = \Big(K_{w,1} = g_1^{b_w}, K_{w,2} = g_1^{d_w}, K_{w,3} = g_j^{r_w - b_w r_{\mathbb{A}(w)} - d_w r_{\mathbb{B}(w)}}\Big).$$

3. It provides the decryption key $\mathsf{SK}_f^{(\mathsf{DEC})} = (f, \mathcal{K}, \{\mathcal{K}_w\}_{w \in \{1,\ldots,n+q\}})$ to the decrypter.

$\mathsf{ABE.Encrypt}(\mathsf{PP}, x, M)$: Taking as input the public parameters PP, an encryption input string $x = x_1 \ldots x_n \in \{0,1\}^n$, and a message $M \in \mathbb{G}_k$, the encrypter forms the ciphertext as follows:

1. It picks random $s \in \mathbb{Z}_p$ and computes
$$C_M = e(h, A_{1,x_1}, \ldots, A_{n,x_n})^s M = g_{n+l+1}^{\alpha s \prod_{i=1}^{n} a_{i,x_i}} M = g_k^{\alpha s \delta(x)} M,$$
where we define $\delta(x) = \prod_{i=1}^{n} a_{i,x_i}$ for the ease of exposition. It also computes $C = g_1^s$.
2. It outputs the ciphertext $\mathsf{CT}_x = (x, C_M, C)$.

$\mathsf{ABE.Decrypt}(\mathsf{PP}, \mathsf{CT}_x, \mathsf{SK}_f^{(\mathsf{DEC})})$: A decrypter, on input the public parameters PP, a ciphertext $\mathsf{CT}_x = (x, C_M, C)$ encrypted for encryption input string $x = x_1 \ldots x_n \in \{0,1\}^n$, along with its decryption key $\mathsf{SK}_f^{(\mathsf{DEC})} = (f, \mathcal{K}, \{\mathcal{K}_w\}_{w \in \{1,\ldots,n+q\}})$ corresponding to its decryption policy circuit $f = (n, q, l, \mathbb{A}, \mathbb{B}, \mathsf{GateType})$, outputs $\perp$, if $f(x) = 0$; otherwise, (i.e., if $f(x) = 1$) proceeds as follows:

1. First, there is a header computation, where it computes
$$D = e(A_{1,x_1}, \ldots, A_{n,x_n}) = g_n^{\delta(x)}$$
$$\text{followed by } \widehat{E} = e(\mathcal{K}, D, C) = e(g_l^{\alpha - r_{n+q}}, g_n^{\delta(x)}, g_1^s) = g_k^{(\alpha - r_{n+q}) s \delta(x)}$$
by extracting $\{A_{i,x_i}\}_{i=1,\ldots,n}$ from PP.
2. Next, it evaluates the circuit from the bottom up. For every wire w with corresponding $\mathsf{depth}(w) = j$, if $f_w(x) = 0$, nothing needs to be computed for that wire, otherwise (if $f_w(x) = 1$), it attempts to compute $E_w = g_{n+j+1}^{r_w s \delta(x)}$ as described below. The decrypter proceeds iteratively starting with computing E_1 and moves forward in order to finally compute E_{n+q}. Note that computing these values in order ensures that the computation on a wire w with $\mathsf{depth}(w) = j - 1$ that evaluates to 1 will be defined before computing for a wire w with $\mathsf{depth}(w) = j$. The computation procedure varies with the category of the wire, i.e., whether the wire is an Input wire, an OR gate, or an AND gate.

   - Input wire: If $w \in \{1, \ldots, n\}$ then it corresponds to the w-th input. Suppose that $x_w = f_w(x) = 1$. The decrypter extracts $\{A_{i,x_i}\}_{i,\ldots,n}$ from PP and computes
   $$\begin{aligned} E_w &= e(\mathcal{K}_w, A_{1,x_1}, \ldots, A_{w-1,x_{w-1}}, A_{w+1,x_{w+1}}, \ldots, A_{n,x_n}, C) \\ &= e(g_2^{r_w a_{w,x_w}}, g_1^{a_{1,x_1}}, \ldots, g_1^{a_{w-1,x_{w-1}}}, g_1^{a_{w+1,x_{w+1}}}, \ldots, g_1^{a_{n,x_n}}, g_1^s) = g_{n+2}^{r_w s \delta(x)}. \end{aligned}$$

- OR gate: Consider a wire $w \in \mathsf{Gates}$ with $\mathsf{GateType}(w) = \mathsf{OR}$ and $j = \mathsf{depth}(w)$. Assume that $f_w(x) = 1$. Then either $f_{\mathbb{A}(w)}(x) = 1$ or $f_{\mathbb{B}(w)}(x) = 1$. If $f_{\mathbb{A}(w)}(x) = 1$, i.e., the first input of gate w evaluates to 1, then the decrypter computes

$$\begin{aligned} E_w = \; & e(E_{\mathbb{A}(w)}, K_{w,1}) e(K_{w,3}, D, C) \\ = \; & e(g_{n+j}^{r_{\mathbb{A}(w)} s\delta(x)}, g_1^{b_w}) e(g_j^{r_w - b_w r_{\mathbb{A}(w)}}, g_n^{\delta(x)}, g_1^{s}) = g_{n+j+1}^{r_w s\delta(x)}. \end{aligned}$$

  Note that $E_{\mathbb{A}(w)}$ is already computed at this stage in the bottom-up circuit evaluation as $\mathsf{depth}(\mathbb{A}(w)) = j - 1$.
  Alternatively, if $f_{\mathbb{A}(w)}(x) = 0$ but $f_{\mathbb{B}(w)}(x) = 1$, then it computes

$$E_w = e(E_{\mathbb{B}(w)}, K_{w,2}) e(K_{w,4}, D, C) = g_{n+j+1}^{r_w s\delta(x)}.$$

- AND gate: Consider a wire $w \in \mathsf{Gates}$ with $\mathsf{GateType}(w) = \mathsf{AND}$ and $j = \mathsf{depth}(w)$. Suppose that $f_w(x) = 1$. Then $f_{\mathbb{A}(w)}(x) = f_{\mathbb{B}(w)}(x) = 1$. The decrypter computes

$$\begin{aligned} E_w = \; & e(E_{\mathbb{A}(w)}, K_{w,1}) e(E_{\mathbb{B}(w)}, K_{w,2}) e(K_{w,3}, D, C) \\ = \; & e(g_{n+j}^{r_{\mathbb{A}(w)} s\delta(x)}, g_1^{b_w}) e(g_{n+j}^{r_{\mathbb{B}(w)} s\delta(x)}, g_1^{d_w}) e(g_j^{r_w - b_w r_{\mathbb{A}(w)} - d_w r_{\mathbb{B}(w)}}, g_n^{\delta(x)}, g_1^{s}) = g_{n+j+1}^{r_w s\delta(x)}. \end{aligned}$$

  In this process, the decrypter ultimately computes $E_{n+q} = g_k^{r_{n+q} s\delta(x)}$, as $f(x) = f_{n+q}(x) = 1$.

3. Finally, the decrypter computes $E = \widehat{E} E_{n+q} = g_k^{\alpha s\delta(x)}$ and retrieves the message by the computation $C_M E^{-1} = g_k^{\alpha s\delta(x)} M (g_k^{\alpha s\delta(x)})^{-1} = M$.

Security

The security property of the above ABE construction is formally stated in the following theorem, the proof of which is presented in Appendix A.2.

Theorem 1 (Security of ABE). *The* ABE *scheme of Section 3 supporting arbitrary circuits of depth l and input length n, characterizing decryption rights achieves selective* CPA*-security as per the security model of Appendix A.1 under the k-*MDDH *assumption where $k = n + l + 1$.*

Efficiency

Table 1 compares the communication and storage requirements of our proposed ABE scheme with previously known multilinear map-based ABE constructions supporting general circuits in terms of the number of group elements comprising the public parameters PP, ciphertext CT_x, and decryption key $\mathsf{SK}_f^{(\mathsf{DEC})}$. As is clear from the table, the most significant achievement of our construction is that our ABE ciphertext involves only 2 (constant) group elements which is smaller than all earlier constructions. Also, our decryption key contains only a single group element corresponding to each input wire of the decryption policy circuits. In all existing constructions, number of group elements required for each input wire of the decryption policy circuits is strictly greater than one.

Looking from a different view point, observe that in current non-ideal multilinear map candidates [GGH13a], [CLT13], [CLT15], the bit length of an encoding is $\widetilde{O}(k\lambda^2)$ where k is the maximum allowed multilinearity level and λ is the underlying security parameter. Thus, our ABE ciphertext has bit size $\widetilde{O}((n + l)\lambda^2)$, in contrast to $\widetilde{O}(nl\lambda^2)$ for [GGH+13b], [Att14], or $\widetilde{O}((nq + q^2)\lambda^2)$ for [GGHZ14]. Consequently, we can see that, in terms of bit length as well, our ciphertext is shorter compared to [GGH+13b], [Att14], [GGHZ14]. The multilinear map based ABE construction of [BGG+14] has ciphertext bit length $\widetilde{O}(l\lambda^2)$, while its decryption key size is $\widetilde{O}((n+n^2+q)l\lambda^2)$ which is larger than the corresponding value $\widetilde{O}((n^2+(n+q)l)\lambda^2)$ for our scheme.

Table 1: Communication and storage comparison

ABE	Security	Complexity Assumptions	k	$\lvert\mathsf{PP}\rvert$	$\lvert\mathsf{CT}_x\rvert$	$\lvert\mathsf{SK}_f^{(\mathsf{DEC})}\rvert$
[GGH+13b]	selective	MDDH	$l+1$	$n+1$	$n+2$	$2n+4q+1$
[BGG+14]	selective	MDHE	$l+1$	$n+1$	3	$n+n^2+4q+1$
[GGHZ14]	adaptive	3 new non-standard assumptions	$n+2q+2$	$2n+4q+3$	$4q+3$	$4q+2$
[Att14]	adaptive	$\mathsf{SD}_1, \mathsf{SD}_2, \mathsf{EMDDH}_1, \mathsf{EMDDH}_2$	$3l$	$n+4$	$n+4$	$2n+4q+3$
Ours	selective	MDDH	$n+l+1$	$2n+1$	2	$n+4q+1$

Here, $\mathsf{MDDH}, \mathsf{MDHE}, \mathsf{SD}_1, \mathsf{SD}_2, \mathsf{EMDDH}_1, \mathsf{EMDDH}_2$ stand respectively for the Multilinear Decisional Diffie-Hellman [GGH+13b], Multilinear Diffie-Hellman Exponent [BGG+14], two variants of Multilinear Subgroup Decision [Att14], and the two versions of the Expanded Multilinear Decisional Diffie-Hellman assumptions [Att14]. In this table, k denotes the maximum multilinearity level of the underlying multilinear maps, n, q, and l represent respectively the input length, number of gates, and depth of the decryption policy circuits, while $\lvert\mathsf{PP}\rvert$, $\lvert\mathsf{CT}_x\rvert$, and $\lvert\mathsf{SK}_f^{(\mathsf{DEC})}\rvert$ stand respectively for the number of group elements comprising $\mathsf{PP}, \mathsf{CT}_x$, and $\mathsf{SK}_f^{(\mathsf{DEC})}$.

Table 2: Comparison of multilinear operation count

ABE	ABE.Setup	ABE.KeyGen	ABE.Encrypt	ABE.Decrypt
[GGH+13b]	$n+2$	$3n+4q+1$	$n+2$	$2n+3q+1$
[BGG+14]	$n+2$	$n^2+2n+4q+1$	3	$2n+3q+1$
Ours	$2n+2$	$2n+4q+1$	3	$n+3q+3$

In this table, n and q denote respectively the input size and number of gates in the decryption policy circuits.

Further, the selective security of the multilinear map based ABE construction of [BGG+14] is derived from the Multilinear Diffie-Hellman Exponent assumption which is proven to hold in the generic multilinear map model [BGG+14]. On the contrary, the validity of our complexity assumption, namely, the Multilinear Decisional Diffie-Hellman assumption in present multilinear map candidates has been established [GGH13a], [CLT13], [CLT15] without using such a folklore framework.

Regarding computational efficiency, notice that unlike traditional bilinear map setting, in current multilinear map candidates [GGH13a], [CLT13], [CLT15], exponentiation is also realized through multilinear operation. Since multilinear operations are costlier compared to group operations in multilinear groups, we consider the count of multilinear operations required in each algorithm of ABE scheme as a parameter for comparing computational cost. Table 2 demonstrates the number of multilinear operations involved in the setup, key generation, encryption and decryption algorithms of our ABE scheme in comparison to existing multilinear map-based selectively secure ABE constructions for arbitrary circuits. From the table it readily follows that the key generation, encryption, as well as decryption algorithms of our scheme requires the least number of multilinear operations among all the three schemes. We exclude the adaptively secure constructions [GGHZ14], [Att14] from the comparative analysis of computational complexity as the adaptive security of those constructions has been achieved at the expense of computational efficiency.

4 Our ABSC Scheme

The Construction

$\mathsf{ABSC.Setup}(1^\lambda, n, n', l)$: The trusted key generation center takes as input a security parameter 1^λ, the length n of inputs to decryption policy circuits, the length n' of inputs to signing policy circuits, and the common allowed depth l of both types of circuits. It proceeds as follows:

1. It runs $\mathcal{G}(1^\lambda, k = n + n' + l + 1)$ to produce group sequence $\vec{\mathbb{G}} = (\mathbb{G}_1, \ldots, \mathbb{G}_k)$ of prime order $p > 2^\lambda$ with canonical generators $g_1, \ldots, g_k$.
2. It selects random $\alpha_1, \alpha_2 \in \mathbb{Z}_p$ along with random $(a_{1,0}, a_{1,1}), \ldots, (a_{n,0}, a_{n,1}), (b_{1,0}, b_{1,1}), \ldots, (b_{n',0}, b_{n',1}) \in \mathbb{Z}_p^2$, sets $\alpha = \alpha_1 + \alpha_2$, and computes $H = g_{l+1}^{\alpha_1}$, $A_{i,\beta} = g_1^{a_{i,\beta}}$, $B_{t,\beta} = g_1^{b_{t,\beta}}$ for $i = 1, \ldots, n;\ t = 1, \ldots, n';\ \beta \in \{0, 1\}$.
3. Additionally, it chooses random $\theta \in \mathbb{Z}_p$ and computes $\Theta = g_n^\theta, Y = g_{n+l+1}^{\theta\alpha_2}$.
4. It publishes the public parameters PP consisting of the group sequence description together with $\{A_{i,\beta}\}_{i=1,\ldots,n;\ \beta\in\{0,1\}}, \{B_{t,\beta}\}_{t=1,\ldots,n';\ \beta\in\{0,1\}}, H, \Theta, Y$, while keeps the master secret key $\mathsf{MK} = (g_l^\alpha, g_l^{\alpha_2})$.

$\mathsf{ABSC.SKeyGen}(\mathsf{PP}, \mathsf{MK}, g)$: On input the public parameters PP, the master secret key MK, and the description $g = (n', q', l, \mathbb{A}, \mathbb{B}, \mathsf{GateType})$ of a signing policy circuit from a signcrypter, the key generation center forms a signing key as described below. Recall that the circuit g has $n' + q'$ wires $\{1, \ldots, n' + q'\}$ with n' input wires $\{1, \ldots, n'\}$, q' gates $\{n' + 1, \ldots, n' + q'\}$, and the wire $n' + q'$ designated as the output wire.

1. It chooses random $r'_1, \ldots, r'_{n'+q'-1} \in \mathbb{Z}_p$ and sets $r'_{n'+q'} = \alpha_2$, where again we will think of the random value r'_w as being associated with the wire w.
2. It proceeds to generate the key components for every wire w. Here also the structure of the key component depends upon the category of the wire $w \in \{1, \ldots, n' + q'\}$, i.e., whether w is an Input wire, an OR gate, or an AND gate. We describe how it generates the key component in each case.

   - Input wire: If $w \in \{1, \ldots, n'\}$ then it corresponds to the w-th input. It computes the key component
   $$\mathcal{K}'_w = e(B_{w,1}, g_1)^{r'_w} = g_2^{r'_w b_{w,1}}.$$
   - OR gate: Suppose that wire $w \in \mathsf{Gates}$, $\mathsf{GateType}(w) = \mathsf{OR}$, and $j = \mathsf{depth}(w)$. It picks random $b'_w, d'_w \in \mathbb{Z}_p$ and creates the key component
   $$\mathcal{K}'_w = \Big(K'_{w,1} = g_1^{b'_w}, K'_{w,2} = g_1^{d'_w}, K'_{w,3} = g_j^{r'_w - b'_w r'_{\mathbb{A}(w)}}, K'_{w,4} = g_j^{r'_w - d'_w r'_{\mathbb{B}(w)}}\Big).$$
   - AND gate: Let wire $w \in \mathsf{Gates}$, $\mathsf{GateType}(w) = \mathsf{AND}$, and $j = \mathsf{depth}(w)$. It selects random $b'_w, d'_w \in \mathbb{Z}_p$ and generates the key component
   $$\mathcal{K}'_w = \Big(K'_{w,1} = g_1^{b'_w}, K'_{w,2} = g_1^{d'_w}, K'_{w,3} = g_j^{r'_w - b'_w r'_{\mathbb{A}(w)} - d'_w r'_{\mathbb{B}(w)}}\Big).$$

   Notice that while computing the key component $\mathcal{K}'_{n'+q'}$ for the output gate $n' + q'$ which has depth l, the required $g_l^{r'_{n'+q'}} = g_l^{\alpha_2}$ is retrieved from MK.
3. It gives the signing key $\mathsf{SK}_g^{(\mathsf{SIG})} = (g, \{\mathcal{K}'_w\}_{w\in\{1,\ldots,n'+q'\}})$ to the signcrypter.

$\mathsf{ABSC.DKeyGen}(\mathsf{PP}, \mathsf{MK}, f)$: Taking as input the public parameters PP, the master secret key MK, and the description $f = (n, q, l, \mathbb{A}, \mathbb{B}, \mathsf{GateType})$ of a decryption policy circuit from a decrypter, the key generation center creates a decryption key $\mathsf{SK}_f^{(\mathsf{DEC})} = (f, \mathcal{K}, \{\mathcal{K}_w\}_{w\in\{1,\ldots,n+q\}})$

in the same manner as the $\mathsf{ABE.KeyGen}(\mathsf{PP}, \mathsf{MK}, f)$ algorithm described in Section 3 using $\{A_{i,\beta}\}_{i=1,\ldots,n;\ \beta\in\{0,1\}}$ obtained from PP and g_l^{α} extracted from MK. We omit the details here. It hands the decryption key $\mathsf{SK}_f^{(\mathsf{DEC})}$ to the decrypter.

$\mathsf{ABSC.Signcrypt}(\mathsf{PP}, \mathsf{SK}_g^{(\mathsf{SIG})}, x, y, M)$: A signcrypter takes as input the public parameters PP, its signing key $\mathsf{SK}_g^{(\mathsf{SIG})} = (g, \{\mathcal{K}'_w\}_{w\in\{1,\ldots,n'+q'\}})$ corresponding to some signing policy circuit $g = (n', q', l, \mathbb{A}, \mathbb{B}, \mathsf{GateType})$, an encryption input string $x = x_1 \ldots x_n \in \{0,1\}^n$, a signature input string $y = y_1 \ldots y_{n'} \in \{0,1\}^{n'}$ satisfying $g(y) = 1$, and a message $M \in \mathbb{G}_k$. It prepares the ciphertext as follows:

1. It first evaluates the signing policy circuit from the bottom up. As before, we define $\delta(x) = \prod_{i=1}^{n} a_{i,x_i}$ and $\delta'(y) = \prod_{t=1}^{n'} b_{t,y_t}$ for improving readability. It starts by computing

$$D' = e(B_{1,y_1}, \ldots, B_{n',y_{n'}}) = g_{n'}^{\delta'(y)},$$

where $\{B_{t,y_t}\}_{t=1,\ldots,n'}$ are extracted from PP. For every wire w in g with corresponding $\mathsf{depth}(w) = j$, if $g_w(y) = 0$ then nothing needs to be computed for that wire; on the other hand, if $g_w(y) = 1$ then it computes $E'_w = g_{n'+j}^{r'_w\delta'(y)}$ as described below. The signcrypter proceeds iteratively starting with computing E'_1 and moves forward in order to ultimately compute $E'_{n'+q'} = g_{n'+l}^{r'_{n'+q'}\delta'(y)} = g_{n'+l}^{\alpha_2\delta'(y)}$. Note that $r'_{n'+q'}$ has been set to α_2 by the key generation center. Moreover, observe that computing the E'_w values in order ensures that the computation on a wire w with $\mathsf{depth}(w) = j - 1$ that evaluates to 1 will be defined before computing for a wire w with $\mathsf{depth}(w) = j$. The computation procedure varies with the category of the wire, i.e., Input wire, OR gate, or AND gate in this case as well.

- $\underline{\mathsf{Input}\text{ wire}}$: If $w \in \{1, \ldots, n'\}$ then it corresponds to the w-th input. Suppose that $y_w = 1$. The signcrypter computes

$$E'_w = e(\mathcal{K}'_w, B_{1,y_1}, \ldots, B_{w-1,y_{w-1}}, B_{w+1,y_{w+1}}, \ldots, B_{n',y_{n'}}) = g_{n'+1}^{r'_w\delta'(y)}.$$

- $\underline{\mathsf{OR}\text{ gate}}$: Consider a wire $w \in \mathsf{Gates}$ with $\mathsf{GateType}(w) = \mathsf{OR}$ and $j = \mathsf{depth}(w)$. Assume that $g_w(y) = 1$. Then either $g_{\mathbb{A}(w)}(y) = 1$ or $g_{\mathbb{B}(w)}(y) = 1$. If $g_{\mathbb{A}(w)}(y) = 1$ then the signcrypter computes

$$E'_w = e(E'_{\mathbb{A}(w)}, K'_{w,1})e(K'_{w,3}, D') = g_{n'+j}^{r'_w\delta'(y)}.$$

Alternatively, if $g_{\mathbb{A}(w)}(y) = 0$ but $g_{\mathbb{B}(w)}(y) = 1$ then it computes

$$E'_w = e(E'_{\mathbb{B}(w)}, K'_{w,2})e(K'_{w,4}, D') = g_{n'+j}^{r'_w\delta'(y)}.$$

- $\underline{\mathsf{AND}\text{ gate}}$: Consider a wire $w \in \mathsf{Gates}$ with $\mathsf{GateType}(w) = \mathsf{AND}$ and $j = \mathsf{depth}(w)$. Suppose that $g_w(y) = 1$. Hence $g_{\mathbb{A}(w)}(y) = g_{\mathbb{B}(w)}(y) = 1$. The signcrypter computes

$$E'_w = e(E'_{\mathbb{A}(w)}, K'_{w,1})e(E'_{\mathbb{B}(w)}, K'_{w,2})e(K'_{w,3}, D') = g_{n'+j}^{r'_w\delta'(y)}.$$

2. Next the signcrypter picks random $s \in \mathbb{Z}_p$ and computes

$$C_M = \Big(e(H, A_{1,x_1}, \ldots, A_{n,x_n}, D')e(E'_{n'+q'}, A_{1,x_1}, \ldots, A_{n,x_n}, g_1)\Big)^s M = g_k^{\alpha s\delta(x)\delta'(y)}M,$$
$$C = g_1^s,\ C' = e(\Theta, E'_{n'+q'}) = g_{k-1}^{\theta\alpha_2\delta'(y)}.$$

Here, H, Θ, and $\{A_{i,x_i}\}_{i=1,\ldots,n}$ are extracted from PP.

3. The signcrypter outputs the ciphertext $\mathsf{CT}_{x,y} = (x, y, C_M, C, C')$.

$\mathsf{ABSC.Unsigncrypt}(\mathsf{PP}, \mathsf{CT}_{x,y}, \mathsf{SK}_f^{(\mathsf{DEC})})$: A decrypter takes as input the public parameters PP, a ciphertext $\mathsf{CT}_{x,y} = (x, y, C_M, C, C')$ signcrypted with an encryption input string $x = x_1 \dots x_n \in \{0,1\}^n$ and a signature input string $y = y_1 \dots y_{n'} \in \{0,1\}^{n'}$, as well as its decryption key $\mathsf{SK}_f^{(\mathsf{DEC})} = (f, \mathcal{K}, \{\mathcal{K}_w\}_{w \in \{1,\dots,n+q\}})$ corresponding to its legitimate decryption circuit $f = (n, q, l, \mathbb{A}, \mathbb{B}, \mathsf{GateType})$. It performs the following steps:

1. It first computes $D' = e(B_{1,y_1}, \dots, B_{n',y_{n'}}) = g_{n'}^{\delta'(y)}$ and checks the validity of the ciphertext as
$$e(C', g_1) = e(Y, D').$$
Note that if the ciphertext is valid then both sides of the above equality should evaluate to $g_k^{\theta\alpha_2\delta'(y)}$. If the above equation is invalid then it outputs $\perp$; otherwise, it proceeds to the next step.
2. If $f(x) = 0$ then it outputs $\perp$; on the other hand, if $f(x) = 1$ then it proceeds in the same way as in the case of $\mathsf{ABE.Decrypt}(\mathsf{PP}, \mathsf{CT}_x, \mathsf{SK}_f^{(\mathsf{DEC})})$ algorithm of Section 3 to compute the header $\widehat{E} = g_{n+l+1}^{(\alpha - r_{n+q})s\delta(x)}$ followed by a computation of the circuit from the bottom up ultimately obtaining $E_{n+q} = g_{n+l+1}^{r_{n+q}s\delta(x)}$. In this computation it makes use of C obtained from $\mathsf{CT}_{x,y}$ and $\{A_{i,x_i}\}_{i=1,\dots,n}$ extracted from PP along with its decryption key components. We omit the details here.
3. Finally, the decrypter retrieves the message by computing
$$C_M \big[e(\widehat{E}E_{n+q}, D')\big]^{-1} = g_k^{\alpha s\delta(x)\delta'(y)} M \big[e(g_{n+l+1}^{(\alpha - r_{n+q})s\delta(x)} g_{n+l+1}^{r_{n+q}s\delta(x)}, g_{n'}^{\delta'(y)})\big]^{-1} = M.$$

Security

The security of the proposed ABSC construction is captured by the following two theorems, the proofs of which are given in Appendix B.2.

Theorem 2 (Message Confidentiality of ABSC). *The* ABSC *scheme supporting arbitrary decryption policy circuits of input length n and depth l, as well as, arbitrary signing policy circuits of input length n' and the same depth l, described in Section 4, achieves selective message confidentiality against* CPA *as per the model of Appendix B.1 under the k-*MDDH *assumption, where $k = n + n' + l + 1$.*

Theorem 3 (Ciphertext Unforgeability of ABSC). *The* ABSC *scheme supporting arbitrary decryption policy circuits of input length n and depth l, as well as, arbitrary signing policy circuits of input length n' and depth l, described in Section 4, achieves selective ciphertext unforgeability against* CMA *as per the model of Appendix B.1 under the k-*MCDH *assumption, where $k = n + n' + l + 1$.*

Efficiency

Regarding communication and storage complexity of the proposed ABSC construction, the number of multilinear group elements comprising the public parameters PP, ciphertext $\mathsf{CT}_{x,y}$, decryption key $\mathsf{SK}_f^{(\mathsf{DEC})}$, and signing key $\mathsf{SK}_g^{(\mathsf{SIG})}$ are respectively $2n + 2n' + 3$, 3, $n + 4q + 1$, and $n' + 4q'$ where we have used a multilinear map with multilinearity level $k = n + n' + l + 1$, n, q being respectively the input length and number of gates of the decryption policy circuits, n', q' being the corresponding values for the signing policy circuits, and l being the allowed depth of both kinds of circuits. On the other hand, about computational cost, notice that the count of multilinear operations involved in the setup, signing key generation, decryption key generation, encryption, and decryption algorithms of our ABSC scheme are respectively $2n{+}2n'{+}3$, $2n'{+}4q'$, $2n + 4q + 1$, $n' + 3q' + 6$, and $n + 3q + 5$. We emphasize that our ABSC construction supports arbitrary polynomial-size circuits of unbounded fan-out, whereas, all the earlier constructions could support at most circuits of fan-out one. Moreover, our scheme utilizes multilinear map whose implementation is completely different from that of the bilinear map employed in all existing ABSC schemes.

5 Conclusion

In this work, we designed an ABE scheme followed by an ABSC scheme both supporting general circuit realizable access policies. Our constructions were proven selectively secure under Multilinear Decisional Diffie-Hellman and Multilinear Computational Diffie-Hellman assumptions. The ciphertext sizes of both our constructions are very short. Most importantly, our ABSC scheme is the first to support signing and decryption policies representable as arbitrary polynomial-size circuits which are highly expressive. It is worth noting that although our selectively secure constructions can be made adaptively secure using the technique of complexity leveraging, the reduction would lose an exponential factor. Constructing adaptively secure ABE and ABSC featuring a constant number of ciphertext components with only a polynomial loss in the security reduction would be of immense theoretical and practical significance.

References

[Att14] Nuttapong Attrapadung. Fully secure and succinct attribute based encryption for circuits from multi-linear maps. Technical report, IACR Cryptology ePrint Archive, 2014: 772, 2014.

[BB11] Dan Boneh and Xavier Boyen. Efficient selective identity-based encryption without random oracles. *Journal of Cryptology*, 24(4):659–693, 2011.

[BGG$^+$14] Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit abe and compact garbled circuits. In *Advances in Cryptology–EUROCRYPT 2014*, pages 533–556. Springer, 2014.

[BS03] Dan Boneh and Alice Silverberg. Applications of multilinear forms to cryptography. *Contemporary Mathematics*, 324(1):71–90, 2003.

[CHL$^+$14] Jung Hee Cheon, Kyoohyung Han, Changmin Lee, Hansol Ryu, and Damien Stehlé. Cryptanalysis of the multilinear map over the integers. Technical report, IACR Cryptology ePrint Archive, 2014: 906, 2014.

[CLT13] Jean-Sébastien Coron, Tancrede Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. In *Advances in Cryptology–CRYPTO 2013*, pages 476–493. Springer, 2013.

[CLT15] Jean-Sebastien Coron, Tancrede Lepoint, and Mehdi Tibouchi. New multilinear maps over the integers. Technical report, IACR Cryptology ePrint Archive, 2015: 162, 2015.

[EMR12] Keita Emura, Atsuko Miyaji, and Mohammad Shahriar Rahman. Dynamic attribute-based signcryption without random oracles. *International Journal of Applied Cryptography*, 2(3):199–211, 2012.

[GGH13a] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In *Advances in Cryptology–EUROCRYPT 2013*, pages 1–17. Springer, 2013.

[GGH$^+$13b] Sanjam Garg, Craig Gentry, Shai Halevi, Amit Sahai, and Brent Waters. Attribute-based encryption for circuits from multilinear maps. In *Advances in Cryptology–CRYPTO 2013*, pages 479–499. Springer, 2013.

[GGHZ14] Sanjam Garg, Craig Gentry, Shai Halevi, and Mark Zhandry. Fully secure attribute based encryption from multilinear maps. Technical report, IACR Cryptology ePrint Archive, 2014: 622, 2014.

[GKP$^+$13a] Shafi Goldwasser, Yael Kalai, Raluca Ada Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. Reusable garbled circuits and succinct functional encryption. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 555–564. ACM, 2013.

[GKP$^+$13b] Shafi Goldwasser, Yael Tauman Kalai, Raluca Ada Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. How to run turing machines on encrypted data. In *Advances in Cryptology–CRYPTO 2013*, pages 536–553. Springer, 2013.

[GNSN10] Martin Gagné, Shivaramakrishnan Narayan, and Reihaneh Safavi-Naini. Threshold attribute-based signcryption. In *Security and Cryptography for Networks*, 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 *Proceedings of the 13th ACM conference on Computer and communications security*, pages 89–98. ACM, 2006.

[GVW13] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute-based encryption for circuits. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 545–554. ACM, 2013.

[HSW13] Susan Hohenberger, Amit Sahai, and Brent Waters. Full domain hash from (leveled) multilinear maps and identity-based aggregate signatures. In *Advances in Cryptology–CRYPTO 2013*, pages 494–512. Springer, 2013.

[PRV12] Bryan Parno, Mariana Raykova, and Vinod Vaikuntanathan. How to delegate and verify in public: Verifiable computation from attribute-based encryption. In *Theory of Cryptography*, pages 422–439. Springer, 2012.

[RD14a] Y Sreenivasa Rao and Ratna Dutta. Expressive attribute based signcryption with constant-size ciphertext. In *Progress in Cryptology–AFRICACRYPT 2014*, pages 398–419. Springer, 2014.

[RD14b] Y Sreenivasa Rao and Ratna Dutta. Expressive bandwidth-efficient attribute based signature and signcryption in standard model. In *Information Security and Privacy*, pages 209–225. Springer, 2014.

[SW05] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *Advances in Cryptology–EUROCRYPT 2005*, pages 457–473. Springer, 2005.

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

[WH11] Changji Wang and Jiasen Huang. Attribute-based signcryption with ciphertext-policy and claim-predicate mechanism. In *Computational Intelligence and Security–CIS 2011, Seventh International Conference on*, pages 905–909. IEEE, 2011.

[WHL14] Jianghong Wei, Xuexian Hu, and Wenfen Liu. Traceable attribute-based signcryption. *Security and Communication Networks*, 7(12):2302–2317, 2014.

A Security Analysis of the ABE Scheme of Section 3

A.1 Security Definition for ABE

The *selective* security notion of ABE for circuits against chosen plaintext attack (CPA) is defined in terms of the following game between a probabilistic adversary $\mathcal{A}$ and a probabilistic challenger $\mathcal{B}$:

Init: $\mathcal{A}$ commits to a challenge encryption input string $x^* \in \{0,1\}^n$ that would be used by $\mathcal{B}$ to create the challenge ciphertext.

Setup: $\mathcal{B}$ performs $\mathsf{ABE.Setup}(1^\lambda, n, l)$ to obtain PP, MK, and hands PP to $\mathcal{A}$.

Query Phase 1: $\mathcal{A}$ may adaptively make any polynomial number of decryption key queries for circuit description $f \in \mathbb{F}_{n,l}$ of its choice subject to the restriction that $f(x^*) = 0$. $\mathcal{B}$ returns the corresponding decryption keys $\mathsf{SK}_f^{(\mathsf{DEC})}$ to $\mathcal{A}$ by executing $\mathsf{ABE.KeyGen}(\mathsf{PP}, \mathsf{MK}, f)$.

Challenge: $\mathcal{A}$ submits two equal length messages $M_0^*, M_1^* \in \mathbb{M}$. Then $\mathcal{B}$ flips a random coin $b \in \{0,1\}$, and computes the challenge ciphertext CT^* by running $\mathsf{ABE.Encrypt}(\mathsf{PP}, x, M_b)$. The challenge ciphertext CT^* is given to $\mathcal{A}$.

Query Phase 2: $\mathcal{A}$ may continue adaptively to make decryption key queries as in **Query Phase 1** with the same restriction as above.

Guess: $\mathcal{A}$ eventually outputs a guess b' for b and wins the game if $b' = b$.

The advantage of the adversary $\mathcal{A}$ in the above game is defined as

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{ABE,s\text{-}IND\text{-}CPA}}(\lambda) = |\mathsf{Pr}[b' = b] - 1/2|.$$

Definition 1. *An* ABE *scheme for circuits is defined to be selectively secure against* CPA *if for all probabilistic polynomial-time* (PPT) *adversaries* $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{ABE,s\text{-}IND\text{-}CPA}}(\lambda)$ *is at most negligible.*

A.2 Proof of Theorem 1

Theorem 1 (Security of ABE). *The* ABE *scheme of Section 3 supporting arbitrary circuits of depth l and input length n, characterizing decryption rights achieves selective* CPA*-security as per the security model of Appendix A.1 under the k-*MDDH *assumption where $k = n + l + 1$. More precisely, for any* PPT *adversary $\mathcal{A}$ against the* ABE *scheme of Section 3, there exists a probabilistic algorithm $\mathcal{B}$, whose running time is essentially the same as that of $\mathcal{A}$, such that for any security parameter λ, $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{ABE,s\text{-}IND\text{-}CPA}}(\lambda) = \mathsf{Adv}_{\mathcal{B}}^{k\text{-}\mathsf{MDDH}}(\lambda)$.*

Proof. Suppose there exists a PPT adversary $\mathcal{A}$ that breaks with non-negligible advantage the selective CPA security of the ABE scheme supporting arbitrary circuits of depth l and input length n, described in Section 3. We construct a PPT algorithm $\mathcal{B}$ that attempts to solve an instance of the k-MDDH problem, where $k = n+l+1$, using $\mathcal{A}$ as a sub-routine. $\mathcal{B}$ is given an instance of the k-MDDH problem $\varrho_{\overline{b}} = (\overrightarrow{\mathbb{G}}, g_1, \overline{S}, C_1, \ldots, C_k, T_{\overline{b}})$ such that $\overline{S} = g_1^{\overline{s}}, C_1 = g_1^{c_1}, \ldots, C_k = g_1^{c_k}$. $\mathcal{B}$ plays the role of the challenger in the selective CPA-security game of Appendix A.1 and interacts with $\mathcal{A}$ as follows:

Init: $\mathcal{A}$ declares the challenge encryption input string $x^* = x_1^* \ldots x_n^* \in \{0,1\}^n$ to $\mathcal{B}$.

Setup: $\mathcal{B}$ chooses random $z_1, \ldots, z_n \in \mathbb{Z}_p$ and sets $a_{i,\beta} = c_i$ *implicitly*, if $\beta = x_i^*$, while $a_{i,\beta} = z_i$, if $\beta \neq x_i^*$, for $i = 1, \ldots, n$; $\beta \in \{0,1\}$. This corresponds to setting $A_{i,\beta} = C_i = g_1^{c_i}$, if $\beta = x_i^*$, while $A_{i,\beta} = g_1^{z_i}$, if $\beta \neq x_i^*$, for $i = 1, \ldots, n$; $\beta \in \{0,1\}$. Observe that the values $A_{i,\beta}$ are distributed identically as in the real scheme. In addition $\mathcal{B}$ picks random $\xi \in \mathbb{Z}_p$ and *implicitly* sets $\alpha = \xi + \prod_{h=1}^{l+1} c_{n+h}$. For enhancing readability we define $\gamma(u,v) = \prod_{h=u}^{v} c_h$ for positive integers u and v. Then, $\mathcal{B}$'s view point of α is $\alpha = \xi + \gamma(n+1, n+l+1)$. $\mathcal{B}$ computes $H = e(C_{n+1}, \ldots, C_{n+l+1}) g_{l+1}^{\xi} = g_{l+1}^{\alpha}$. $\mathcal{B}$ hands the public parameters PP consisting of the group sequence description together with $H, \{A_{i,\beta}\}_{i=1,\ldots,n;\ \beta\in\{0,1\}}$ to $\mathcal{A}$.

Query Phase 1 and **Query Phase 2**: Both the key query phases are executed in the same manner by $\mathcal{B}$. So, we describe them once here. $\mathcal{A}$ queries a decryption key for a circuit $f = (n, q, l, \mathbb{A}, \mathbb{B}, \mathsf{GateType})$ to $\mathcal{B}$ subject to the restriction that $f(x^*) = 0$. As in [GGH$^+$13b], we will think of the proof as having some invariant property on the depth of the wire we are looking at. Consider a wire w with $\mathsf{depth}(w) = j$ and $\mathcal{B}$'s view point (symbolically) of r_w. If $f_w(x^*) = 0$, then $\mathcal{B}$ will *implicitly* view r_w as the term $\gamma(n+1, n+j+1)$ plus some additional known randomization term. On the other hand, if $f_w(x^*) = 1$ then $\mathcal{B}$ will view r_w as 0 plus some additional known randomization term. Keeping this property intact for simulating the keys up the circuit, $\mathcal{B}$ will ultimately view r_{n+q} as $\gamma(n+1, n+l+1)$ plus some additional known randomization term since $f_{n+q}(x^*) = f(x^*) = 0$. As will be demonstrated shortly, this would allow $\mathcal{B}$ to simulate the header component $\mathcal{K}$ by cancelation.

The bottom up simulation of the key component for each wire w by $\mathcal{B}$ varies depending on whether w is an Input wire, an OR gate, or an AND gate.

- <u>Input wire</u>: Consider $w \in \{1, \ldots, n\}$, i.e., an input wire.
  - If $x_w^* = 1$, then $\mathcal{B}$ picks random $r_w \in \mathbb{Z}_p$ (as is done honestly) and sets the key component
$$\mathcal{K}_w = e(C_w, g_1)^{r_w} = g_2^{r_w c_w} = g_2^{r_w a_{w,1}}.$$
  - Otherwise, if $x_w^* = 0$, then $\mathcal{B}$ *implicitly* lets $r_w = \gamma(n+1, n+2) + \eta_w$, where $\eta_w \in \mathbb{Z}_p$ is randomly selected by $\mathcal{B}$, and sets the key component
$$\mathcal{K}_w = (e(C_{n+1}, C_{n+2}) g_2^{\eta_w})^{z_w} = g_2^{(\gamma(n+1,n+2)+\eta_w) z_w} = g_2^{r_w a_{w,1}}.$$
- <u>OR gate</u>: Consider a wire $w \in \mathsf{Gates}$ with $\mathsf{GateType}(w) = \mathsf{OR}$ and $j = \mathsf{depth}(w)$.

– If $f_w(x^*) = 1$, then $f_{\mathbb{A}(w)}(x^*) = 1$ or $f_{\mathbb{B}(w)}(x^*) = 1$. $\mathcal{B}$ chooses random $b_w, d_w, r_w \in \mathbb{Z}_p$ as in the real scheme, and forms the key component as

$$\mathcal{K}_w = \Big(K_{w,1} = g_1^{b_w}, K_{w,2} = g_1^{d_w}, K_{w,3} = g_j^{r_w - b_w r_{\mathbb{A}(w)}}, K_{w,4} = g_j^{r_w - d_w r_{\mathbb{B}(w)}}\Big).$$

Observe that, due to the bottom up simulation, $r_{\mathbb{A}(w)}$ and $r_{\mathbb{B}(w)}$ are already selected or implicitly set by $\mathcal{B}$ according as the corresponding gates, i.e., $\mathbb{A}(w)$ and $\mathbb{B}(w)$, evaluate to 1 or 0 upon input x^*. Note that even if $\mathbb{A}(w)$ or $\mathbb{B}(w)$ gate evaluates to 0 upon input x^*, $\mathcal{B}$ can still simulate its corresponding component, i.e., $K_{w,3}$ or $K_{w,4}$ in $\mathcal{K}_w$ using multilinear map. For instance, $f_{\mathbb{A}(w)}(x^*) = 0$ implies $r_{\mathbb{A}(w)}$ has been implicitly set as $\gamma(n{+}1, n{+}j){+}\eta_{\mathbb{A}(w)}$ by $\mathcal{B}$, as $\mathsf{depth}(\mathbb{A}(w)) = j{-}1$ for the reason that our circuit is layered. Thus, in this case $\mathcal{B}$ can create $K_{w,3}$ as $K_{w,3} = e(C_{n+1}, \ldots, C_{n+j})^{-b_w} g_j^{r_w} = g_j^{r_w - b_w r_{\mathbb{A}(w)}}$.

– On the other hand, if $f_w(x^*) = 0$, then $f_{\mathbb{A}(w)}(x^*) = f_{\mathbb{B}(w)}(x^*) = 0$. $\mathcal{B}$ picks random $\psi_w, \phi_w, \eta_w \in \mathbb{Z}_p$, *implicitly* sets $b_w = c_{n+j+1} + \psi_w, d_w = c_{n+j+1} + \phi_w$, and $r_w = \gamma(n + 1, n{+}j{+}1){+}\eta_w$, and creates the decryption key component $\mathcal{K}_w = (K_{w,1}, K_{w,2}, K_{w,3}, K_{w,4})$ where

$$\begin{aligned}
K_{w,1} &= C_{n+j+1} g_1^{\psi_w} = g_1^{b_w}, K_{w,2} = C_{n+j+1} g_1^{\phi_w} = g_1^{d_w},\\
K_{w,3} &= e(C_{n+j+1}, g_{j-1})^{-\eta_{\mathbb{A}(w)}} e(C_{n+1}, \ldots, C_{n+j})^{-\psi_w} g_j^{\eta_w - \psi_w \eta_{\mathbb{A}(w)}}\\
&= g_j^{\eta_w - c_{n+j+1}\eta_{\mathbb{A}(w)} - \psi_w(\gamma(n+1,n+j)+\eta_{\mathbb{A}(w)})} = g_j^{r_w - b_w r_{\mathbb{A}(w)}},\\
K_{w,4} &= e(C_{n+j+1}, g_{j-1})^{-\eta_{\mathbb{B}(w)}} e(C_{n+1}, \ldots, C_{n+j})^{-\phi_w} g_j^{\eta_w - \phi_w \eta_{\mathbb{B}(w)}}\\
&= g_j^{\eta_w - c_{n+j+1}\eta_{\mathbb{B}(w)} - \phi_w(\gamma(n+1,n+j)+\eta_{\mathbb{B}(w)})} = g_j^{r_w - d_w r_{\mathbb{B}(w)}}.
\end{aligned}$$

Note that according to our bottom up simulation, $r_{\mathbb{A}(w)}$ has been implicitly set as $r_{\mathbb{A}(w)} = \gamma(n + 1, n + j) + \eta_{\mathbb{A}(w)}$ by $\mathcal{B}$ and similarly for $r_{\mathbb{B}(w)}$. Therefore,

$$\begin{aligned}
r_w - b_w r_{\mathbb{A}(w)} &= (\gamma(n + 1, n + j + 1) + \eta_w) - (c_{n+j+1} + \psi_w)(\gamma(n + 1, n + j) + \eta_{\mathbb{A}(w)})\\
&= \eta_w - c_{n+j+1}\eta_{\mathbb{A}(w)} - \psi_w(\gamma(n + 1, n + j) + \eta_{\mathbb{A}(w)})
\end{aligned}$$

which enables $\mathcal{B}$ to simulate $K_{w,3}$ and analogously $K_{w,4}$ in this case.

• $\underline{\mathsf{AND}\text{ gate}}$: Consider wire $w \in \mathsf{Gates}$ with $\mathsf{GateType}(w) = \mathsf{AND}$ and $j = \mathsf{depth}(w)$.

– If $f_w(x^*) = 1$, then $f_{\mathbb{A}(w)}(x^*) = f_{\mathbb{B}(w)}(x^*) = 1$. $\mathcal{B}$ selects random $b_w, d_w, r_w \in \mathbb{Z}_p$ and forms the key component as

$$\mathcal{K}_w = \Big(K_{w,1} = g_1^{b_w}, K_{w,2} = g_1^{d_w}, K_{w,3} = g_j^{r_w - b_w r_{\mathbb{A}(w)} - d_w r_{\mathbb{B}(w)}}\Big).$$

Notice that since $f_{\mathbb{A}(w)}(x^*) = f_{\mathbb{B}(w)}(x^*) = 1$, $r_{\mathbb{A}(w)}$ and $r_{\mathbb{B}(w)}$ are random values which have already been chosen by $\mathcal{B}$ in the course of simulation.

– Alternatively, if $f_w(x^*) = 0$, then $f_{\mathbb{A}(w)}(x^*) = 0$ or $f_{\mathbb{B}(w)}(x^*) = 0$. If $f_{\mathbb{A}(w)}(x^*) = 0$, then $\mathcal{B}$ selects random $\psi_w, \phi_w, \eta_w \in \mathbb{Z}_p$, *implicitly* lets $b_w = c_{n+j+1} + \psi_w, d_w = \phi_w$, and $r_w = \gamma(n + 1, n + j + 1) + \eta_w$, and forms the decryption key component as $\mathcal{K}_w = (K_{w,1}, K_{w,2}, K_{w,3})$ where

$$\begin{aligned}
K_{w,1} &= C_{n+j+1} g_1^{\psi_w} = g_1^{b_w}, K_{w,2} = g_1^{\phi_w} = g_1^{d_w},\\
K_{w,3} &= e(C_{n+j+1}, g_{j-1})^{-\eta_{\mathbb{A}(w)}} e(C_{n+1}, \ldots, C_{n+j})^{-\psi_w} g_j^{\eta_w - \psi_w \eta_{\mathbb{A}(w)} - \phi_w r_{\mathbb{B}(w)}}\\
&= g_j^{\eta_w - c_{n+j+1}\eta_{\mathbb{A}(w)} - \psi_w(\gamma(n+1,n+j)+\eta_{\mathbb{A}(w)}) - \phi_w r_{\mathbb{B}(w)}}\\
&= g_j^{r_w - b_w r_{\mathbb{A}(w)} - d_w r_{\mathbb{B}(w)}}.
\end{aligned}$$

Observe that $\mathcal{B}$ can form $K_{w,3}$ due to a similar cancelation as explained in case of OR gates since, the $\mathbb{A}(w)$ gate being evaluated to 0, $r_{\mathbb{A}(w)} = \gamma(n+1, n+j) + \eta_{\mathbb{A}(w)}$ has already been implicitly set by $\mathcal{B}$. Moreover, $g_j^{r_{\mathbb{B}(w)}}$ is always computable by $\mathcal{B}$ from the available information regardless of whether $f_{\mathbb{B}(w)}(x^*) = 1$, in which case $r_{\mathbb{B}(w)}$ is a random value chosen by $\mathcal{B}$ itself, or $f_{\mathbb{B}(w)}(x^*) = 0$, for which $r_{\mathbb{B}(w)}$ has been implicitly set to be $r_{\mathbb{B}(w)} = \gamma(c_{n+1}, c_{n+j}) + \eta_{\mathbb{B}(w)}$ by $\mathcal{B}$ and, hence, $\mathcal{B}$ can compute $e(C_{n+1}, \ldots, C_{n+j}) g_j^{\eta_{\mathbb{B}(w)}} = g_j^{r_{\mathbb{B}(w)}}$. The case where $f_{\mathbb{B}(w)}(x^*) = 0$ and $f_{\mathbb{A}(w)}(x^*) = 1$ is performed in a symmetric manner, with the roles of b_w and d_w reversed.

Since $f(x^*) = f_{n+q}(x^*) = 0$, r_{n+q} at the output gate is *implicitly* set as $\gamma(n+1, n+l+1) + \eta_{n+q}$ by $\mathcal{B}$. This allows $\mathcal{B}$ to perform a final cancelation in computing the "header" component of the key as $\mathcal{K} = g_l^{\xi - \eta_{n+q}} = g_l^{\alpha - r_{n+q}}$. $\mathcal{B}$ provides the decryption key $\mathsf{SK}_f^{(\mathsf{DEC})} = (f, \mathcal{K}, \{\mathcal{K}_w\}_{w \in \{1,\ldots,n+q\}})$ to $\mathcal{A}$.

Challenge: $\mathcal{A}$ submits two challenge messages $M_0^*, M_1^* \in \mathbb{G}_k$ to $\mathcal{B}$. $\mathcal{B}$ flips a random coin $b \in \{0,1\}$, *implicitly* views $\overline{s}$ as the randomness used in generating the challenge ciphertext, and sets challenge ciphertext

$$\mathsf{CT}^* = (x^*, C_M^* = T_{\overline{b}} e(\overline{S}, C_1, \ldots, C_n, g_l^{\xi}) M_b^* = T_{\overline{b}} g_k^{\xi \overline{s} \gamma(1,n)} M_b^*, C^* = \overline{S}),$$

and gives it to $\mathcal{A}$.

Guess: $\mathcal{B}$ eventually receives back the guess $b' \in \{0,1\}$ from $\mathcal{A}$. If $b = b'$, $\mathcal{B}$ outputs $\overline{b}' = 1$; otherwise, it outputs $\overline{b}' = 0$.

Note that if $\overline{b} = 0$, the challenge ciphertext CT^* is properly generated by $\mathcal{B}$. On the other hand, if $\overline{b} = 1$ the challenge ciphertext is random. Hence the theorem. □

B Security Analysis of the ABSC Scheme of Section 4

B.1 Security Definition for ABSC

The notions of message confidentiality and ciphertext unforgeability for ABSC supporting arbitrary circuits are discussed below:

(I) **Message Confidentiality**: We define this security notion on indistinguishability of ciphertexts under chosen plaintext attack (CPA) in the *selective* encryption input string model through the following game between a probabilistic adversary $\mathcal{A}$ and a probabilistic challenger $\mathcal{B}$:

Init: $\mathcal{A}$ commits to a challenge encryption input string $x^* \in \{0,1\}^n$ that will be used to create the challenge ciphertext.

Setup: $\mathcal{B}$ performs $\mathsf{ABSC.Setup}(1^\lambda, n, n', l)$ to obtain PP, MK and hands PP to $\mathcal{A}$.

Query Phase 1: $\mathcal{A}$ may adaptively make any polynomial number of queries which may be of the following types to be answered by $\mathcal{B}$.

- ▷ *Signing key query*: Upon receiving a signing key query corresponding to a circuit $g \in \mathbb{F}_{n',l}^{(\mathsf{SIG})}$ from $\mathcal{A}$, $\mathcal{B}$ returns $\mathsf{SK}_g^{(\mathsf{SIG})}$ by running $\mathsf{ABSC.SKeyGen}(\mathsf{PP}, \mathsf{MK}, g)$.
- ▷ *Decryption key query*: When $\mathcal{A}$ queries a decryption key for a circuit $f \in \mathbb{F}_{n,l}^{(\mathsf{DEC})}$ subject to the constraint that $f(x^*) = 0$, $\mathcal{B}$ provides $\mathsf{SK}_f^{(\mathsf{DEC})}$ to $\mathcal{A}$ by executing $\mathsf{ABSC.DKeyGen}(\mathsf{PP}, \mathsf{MK}, f)$.

▷ *Signcryption query*: In response to a signcryption query made by $\mathcal{A}$ for a message M, a signature input string $y \in \{0,1\}^{n'}$, and an encryption input string $x \in \{0,1\}^n$, $\mathcal{B}$ samples a signing policy circuit $g \in \mathbb{F}_{n',l}^{(\mathsf{SIG})}$ such that $g(y) = 1$ and sends the ciphertext $\mathsf{CT}_{x,y}$ to $\mathcal{A}$ by performing $\mathsf{ABSC.Signcrypt}(\mathsf{PP}, \mathsf{SK}_g^{(\mathsf{SIG})}, x, y, M)$. Here $\mathsf{SK}_g^{(\mathsf{SIG})}$ is obtained from $\mathsf{ABSC.SKeyGen}(\mathsf{PP}, \mathsf{MK}, g)$ by $\mathcal{B}$.

Challenge: $\mathcal{A}$ submits two equal length messages M_0^*, M_1^* and a signature input string y^*. $\mathcal{B}$ picks a signing policy circuit $g^* \in \mathbb{F}_{n',l}^{(\mathsf{SIG})}$ such that $g^*(y^*) = 1$ and gives the challenge ciphertext CT^* to $\mathcal{A}$ by executing $\mathsf{ABSC.Signcrypt}(\mathsf{PP}, \mathsf{ABSC.SKeyGen}(\mathsf{PP}, \mathsf{MK}, g^*), x^*, y^*, M_b^*)$ where $b \in \{0,1\}$ is a random coin chosen by $\mathcal{B}$.

Query Phase 2: $\mathcal{A}$ may continue adaptively to make queries as in **Query Phase 1** subject to the same restrictions as earlier and $\mathcal{B}$ keeps on answering those queries.

Guess: $\mathcal{A}$ eventually outputs a guess b' for b and wins the game if $b' = b$.

The advantage of the adversary $\mathcal{A}$ in the above game is defined as

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{ABSC,s\text{-}IND\text{-}CPA}}(\lambda) = |\mathsf{Pr}[b' = b] - 1/2|.$$

Definition 2. *An* ABSC *scheme for circuits is defined to be selectively message confidential against* CPA *if for all* PPT *adversaries* $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{ABSC,s\text{-}IND\text{-}CPA}}(\lambda)$ *is at most negligible.*

(II) **Ciphertext Unforgeability**: We define this notion of security on existential unforgeability under adaptive chosen message attack (CMA) in the *selective* signature input string model through the following game between a probabilistic adversary $\mathcal{A}$ and a probabilistic challenger $\mathcal{B}$.

Init: $\mathcal{A}$ declares a signature input string $y^* \in \{0,1\}^{n'}$ to $\mathcal{B}$ that will be used to forge a signcryption.

Setup: $\mathcal{B}$ runs $\mathsf{ABSC.Setup}(1^\lambda, n, n', l)$ to obtain PP, MK and hands PP to $\mathcal{A}$.

Query Phase: $\mathcal{A}$ may adaptively make a polynomial number of queries of the following types to $\mathcal{B}$ and $\mathcal{B}$ provides the answer to them.

▷ *Signing key query*: Upon receiving a signing key query from $\mathcal{A}$ corresponding to a signing policy circuit $g \in \mathbb{F}_{n',l}^{(\mathsf{SIG})}$ subject to the constraint that $g(y^*) = 0$, $\mathcal{B}$ returns the $\mathsf{SK}_g^{(\mathsf{SIG})}$ to $\mathcal{A}$ by executing $\mathsf{ABSC.SKeyGen}(\mathsf{PP}, \mathsf{MK}, g)$.

▷ *Decryption key query*: When $\mathcal{A}$ queries a decryption key for a decryption policy circuit $f \in \mathbb{F}_{n,l}^{(\mathsf{DEC})}$, $\mathcal{B}$ gives $\mathsf{SK}_f^{(\mathsf{DEC})}$ to $\mathcal{A}$ by performing $\mathsf{ABSC.DKeyGen}(\mathsf{PP}, \mathsf{MK}, f)$.

▷ *Signcryption query*: $\mathcal{A}$ queries a signcryption of a message M for a signature input string $y(\neq y^*) \in \{0,1\}^{n'}$ along with an encryption input string $x \in \{0,1\}^n$. $\mathcal{B}$ samples a signing policy circuit $g \in \mathbb{F}_{n',l}^{(\mathsf{SIG})}$ such that $g(y) = 1$ and returns the ciphertext $\mathsf{CT}_{x,y}$ to $\mathcal{A}$ by performing $\mathsf{ABSC.Signcrypt}(\mathsf{PP}, \mathsf{SK}_g^{(\mathsf{SIG})}, x, y, M)$, where $\mathsf{SK}_g^{(\mathsf{SIG})}$ is got from $\mathsf{ABSC.SKeyGen}(\mathsf{PP}, \mathsf{MK}, g)$ by $\mathcal{B}$.

- ▷ *Unsigncryption query*: In response to a unsigncryption query from $\mathcal{A}$ for a ciphertext $\mathsf{CT}_{x,y}$ under the decryption policy circuit $f \in \mathbb{F}_{n,l}^{(\mathsf{DEC})}$, $\mathcal{B}$ obtains the decryption key $\mathsf{SK}_f^{(\mathsf{DEC})}$ by running $\mathsf{ABSC.DkeyGen}(\mathsf{PP}, \mathsf{MK}, f)$ and sends output of $\mathsf{ABSC.Unsigncrypt}(\mathsf{PP}, \mathsf{CT}_{x,y}, \mathsf{SK}_f^{(\mathsf{DEC})})$ to $\mathcal{A}$.

Forgery: $\mathcal{A}$ eventually outputs a forgery CT^* for some message M^* with the signature input string y^* and an encryption input string x^*. $\mathcal{A}$ wins the game if the ciphertext CT^* is valid, i.e., $M^*(\neq \perp)$ is the output of $\mathsf{ABSC.Unsigncrypt}(\mathsf{PP}, \mathsf{CT}^*, \mathsf{SK}_{f^*}^{(\mathsf{DEC})})$ for any $f^* \in \mathbb{F}_{n,l}^{(\mathsf{DEC})}$ satisfying $f^*(x^*) = 1$, and CT^* is not obtained from any signcryption query to $\mathcal{B}$.

The advantage of $\mathcal{A}$ in the above game is defined as

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{ABSC,s\text{-}UF\text{-}CMA}}(\lambda) = \mathsf{Pr}[\mathcal{A} \text{ wins}].$$

Definition 3. *An* ABSC *scheme for circuits is defined to be selectively ciphertext unforgeable against* CMA *if for all* PPT *adversaries* $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{ABSC,s\text{-}UF\text{-}CMA}}(\lambda)$ *is at most negligible.*

B.2 Proofs of Theorems 2 and 3

Theorem 2 (Message Confidentiality of ABSC). *The* ABSC *scheme supporting arbitrary decryption policy circuits of input length* n *and depth* l*, as well as, arbitrary signing policy circuits of input length* n' *and the same depth* l*, described in Section 4, achieves selective message confidentiality against* CPA *as per the model of Appendix B.1 under the* k-MDDH *assumption, where* $k = n + n' + l + 1$*. More precisely, for any* PPT *adversary* $\mathcal{A}$ *against the* ABSC *scheme of Section 4, there exists a probabilistic algorithm* $\mathcal{B}$*, whose the running time is essentially the same as that of* $\mathcal{A}$*, such that for any security parameter* λ, $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{ABSC,s\text{-}IND\text{-}CPA}}(\lambda) = \mathsf{Adv}_{\mathcal{B}}^{k\text{-}\mathsf{MDDH}}(\lambda)$.

Proof. Suppose there exists a PPT adversary $\mathcal{A}$ that breaks with non-negligible advantage the selective CPA message confidentiality of the ABSC scheme supporting decryption policy circuit of input length n and depth l, as well as, signing policy circuits of input length n' and same depth l, described in Section 4. We construct a PPT algorithm $\mathcal{B}$ that attempts to solve an instance of the k-MDDH problem, where $k = n+n'+l+1$, using $\mathcal{A}$ as a sub-routine. $\mathcal{B}$ is given an instance of the k-MDDH problem $\varrho_{\bar{b}} = (\vec{\mathbb{G}}, g_1, \overline{S}, C_1, \ldots, C_k, T_{\bar{b}})$ such that $\overline{S} = g_1^{\bar{s}}, C_1 = g_1^{c_1}, \ldots, C_k = g_1^{c_k}$. $\mathcal{B}$ plays the role of the challenger in the selective CPA message confidentiality game of Appendix B.1 and interacts with $\mathcal{A}$ as follows:

Init: $\mathcal{A}$ declares the challenge encryption input string $x^* = x_1^* \ldots x_n^* \in \{0,1\}^n$ to $\mathcal{B}$.

Setup: $\mathcal{B}$ chooses random $z_1, \ldots, z_n \in \mathbb{Z}_p$ and sets $a_{i,\beta} = c_i$ *implicitly*, if $\beta = x_i^*$, while $a_{i,\beta} = z_i$, if $\beta \neq x_i^*$, for $i = 1, \ldots, n;\ \beta \in \{0,1\}$. This corresponds to setting $A_{i,\beta} = C_i = g_1^{c_i}$, if $\beta = x_i^*$, whereas, $A_{i,\beta} = g_1^{z_i}$, if $\beta \neq x_i^*$, for $i = 1, \ldots, n;\ \beta \in \{0,1\}$. $\mathcal{B}$ also picks random $z'_1, \ldots, z'_{n'} \in \mathbb{Z}_p$ and *implicitly* sets $b_{t,\beta} = c_{n+t}$, if $\beta = 1$, whereas $b_{t,\beta} = c_{n+t} + z'_t$, if $\beta = 0$, for $t = 1, \ldots, n'$. This is equivalent to setting $B_{t,\beta} = C_{n+t}$, if $\beta = 1$, and $B_{t,\beta} = C_{n+t} g_1^{z'_t}$, if $\beta = 0$, for $t = 1, \ldots, n'$. Observe that the values $A_{i,\beta}$ and $B_{t,\beta}$ are distributed identically as in the real scheme. Here also we will use the notation $\gamma(u,v)$ to denote $\prod_{h=u}^{v} c_h$ for integers u and v. In addition, $\mathcal{B}$ selects $\xi_1 \in \mathbb{Z}_p$, *implicitly* views $\alpha_1 = \xi_1 + \gamma(n+n'+1, n+n'+l+1)$, and computes $H = e(C_{n+n'+1}, \ldots, C_{n+n'+l+1}) g_{l+1}^{\xi_1} = g_{l+1}^{\alpha_1}$. Moreover, $\mathcal{B}$ chooses random $\theta, \alpha_2 \in \mathbb{Z}_p$ and computes $\Theta = g_n^{\theta}, Y = g_{n+l+1}^{\theta\alpha_2}$ as is done in the real scheme. $\mathcal{B}$ hands the public parameters PP consisting of the group sequence description together with $\{A_{i,\beta}\}_{i=1,\ldots,n;\ \beta\in\{0,1\}}, \{B_{t,\beta}\}_{t=1,\ldots,n';\ \beta\in\{0,1\}}, H, \Theta, Y$ to $\mathcal{A}$. Observe that all the simulated PP

components are identically distributed as in the original scheme.

Query Phase 1 and **Query Phase 2**: Both the query phases are executed in same fashion by $\mathcal{B}$, so we describe them once here.

- ▷ *Signing key query*: Note that $\mathcal{B}$ knows α_2 and $\{B_{t,\beta}\}_{t=1,\ldots,n';\ \beta\in\{0,1\}}$, therefore, $\mathcal{B}$ can provide signing keys corresponding to any signing policy circuit $g = (n', q', l, \mathbb{A}, \mathbb{B}, \mathsf{GateType})$ queried by $\mathcal{A}$.
- ▷ *Decryption key query*: $\mathcal{A}$ queries a decryption key for a decryption policy circuit $f = (n, q, l, \mathbb{A}, \mathbb{B}, \mathsf{GateType})$ to $\mathcal{B}$ subject to the restriction that $f(x^*) = 0$. Note that the structure of the decryption keys of our ABSC and ABE constructions are analogous. Thus, as for the simulation in the proof of Theorem 1, we will maintain an invariant property on the depth of the wire of f we are looking at. Consider a wire w with $\mathsf{depth}(w) = j$. If $f_w(x^*) = 0$ then $\mathcal{B}$ will view r_w as the term $\gamma(n + n' + 1, n + n' + j + 1)$ plus some additional known randomization term. On the other hand, if $f_w(x^*) = 1$ then $\mathcal{B}$ will view r_w as 0 plus some additional known randomization term. The key components $\mathcal{K}_w$ corresponding to the input wires, OR, and AND gates are simulated in an identical manner as in the proof of Theorem 1. Thus, $\mathcal{B}$ will ultimately view $r_{n+q} = \gamma(n + n' + 1, n + n' + l + 1) + \eta_{n+q}$, where $\eta_{n+q} \in \mathbb{Z}_p$ is randomly chosen by $\mathcal{B}$, and computes the header component $\mathcal{K} = g_l^{\xi_1+\alpha_2-\eta_{n+q}} = g_l^{\alpha-r_{n+q}}$. $\mathcal{B}$ returns the queried decryption key $\mathsf{SK}_f^{(\mathsf{DEC})} = (f, \mathcal{K}, \{\mathcal{K}_w\}_{w\in\{1,\ldots,n+q\}})$ to $\mathcal{A}$.
- ▷ *Signcryption query*: $\mathcal{A}$ queries a signcryption of some message M corresponding to some encryption input string $x = x_1 \ldots x_n \in \{0,1\}^n$ and signing input string $y = y_1 \ldots y_{n'} \in \{0,1\}^{n'}$. $\mathcal{B}$ knows ξ_1, α_2, extracts $\Theta, \{A_{i,x_i}\}_{i=1,\ldots,n}, \{B_{t,y_t}\}_{t=1,\ldots,n'}$ from PP, and uses the elements $\{C_{n+n'+h}\}_{h=1,\ldots,l+1}$ from the given k-MDDH instance to simulate the query as follows: $\mathcal{B}$ picks random $s \in \mathbb{Z}_p$ and computes

$$\begin{aligned} C_M = &\ \Big(e(C_{n+n'+1}, \ldots, C_{n+n'+l+1}, A_{1,x_1}, \ldots, A_{n,x_n}, B_{1,y_1}, \ldots, B_{n',y_{n'}})\cdot \\ &\qquad e(g_{l+1}^{\xi_1+\alpha_2}, A_{1,x_1}, \ldots, A_{n,x_n}, B_{1,y_1}, \ldots, B_{n',y_{n'}})\Big)^s M \\ = &\ \Big(g_k^{\gamma(n+n'+1,n+n'+l+1)\delta(x)\delta'(y)} \cdot g_k^{(\xi_1+\alpha_2)\delta(x)\delta'(y)}\Big)^s M = g_k^{\alpha s\delta(x)\delta'(y)} M, \\ C = &\ g_1^s,\ C' = \ e(g_l^{\alpha_2}, \Theta, B_{1,y_1}, \ldots, B_{n',y_{n'}}) = g_{k-1}^{\theta\alpha_2\delta'(y)}, \end{aligned}$$

  where $\alpha = \alpha_1 + \alpha_2 = \gamma(n + n' + 1, n + n' + l + 1) + \xi_1 + \alpha_2$, $\delta(x) = \prod_{i=1}^{n} a_{i,x_i}$, and $\delta'(y) = \prod_{t=1}^{n'} b_{t,y_t}$. $\mathcal{B}$ gives the ciphertext $\mathsf{CT}_{x,y} = (x, y, C_M, C, C')$ to $\mathcal{A}$.

Challenge: $\mathcal{A}$ submits two challenge messages $M_0^*, M_1^* \in \mathbb{G}_k$ along with a signature input string $y^* = y_1^* \ldots y_{n'}^* \in \{0,1\}^{n'}$. $\mathcal{B}$ flips a random coin $b \in \{0,1\}$, *implicitly* views $\overline{s}$ as the randomness in creation of the challenge ciphertext and computes components of the challenge ciphertext as

$$C_M^* = T_{\overline{b}} Z M_b^*, C^* = \overline{S}, C'^* = e(g_l^{\alpha_2}, \Theta, B_{1,y_1^*}, \ldots, B_{n',y_{n'}^*}),$$

where $Z = g_k^{\vartheta}$ such that $\vartheta = \alpha\overline{s}\delta(x^*)\delta'(y^*) - \overline{s}\gamma(1,k) = (\gamma(n + n' + 1, n + n' + l + 1) + \xi_1 + \alpha_2)\overline{s}\gamma(1,n)\delta'(y^*) - \overline{s}\gamma(1,k)$. The fact that Z is computable by $\mathcal{B}$ from available information using the multilinear map can be understood from the following: Note that $\delta'(y^*) = \gamma(n+1, n+n') + \kappa$ where all the terms contained in κ include a product of at most $n' - 1$ number of c_{n+t}'s and, hence,

$$\begin{aligned} \vartheta =&(\gamma(n + n' + 1, n + n' + l + 1) + \xi_1 + \alpha_2)\overline{s}\gamma(1,n)(\gamma(n + 1, n + n') + \kappa) - \overline{s}\gamma(1, n + n' + l + 1) \\ =&\ \overline{s}(\gamma(1, n + n' + l + 1) + \sigma) - \overline{s}\gamma(1, n + n' + l + 1) = \overline{s}\sigma \end{aligned}$$

where σ consists of terms including a product of at most $k-1$ number of c_h's. $\mathcal{B}$ sends the challenge ciphertext $\mathsf{CT}^* = (x^*, y^*, C_M^*, C^*, C'^*)$ to $\mathcal{A}$.

Guess: $\mathcal{A}$ eventually outputs a bit b'. If $b = b'$ then, $\mathcal{B}$ outputs $\overline{b}' = 1$; otherwise, it outputs $\overline{b}' = 0$.

Note that if $\overline{b} = 0$ then the challenge ciphertext is properly generated by $\mathcal{B}$. On the other hand, if $\overline{b} = 1$ then the challenge ciphertext is completely random. Hence the theorem. □

Theorem 3 (Ciphertext Unforgeability of ABSC). *The* ABSC *scheme supporting arbitrary decryption policy circuits of input length n and depth l, as well as, arbitrary signing policy circuits of input length n' and depth l, described in Section 4, achieves selective ciphertext unforgeability against* CMA *as per the model of Appendix B.1 under the k-*MCDH *assumption, where $k = n + n' + l + 1$. More precisely, for any* PPT *adversary $\mathcal{A}$ against the* ABSC *scheme of Section 4, there exists a probabilistic algorithm $\mathcal{B}$, whose running time is essentially the same as that of $\mathcal{A}$, such that for any security parameter λ, $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{ABSC,s\text{-}UF\text{-}CMA}}(\lambda) = \mathsf{Adv}_{\mathcal{B}}^{k\text{-}\mathsf{MCDH}}(\lambda)$.*

Proof. Assume that there exist a PPT adversary $\mathcal{A}$ that breaks with non-negligible advantage the selective CMA ciphertext unforgeability of the ABSC scheme supporting decryption policy circuits of input length n and depth l, as well as, signing policy circuits of input length n' and the same depth l, proposed in Section 4. We construct a PPT algorithm $\mathcal{B}$ that attempts to solve an instance of the k-MCDH problem, where $k = n+n'+l+1$, using $\mathcal{A}$ as a sub-routine. $\mathcal{B}$ is given an instance of the k-MCDH problem $\varrho = (\overrightarrow{\mathbb{G}}, g_1, C_1, \ldots, C_k)$ such that $C_1 = g_1^{c_1}, \ldots, C_k = g_1^{c_k}$. $\mathcal{B}$ plays the role of the challenger in the selective CMA ciphertext unforgeability game of Appendix B.1 and interacts with $\mathcal{A}$ as follows:

Init: $\mathcal{A}$ declares a signature input string $y^* = y_1^* \ldots y_{n'}^* \in \{0,1\}^{n'}$ to $\mathcal{B}$ that will be used to forge a signcryption.

Setup: $\mathcal{B}$ picks random $a_{i,\beta} \in \mathbb{Z}_p$ and computes $A_{i,\beta} = g_1^{a_{i,\beta}}$ for $i = 1, \ldots, n;\ \beta \in \{0,1\}$ as is done in the original scheme. Further $\mathcal{B}$ selects random $z'_1, \ldots, z'_{n'} \in \mathbb{Z}_p$ and *implicitly* sets $b_{t,\beta} = c_{n+t}$, if $\beta = y_t^*$, and $b_{t,\beta} = z'_t$, if $\beta \neq y_t^*$, for $t = 1, \ldots, n';\ \beta \in \{0,1\}$. This corresponds to setting $B_{t,\beta} = C_{n+t}$, if $\beta = y_t^*$, while $B_{t,\beta} = g_1^{z'_t}$, if $\beta \neq y_t^*$, for $t = 1, \ldots, n';\ \beta \in \{0,1\}$. Additionally, $\mathcal{B}$ selects random $\alpha \in \mathbb{Z}_p$, *implicitly* lets $\theta = \gamma(1,n)$, $\alpha_1 = \alpha - \gamma(n+n'+1, n+n'+l+1)$, $\alpha_2 = \gamma(n+n'+1, n+n'+l+1)$, where $\gamma(u,v) = \prod_{h=u}^{v} c_h$ for integers u, v, and sets $\Theta = e(C_1, \ldots, C_n) = g_n^{\theta}$, $H = e(C_{n+n'+1}, \ldots, C_{n+n'+l+1})^{-1} g_{l+1}^{\alpha} = g_{l+1}^{\alpha_1}$, $Y = e(C_1, \ldots, C_n, C_{n+n'+1}, \ldots, C_{n+n'+l+1}) = g_{n+l+1}^{\theta\alpha_2}$. $\mathcal{B}$ hands the public parameters PP consisting of the group sequence description plus $\{A_{i,\beta}\}_{i=1,\ldots,n;\ \beta\in\{0,1\}}, \{B_{t,\beta}\}_{t=1,\ldots,n';\ \beta\in\{0,1\}}, H, \Theta, Y$ to $\mathcal{A}$. Note that all the simulated PP components are identically distributed as in the original scheme.

Query Phase: $\mathcal{A}$ issues a series of queries to which $\mathcal{B}$ answers as follows:

▷ *Signing key query*: $\mathcal{A}$ queries a signing key for a circuit $g = (n', q', l, \mathbb{A}, \mathbb{B}, \mathsf{GateType})$ subject to the constraint that $g(y^*) = 0$. $\mathcal{B}$ proceeds to generate the key components from the bottom up the circuit as described below. Here also we will think the simulation to have some invariant property on the depth of the wire we are looking at. Consider a wire w with $\mathsf{depth}(w) = j$. If $g_w(y^*) = 0$, then $\mathcal{B}$ will view r'_w as $\gamma(n+n'+1, n+n'+j+1)$ plus some additional known randomization term, while if $g_w(y^*) = 1$ then $\mathcal{B}$ will view r'_w as 0 plus some additional known randomization term. Keeping this property intact up the circuit, $\mathcal{B}$ will *implicitly* set $r'_{n'+q'} = \gamma(n+n'+1, n+n'+l+1) = \alpha_2$ as $g_{n'+q'}(y^*) = g(y^*) = 0$. We describe how $\mathcal{B}$ creates the signing key components for each wire w organizing the simulation into the following cases:

- Input wire: Suppose $w \in \{1, \ldots, n'\}$, i.e., an input wire.
  - If $y^*_w = 1$, then $\mathcal{B}$ chooses random $r'_w \in \mathbb{Z}_p$ and computes
  $$\mathcal{K}'_w = e(C_{n+w}, g_1)^{r'_w} = g_2^{r'_w b_{w,1}}.$$
  - If $y^*_w = 0$, then $\mathcal{B}$ picks random $\eta''_w \in \mathbb{Z}_p$, *implicitly* lets $r'_w = \gamma(n+n'+1, n+n'+2)+\eta'_w$, and sets
  $$\mathcal{K}'_w = \big(e(C_{n+n'+1}, C_{n+n'+2})g_2^{\eta'_w}\big)^{z'_w} = g_2^{r'_w b_{w,1}}.$$
- OR gate: Consider a wire $w \in \mathsf{Gates}$ with $\mathsf{GateType}(w) = \mathsf{OR}$ and $j = \mathsf{depth}(w)$.
  - If $g_w(y^*) = 1$, then $g_{\mathbb{A}(w)}(y^*) = 1$ or $g_{\mathbb{B}(w)}(y^*) = 1$. $\mathcal{B}$ chooses random $b'_w, d'_w, r'_w \in \mathbb{Z}_p$ as in the real scheme, and creates the key component as
  $$\mathcal{K}'_w = \Big(K'_{w,1} = g_1^{b'_w}, K'_{w,2} = g_1^{d'_w}, K'_{w,3} = g_j^{r'_w - b'_w r'_{\mathbb{A}(w)}}, K'_{w,4} = g_j^{r'_w - d'_w r'_{\mathbb{B}(w)}}\Big).$$
  Observe that, due to the bottom up simulation, $r'_{\mathbb{A}(w)}$ and $r'_{\mathbb{B}(w)}$ are already selected or implicitly set by $\mathcal{B}$ according as the corresponding gates, i.e., $\mathbb{A}(w)$ and $\mathbb{B}(w)$, evaluate to 1 or 0 upon input y^*. Note that even if $\mathbb{A}(w)$ or $\mathbb{B}(w)$ gate evaluates to 0 upon input y^*, $\mathcal{B}$ can still simulate its corresponding component, i.e., $K'_{w,3}$ or $K'_{w,4}$ in $\mathcal{K}'_w$ using multilinear map in a similar fashion as in simulating the queried decryption key components for OR gates in analogous situation in proof of Theorem 1.
  - On the other hand, if $g_w(y^*) = 0$, then $g_{\mathbb{A}(w)}(y^*) = g_{\mathbb{B}(w)}(y^*) = 0$. $\mathcal{B}$ chooses random $\psi'_w, \phi'_w, \eta'_w \in \mathbb{Z}_p$, *implicitly* lets $b'_w = c_{n+n'+j+1} + \psi'_w, d'_w = c_{n+n'+j+1} + \phi'_w$, and $r'_w = \gamma(n+n'+1, n+n'+j+1) + \eta'_w$, and sets $\mathcal{K}'_w = (K'_{w,1}, K'_{w,2}, K'_{w,3}, K'_{w,4})$ where
  $$\begin{aligned}
  K'_{w,1} &= C_{n+n'+j+1} g_1^{\psi'_w} = g_1^{b'_w}, K'_{w,2} = C_{n+n'+j+1} g_1^{\phi'_w} = g_1^{d'_w},\\
  K'_{w,3} &= e(C_{n+n'+j+1}, g_{j-1})^{-\eta'_{\mathbb{A}(w)}} e(C_{n+n'+1}, \ldots, C_{n+n'+j})^{-\psi'_w} g_j^{\eta'_w - \psi'_w \eta'_{\mathbb{A}(w)}} = g_j^{r'_w - b'_w r'_{\mathbb{A}(w)}},\\
  K'_{w,4} &= e(C_{n+n'+j+1}, g_{j-1})^{-\eta'_{\mathbb{B}(w)}} e(C_{n+n'+1}, \ldots, C_{n+n'+j})^{-\phi'_w} g_j^{\eta'_w - \phi'_w \eta'_{\mathbb{B}(w)}} = g_j^{r'_w - d'_w r'_{\mathbb{B}(w)}}.
  \end{aligned}$$
  Observe that $\mathcal{B}$ can form $K'_{w,3}$ and $K'_{w,4}$ due to a cancelation analogous to the simulation of the decryption key components corresponding to OR gates in similar situation in proof of Theorem 1, since both the $\mathbb{A}(w)$ and $\mathbb{B}(w)$ gates being evaluated to 0, $r'_{\mathbb{A}(w)} = \gamma(n+n'+1, n+n'+j) + \eta'_{\mathbb{A}(w)}$ and similarly $r'_{\mathbb{B}(w)}$ have already been implicitly set by $\mathcal{B}$ in course of the bottom up simulation.

- AND gate: Consider a wire $w \in \mathsf{Gates}$ with $\mathsf{GateType}(w) = \mathsf{AND}$ and $j = \mathsf{depth}(w)$.
  - If $g'_w(y^*) = 1$, then $g_{\mathbb{A}(w)}(y^*) = g_{\mathbb{B}(w)}(y^*) = 1$. $\mathcal{B}$ selects random $b'_w, d'_w, r'_w \in \mathbb{Z}_p$ and forms the key component as
  $$\mathcal{K}'_w = \Big(K'_{w,1} = g_1^{b'_w}, K'_{w,2} = g_1^{d'_w}, K'_{w,3} = g_j^{r'_w - b'_w r'_{\mathbb{A}(w)} - d'_w r'_{\mathbb{B}(w)}}\Big).$$
  Notice that since $g_{\mathbb{A}(w)}(y^*) = g_{\mathbb{B}(w)}(y^*) = 1$, $r'_{\mathbb{A}(w)}$ and $r'_{\mathbb{B}(w)}$ are random values which have already been chosen by $\mathcal{B}$ in the course of the bottom-up simulation.
  - Alternatively, if $g_w(y^*) = 0$, then $g_{\mathbb{A}(w)}(y^*) = 0$ or $g_{\mathbb{B}(w)}(y^*) = 0$. If $g_{\mathbb{A}(w)}(y^*) = 0$, then $\mathcal{B}$ picks $\psi'_w, \phi'_w, \eta'_w \in \mathbb{Z}_p$, *implicitly* lets $b'_w = c_{n+n'+j+1} + \psi'_w, d'_w = \phi'_w$, and $r'_w = \gamma(n+n'+1, n+n'+j+1) + \eta'_w$, and forms $\mathcal{K}'_w = (K'_{w,1}, K'_{w,2}, K'_{w,3})$ where
  $$\begin{aligned}
  K'_{w,1} &= C_{n+n'+j+1} g_1^{\psi'_w} = g_1^{b'_w},\ K'_{w,2} = g_1^{\phi'_w} = g_1^{d'_w},\\
  K'_{w,3} &= e(C_{n+n'+j+1}, g_{j-1})^{-\eta'_{\mathbb{A}(w)}} e(C_{n+n'+1}, \ldots, C_{n+n'+j})^{-\psi'_w} g_j^{\eta'_w - \psi'_w \eta'_{\mathbb{A}(w)} - \phi'_w r'_{\mathbb{B}(w)}}\\
  &= g_j^{r'_w - b'_w r'_{\mathbb{A}(w)} - d'_w r'_{\mathbb{B}(w)}}.
  \end{aligned}$$

Note that $\mathcal{B}$ can generate $K'_{w,3}$ due to a similar cancelation as in the simulation of the decryption key components for AND gates in analogous scenario in the proof of Theorem 1 since, the $\mathbb{A}(w)$ gate being evaluated to 0, $\mathcal{B}$ has already set $r'_{\mathbb{A}(w)} = \gamma(n+n'+1, n+n'+j) + \eta'_{\mathbb{A}(w)}$ implicitly during the bottom up simulation. Moreover, notice that $g_j^{r'_{\mathbb{B}(w)}}$ is always computable for $\mathcal{B}$ regardless of whether $g_{\mathbb{B}(w)}(y^*)$ evaluates to 0 or 1 as $g_j^{\gamma(n+n'+1,n+n'+j)}$ is computable using the multilinear map from the available information for $\mathcal{B}$. The case where $g_{\mathbb{B}(w)}(y^*) = 0$ and $g_{\mathbb{A}(w)}(y^*) = 1$ is executed in a symmetric manner with the roles of b'_w and d'_w reversed.

We mention that at the output gate $n' + q'$, $\mathcal{B}$ will take the additional randomness $\eta'_{n'+q'}$ to be zero while setting $r'_{n'+q'}$. Observe that this would not prevent the distribution of the simulated signing keys from being identical to that of the real scheme. $\mathcal{B}$ provides the signing key $\mathsf{SK}_g^{(\mathsf{SIG})} = (g, \{\mathcal{K}'_w\}_{w\in\{1,\ldots,n'+q'\}})$ to $\mathcal{A}$.

- ▷ *Decryption key query*: Note that $\mathcal{B}$ knows α, therefore, $\mathcal{B}$ can provide the decryption key $\mathsf{SK}_f^{(\mathsf{DEC})}$ corresponding to any decryption policy circuit $f = (n, q, l, \mathbb{A}, \mathbb{B}, \mathsf{GateType})$ queried by $\mathcal{A}$.
- ▷ *Signcryption query*: $\mathcal{A}$ queries the signcryption of a message M relative to a signature input string $y = y_1 \ldots y_{n'} (\neq y^*) \in \{0,1\}^{n'}$ and an encryption input string $x = x_1 \ldots x_n \in \{0,1\}^n$. $\mathcal{B}$ chooses random $s \in \mathbb{Z}_p$ and computes

$$C_M = e(g_{l+1}^{\alpha}, A_{1,x_1}, \ldots, A_{n,x_n}, B_{1,y_1}, \ldots, B_{n',y_{n'}})^s M = g_k^{\alpha s \delta(x)\delta'(y)} M, \; C = g_1^s,$$

where $\delta(x) = \prod_{i=1}^{n} a_{i,x_i}$, $\delta'(y) = \prod_{t=1}^{n'} b_{t,y_t}$. $\mathcal{B}$ also computes C' as described below. Since $y \neq y^*$, there exists some $t \in \{1, \ldots, n'\}$ such that $y_t \neq y_t^*$ and, hence, $B_{t,y_t} = g_1^{z'_t}$ as per the simulation. $\mathcal{B}$ computes

$$C' = e(\Theta, C_{n+n'+1}, \ldots, C_{n+n'+l+1}, B_{1,x_1}, \ldots, B_{t-1,y_{t-1}}, B_{t+1,y_{t+1}}, \ldots, B_{n',y_{n'}})^{z'_t} = g_{n+n'+l}^{\theta\alpha_2\delta'(y)}.$$

$\mathcal{B}$ gives the ciphertext $\mathsf{CT}_{x,y} = (x, y, C_M, C, C')$ to $\mathcal{A}$.

- ▷ *Unsigncryption query*: Note that $\mathcal{B}$ can create the decryption key $\mathsf{SK}_f^{(\mathsf{DEC})}$ corresponding to any decryption policy circuit f. Therefore, when $\mathcal{A}$ queries the unsigncryption of a ciphertext $\mathsf{CT}_{x,y}$ under a decryption policy circuit f, $\mathcal{B}$ first computes $\mathsf{SK}_f^{(\mathsf{DEC})}$ and then provides the result of $\mathsf{ABSC.Unsigncrypt}(\mathsf{PP}, \mathsf{CT}_{x,y}, \mathsf{SK}_f^{(\mathsf{DEC})})$ to $\mathcal{A}$.

Forgery: $\mathcal{A}$ eventually produces a valid forgery $\mathsf{CT}^* = (x^*, y^*, C_M^*, C^*, C'^*)$ for some message M^* with an encryption input string x^* and the committed signature input string y^*. Then $\mathcal{B}$ solves the k-MCDH problem by outputting C'^*.

Note that, since CT^* is a valid forgery, we have

$$C'^* = g_{n+n'+l}^{\theta\alpha_2 \prod_{i'=1}^{n'} b_{i',y_{i'}}} = g_{k-1}^{\gamma(1,n)\gamma(n+n'+1,n+n'+l+1)\gamma(n+1,n+n')} = g_{k-1}^{\gamma(1,k)}$$

which is the desired answer of the k-MCDH problem instance given to $\mathcal{B}$. The theorem follows. □

C Realizing Ciphertext-Policy ABE and ABSC

- **On the Construction of ciphertext-policy ABE**: Applying the technique of *universal circuits* to our key-policy ABE construction of Section 3 in a manner analogous to [GGH+13b], [Att14], we can obtain a selectively secure ciphertext-policy ABE scheme for arbitrary circuits of bounded size achieving *short* ciphertext under the Multilinear Decisional Diffie-Hellman assumption. More specifically, consider a variant of a universal circuit U_x such that $U_x(\mathcal{C}) = \mathcal{C}(x)$, where $\mathcal{C}$ is a canonical representation of an arbitrary *bounded size circuit* by a *bounded size string.* In the ciphertext-policy setting, decryption key corresponds to specific inputs x, and ciphertext corresponds to circuits $\mathcal{C}$. Utilizing our key-policy construction we can implement this by providing decryption keys corresponding to circuits U_x. Thus, when a key is used to decrypt a ciphertext associated with a circuit $\mathcal{C}$, the decrypter will be successful if and only if $U_x(\mathcal{C}) = \mathcal{C}(x) = 1$, as desired. The number of ciphertext components will obviously be constant when we will perform encryption with the canonical representation string $\mathcal{C}$ for a circuit using our construction.

- **On the Construction of ciphertext-policy ABSC**: As for ABE, the method of universal circuits can be applied to our key-policy ABSC construction of Section 4 in order to develop a ABSC with ciphertext-policy and claimed predicate mechanism [WH11] for general bounded size circuits achieving *constant size ciphertext*, *selective message confidentiality* under the Multilinear Decisional Diffie-Hellman assumption, and *selective ciphertext unforgeability* based on the Multilinear Computational Diffie-Hellman assumption. Here again we consider the same variant of universal circuits U_x for encryption and U'_y for signing such that $U_x(\mathcal{C}) = \mathcal{C}(x)$ and $U'_y(\mathcal{C}') = \mathcal{C}'(y)$, where $\mathcal{C}$ and $\mathcal{C}'$ are canonical representations of arbitrary *bounded size circuits* by *bounded-size strings* characterizing respective policies embedded in ciphertexts and claimed predicates of signatures. In ciphertext-policy style ABSC decryption and signing keys correspond to specific inputs x and y respectively, whereas, ciphertexts are computed for circuits $\mathcal{C}$ and signatures are provided against claimed predicates $\mathcal{C}'$. Thus, using our ABSC construction, we can implement this by providing decryption and signing keys respectively for circuits U_x and U'_y enabling a successful decryption if and only if $U_x(\mathcal{C}) = \mathcal{C}(x) = 1$ while a successful signcryption generation if and only if $U'_y(\mathcal{C}') = \mathcal{C}'(y) = 1$. Observe that due to the use of our ABSC construction, we obtain constant number of ciphertext components in this case as well.