FABS: Fast Attribute-Based Signatures

Liqun Chen[1] Long Meng[1]* Yalan Wang[1] Nada El Kassem[1] Christopher JP Newton[1]
Yangguang Tian[1] Jodie Knapp[1] Constantin Cătălin Drăgan[1] Daniel Gardham[1]
Mark Manulis[2]
[1] *University of Surrey*
{liqun.chen, long.meng, yalan.wang, nada.elkassem, c.newton}@surrey.ac.uk
{yangguang.tian, j.knapp, c.dragan, daniel.gardham}@surrey.ac.uk
[2] *Universität der Bundeswehr München*
mark.manulis@unibw.de

Abstract

Attribute-based signatures (ABS) provide fine-grained control over who can generate digital signatures and have many real-world applications. This paper presents a pair of fast ABS schemes: one for Key-Policy ABS (KP-ABS) and another for Signature-Policy ABS (SP-ABS). Both schemes support expressive policies using Monotone Span Programs (MSP), and offer practical features such as large universe, arbitrary attributes, and adaptive security. Most notably, we provide the first implementation of MSP-based ABS schemes and demonstrate that our schemes achieve the best-known asymptotic and concrete performance in this domain. Asymptotically, key generation, signing and verification time scale linearly with the number of attributes; verification requires only two pairing operations. In concrete terms, for 100 attributes, our KP-ABS scheme performs key generation, signing, and verification in 0.16s, 0.10s, and 0.13s, respectively; our SP-ABS scheme achieves times of 0.082s, 0.26s, and 0.21s for the same operations.

1 Introduction

Attribute-Based Signatures (ABS) are a type of cryptographic primitive that enables flexible user authentication based on diverse user attributes while protecting user privacy. In an ABS scheme, a user gets secret signing keys associated with their attributes and uses them to sign a message. The resulting signature can be verified, confirming that the signer possesses sufficient attributes to satisfy an access policy without revealing other information.

In the literature, ABS schemes have found numerous applications, e.g., anonymous credential [1], attribute-based messaging [2], secret leaking [2], cloud computing [3], healthcare industry [4, 5], mobile crowdsensing [6], internet of vehicles [7], privacy-preserving authentication and attestation [8], named data networking architectures [9], e-voting [10], role-based or attribute-based access control [10], and blockchain smart contracts [11].

ABS has also been shown to enhance real-world privacy-preserving applications: (1) For anonymous credentials (e.g., Idemix [12], U-Prove [13] and also see ISO/IEC WD 24843 [14]), Kaaniche and Laurent [15] demonstrated that ABS enables efficient issuance and selective disclosure in anonymous credentials. (2) For attribute-based access control (e.g., Microsoft Azure [16], AWS [17] and also see NIST SP 800-162 [18]), Li *et al.* [19] discussed how to use ABS to achieve flexible access rights with signer privacy in attribute-based access control. These examples illustrate the growing relevance of ABS, though its standardisation and widespread deployment remain limited due to inefficiencies.

Depending on how an access policy is setup, ABS can be classified into Key-Policy ABS (KP-ABS), e.g., [20–22] and Signature-Policy ABS (SP-ABS), e.g., [23–25]. In KP-ABS, a policy is embedded into a signing key and a signature is associated with a set of attributes that satisfy the policy. In SP-ABS, a policy is embedded into a signature and its corresponding signing key is associated with a set of attributes that satisfy the policy.

The landscape of ABS schemes can be categorized by their underlying primitives. There are schemes using classical primitives based on (1) RSA [26, 27], (2) discrete-logarithms with bilinear pairings [2, 21, 28, 29], and (3) discrete-logarithms without pairings [26, 30]. There are also schemes using Post-Quantum (PQ) primitives, such as (1) lattices-based [5, 25, 31, 32], (2) multivariate-based [33], or (3) code-based [34] schemes. These schemes are less efficient when compared to classical ones, but do provide security against quantum attacks. Due to concerns about the maturity of PQ cryptography, it is unlikely that classical ABS schemes will be replaced by PQ ABS schemes in the near future. Instead, many hybrid solutions have been proposed to facilitate a smooth transition from classical cryptography to PQ cryptography [35]. In these hybrid solutions, a valid signature-message pair consists of one message and two signatures: one classical signature and one PQ signature. This indicates that both PQ and classical ABS schemes continue to be attractive research areas. Our literature review in Section 2 and the full

version of this paper [36] shows the landscape of the ABS research and identifies that pairing-based ABS schemes have a better balance between performance and other desirable features than other categories. This paper thus focuses on the designing practical ABS schemes based on pairings.

An ABS scheme aims to satisfy three primary goals: expressiveness, security, and efficiency. An *expressive* ABS scheme must support flexible access policies that can be at least described as Boolean formula (AND, OR gates). A *secure* ABS scheme should provide at least two security properties: (1) Unforgeability: without attributes satisfying the policy, one cannot create a valid signature, and (2) Anonymity: a valid signature does not reveal the signer's information except that they have attributes satisfying the policy. The *efficiency* goal aims to minimize communication and computational overhead. The rationale for the design of a practical ABS scheme is therefore to find an optimal balance between these goals.

There are several methods for defining access policies, such as Monotone Span Programs (MSP), threshold predicates [37], circuits [38], and Turing Machines (TM) [39], each with unique features that will be reviewed in the related work section. Among these, MSPs use a linear algebraic model capable of representing any monotonic access structure using AND, OR, and threshold gates without size restrictions. This makes them expressive enough for most real-world applications and enables fine-grained control over a signer's attributes. Moreover, MSP-based policies have attracted substantial research interest due to their strong balance between expressiveness and implementability. Thus, we focus our research on pairing-based ABS schemes based on MSP policies.

However, existing pairing-based ABS schemes that support MSP policies suffer from significant efficiency bottlenecks. These limitations stem from one or more of the following factors: (1) the running time of key generation or signing grows superlinearly with the number of attributes; (2) the number of pairings in verification scales linearly with the number of attributes; and (3) the constructions rely on inefficient and impractical Type-II pairings. Furthermore, to the best of our knowledge, there has been no implementation of these schemes, leaving their concrete performance unclear. These drawbacks severely hinder the practicality of ABS schemes in real-world applications. Therefore, our question is:

Can we design faster MSP-based ABS schemes than existing ones?

1.1 Our contributions

We begin by comprehensively reviewing ABS research with the goal of identifying existing gaps and challenges in both KP-ABS and SP-ABS schemes. The outcomes of this review are presented in Section 2 and further detailed in [36].

Building on the insights gained from this analysis, our work primarily focuses on the design, security analysis, and implementation of efficient pairing-based ABS schemes that support flexible Monotone Span Program (MSP) policies. Our main contributions can be summarized as follows:

- *Synchronized syntax and security model.* Similar to the ABE generalization [42], we introduce a general syntax and security model that can synchronize both KP-ABS and SP-ABS. The model captures the well-studied security properties, unforgeability and anonymity, and facilitates our design and security analysis.
- *Fast KP-ABS and SP-ABS schemes.* We propose new pairing-based KP-ABS and SP-ABS schemes using MSP policies. Our schemes support the following highly desirable features:
  - *Arbitrary attributes*: No restrictions on the size or type of attributes and any arbitrary string can act as an attribute.
  - *Large universe*[1]: Attributes are not required to be fixed at the setup stage and hence the attribute space is unbounded. The size of master public key remains constant.
  - *Adaptive unforgeability*: An adversary cannot forge a signature even if it is allowed to obtain signatures for its chosen attributes or messages at any time.
  - *Type-III pairing*: Two input groups to the pairing operation are distinct and there are no efficiently computable homomorphisms between them. It has been highly recommended due to their security and performance [44].
  - *Fast algorithms*: key generation, signing and verification time scale linearly with the number of attributes, while the number of pairing operations in verification remains constant (two), independent of the number of attributes.

  We provide a property-wise comparison between our schemes and the state-of-the-art pairing-based ABS schemes supporting MSP policies in Table 1. Our schemes are the only ones supporting all the listed properties.
- *Security analysis.* Following our security models, we prove that our ABS schemes satisfy adaptive unforgeability and anonymity under standard assumptions in the random oracle model.
- *Implementation and evaluation.* We have implemented our KP-ABS and SP-ABS schemes and evaluated their performance. This is the first implementation in the MSP-based ABS fields available in the public domain. The detail given in Section 6 shows that our schemes achieve the overall best running time in the field. We compare with the state-of-the-art KP-ABS scheme [21] and SP-ABS scheme [28]. When the attribute number is set to 100 in the policy and attribute set: the key generation of our KP-ABS runs 0.16s,

[1]The definition here was originally proposed for ABE [43] but it suits ABS as well.

Table 1: Comparison of selected pairing-based ABS schemes with MSP policies

Schemes	Large universe	Arbitrary attributes	Adaptive unforgeability	Type-III pairings	Implementation	**Key generation*** exponentiation	**Signing*** exponentiation	**Signing*** pairing	**Verification*** exponentiation	**Verification*** pairing
KP-ABS										
RD16 [21] Section 3	×	×	×	×	×	$\mathcal{O}(n_1\lvert\mathcal{U}_s\rvert)$	$\mathcal{O}(\lvert I\rvert^2)$	-	$\mathcal{O}(l)$	3
RD16 [21] Section 5	✓*	×	×	×	×	$\mathcal{O}(nn_1)$	$\mathcal{O}(n\lvert I\rvert)$	-	$\mathcal{O}(n+l)$	3
Ours [Fig. 1]	✓	✓	✓	✓	✓	$\mathcal{O}(n_1)$	$\mathcal{O}(\lvert I\rvert)$	-	$\mathcal{O}(\lvert I\rvert)$	2
SP-ABS										
MPR10 [40] Instantiation 1	✓	×	†	✓	×	$\mathcal{O}(\lvert\mathcal{S}\rvert)$	$\mathcal{O}(n_1n_2)$	$\mathcal{O}(n_1)$	$\mathcal{O}(n_1n_2)$	$\mathcal{O}(n_1n_2)$
MPR10 [40] Instantiation 2	×	×	†	×	×	$\mathcal{O}(\lvert\mathcal{S}\rvert)$	$\mathcal{O}(n_1n_2)$	$\mathcal{O}(n_1)$	$\mathcal{O}(n_1n_2)$	$\mathcal{O}(n_1n_2)$
WZFY10 [41]	×	✓	✓	×	×	$\mathcal{O}(\lvert\mathcal{S}\rvert)$	$\mathcal{O}(n_1)$	1	$\mathcal{O}(n_1)$	$\mathcal{O}(n_1)$
KCGD14 [28]	✓	×	†	✓	×	$\mathcal{O}(\lvert\mathcal{S}\rvert)$	$\mathcal{O}(n_1n_2)$	$\mathcal{O}(n_1)$	$\mathcal{O}(n_1n_2)$	$\mathcal{O}(n_1n_2)$
Ours [Fig. 2]	✓	✓	✓	✓	✓	$\mathcal{O}(\lvert\mathcal{S}\rvert)$	$\mathcal{O}(\lvert I\rvert)$	-	$\mathcal{O}(n_1)$	2

$\star$: $\mathcal{U}_s$: the attribute universe; n: the maximum bound on the size of attribute set; n_1 (or n_2): the number of rows (or columns) in the MSP matrix; I (or $\mathcal{S}$): the set of attributes; l: the length of hash output. For the computational workload, we consider exponentiations in the groups and pairing operations. We do not include the operations of $\mathsf{H}_1 : \{0,1\}^* \to \mathbb{G}_1$ that is a conversion function mapping an integer to a point (I2P), since they can be performed offline and reused.
$*$: Claimed large universe, but parameters are setup with an upbound attribute size, not satisfying our definition.
†: The security of this scheme has not been proved.

which is nearly 11 times faster than [21] (1.74s); the signing algorithm of our KP-ABS runs 0.1s, which is 72 times faster than [21] (7.2s); the verification algorithm of our KP-ABS runs (0.13s) a little faster than [21] (0.14s). For same attribute number, the key generation of our SP-ABS runs 0.082s, which is very close to [28] (0.078s); the signing algorithm of our SP-ABS runs 0.26s, which is 43 times faster than [28] (11.28s); the verification algorithm of our SP-ABS runs 0.21s, which is 294 times faster than [28] (61.7s).

1.2 Technical overview and design rationale

Notation. The following notation will be used throughout the paper. The set $\{1, \ldots, n\}$ is denoted as $[n]$. For a prime p, $\mathbb{Z}_p$ denotes the set $\{0, \ldots, p-1\}$ and $\mathbb{Z}_p^*$ is $\{1, ..., p-1\}$. λ denotes the security parameter. For a set S, $s \leftarrow\$ S$ means that s is sampled uniformly at random from S. A vector $\mathbf{v}$ is treated as a column vector and v_k denotes the k-th element of $\mathbf{v}$. For a matrix $\mathbf{M}$, $\mathbf{M}_i$ denotes its i-th row. We use $\mathbf{M}^{\mathrm{T}}$ for the transpose of $\mathbf{M}$. $\|$ denotes concatenation. A probabilistic algorithm is called probabilistic polynomial time (PPT) if its running time is bounded by some polynomial in the length of its input. Other parameters used in our proposed schemes are polynomials of λ, and a negligible function is also associated with λ. We use Type-III pairings [44], $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, where $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ are three cyclic groups with the same order p, and (g_1, g_2) are generators of $(\mathbb{G}_1, \mathbb{G}_2)$, respectively.
Access structure. To create a signature on a message, a signer uses a set of attributes $\mathcal{S}$ satisfying an access policy $\mathcal{P}$, denoted by $\mathsf{P}(\mathcal{P}, \mathcal{S}) = 1$. In our ABS schemes the policies are first written as monotone Boolean formulae[2], consisting of AND and OR gates, where each input is associated with an attribute. The space of all attributes is referred to as an attribute universe, denoted by $\mathcal{U}$. A set of attributes $\mathcal{S} \subseteq \mathcal{U}$ satisfying a Boolean formula means all inputs of the formula that map to an attribute in $\mathcal{S}$ are set to be true and the others to be false. When used in our schemes the monotone Boolean formulae are converted to *Monotone Span Programs* (MSP) [45].

An MSP access structure for a policy is denoted by $(\mathbf{M}, \pi, \{\pi(i)\}_{i\in[n_1]})$, where $\mathbf{M}$ is an $n_1 \times n_2$ matrix with $n_1, n_2 \in \mathbb{Z}_p$ and π is a general function mapping n_1 attributes to $\mathcal{U}$, $\pi : [n_1] \to \mathcal{U}$, and $\{\pi(i)\}_{i\in[n_1]}$ are the corresponding attribute values. Lewko and Waters [46] describe a simple and efficient method to convert any (monotone) Boolean formula F into an MSP policy $(\mathbf{M}, \pi, \{\pi(i)\}_{i\in[n_1]})$, where every row of $\mathbf{M}$ corresponds to an input of F and the number of columns n_2 is one more than the number of AND gates in F. Furthermore, each element in $\mathbf{M}$ is from $\{0, 1, -1\}$. Let $\mathcal{S} = \{u_i\}_{i\in[m]} \subseteq \mathcal{U}$ be a set of m attributes and $I = \{i \mid i \in \{1, \ldots, n_1\}, \pi(i) \in \mathcal{S}\}$ be the set of rows in $\mathbf{M}$ that belong to $\mathcal{S}$. $(\mathbf{M}, \pi, \{\pi(i)\}_{i\in[n_1]})$ accepts $\mathcal{S}$ if there exists a linear combination of rows in I that gives $(1, 0, ..., 0)$. This means, there exist constants $\gamma_i \in \mathbb{Z}_p$ for $i \in I$ such that $\sum_{i\in I} \gamma_i \mathbf{M}_i = (1, 0, \ldots, 0)$. These constants can be computed in polynomial time associated with the size of $\mathbf{M}$. It is worth noting that if the method from [46] is applied to Boolean formulae, then it is always possible to pick coefficients that are either 0 or 1 for resulting MSPs, irrespective of the set $\mathcal{S}$.

An aside on the underlying use of an MSP: given a secret s and $(v_1, \ldots, v_{n_2-1})$ values chosen at random from $\mathbb{Z}_p$, form a vector $\mathbf{v} = (s, v_1, ..., v_{n_2-1})^T$. For each row in the MSP matrix calculate $R_i = M_i \cdot \mathbf{v}$. If we have a set of attributes s.t. $\sum_{i\in I} \gamma_i M_i = (1, 0, \ldots, 0)$ then

$$\sum_{i\in I} \gamma_i R_i = (1, 0, ..., 0) \cdot \mathbf{v} = s.$$

This technique is used in our ABS schemes to guarantee that a signer's attributes satisfy a given policy.
Design rationale. In our ABS schemes, an Attribute Authority (AA)'s master secret key is $\mathsf{msk} = \alpha \in \mathbb{Z}_p^*$ and the corresponding master public key mpk includes $X = e(g_1, g_2)^\alpha$. From the AA, a signer Alice receives an attribute key associated with her attributes. In KP-ABS, AA embeds a policy into

[2] For a monotone Boolean formula, an authorized user who acquires extra attributes does not lose any privileges.

Alice's attribute key. Given a signature signed under this key, a verifier Bob can see which attributes used for signing but not the policy. While in SP-ABS, AA provides an attribute key to Alice without embedding any policy. Instead, Alice embeds a policy into her signature, from this signature Bob can see the policy but not the attributes.

In both ABS schemes, an attribute-based signature consists of an attribute commitment and a Non-Interactive Zero-Knowledge (NIZK) proof of this commitment. The performance of an ABS scheme is largely dependent on the constructions of the commitment and proof. In our design, we modify the key structures from the FABEO Attribute-Based Encryption (ABE) scheme [47], and generate *a batch attribute commitment*. As a result, the size of the batch commitment is independent of the size of attributes or policy. Let an attribute commitment be denoted by com, and its NIZK proof be denoted by proof. Our batch attribute commitment includes only three elements $\mathsf{com} = (A,B,C)$ with $(A,B) \in \mathbb{G}_1$ and $C \in \mathbb{G}_2$. To prove this commitment, we use a Schnorr-type signature [48] with multiple committed secrets as an instance of proof, whose length is linear in the number of attributes, but the computational cost is low. Therefore, we can make both signing and verification operations fast.

KP-ABS design. In our KP-ABS scheme, Alice's policy-embedded attribute secret key sk consists of

$$\mathsf{sk}_1 = g_2^r,\ \forall i \in [n_1]\ \mathsf{sk}_{2,i} = g_1^{\mathbf{M}_i \cdot ((\alpha+r)||\mathbf{v})^{\mathrm{T}}} \cdot \mathsf{H}_1(\pi(i))^r,$$

where $\mathsf{H}_1 : \{0,1\}^* \to \mathbb{G}_1$, $r \in \mathbb{Z}_p$ is a randomn value, and $\mathbf{v} \in \mathbb{Z}_p^{n_2-1}$ is a random vector. To sign a message msg, Alice demonstrates that she has a sufficient set of attributes $\{\pi(i)\}_{i\in I}$ satisfying the policy embedded in her key. She first finds a vector $\{\gamma_i\}_{i\in I}$ such that $\sum_{i\in I}\gamma_i\mathbf{M}_i = (1,0,\ldots,0)$, and then generates a batch commitment for her attributes. The commitment $\mathsf{com} = (A,\ B,\ C)$ satisfies the condition

$$e(A,g_2)/e(B,C) = X^{\beta} = e(g_1,g_2)^{\alpha\cdot\beta}, \tag{1}$$

where β is a random mask so com does not reveal the key or policy. To simplify our discussion, we first ignore the random mask by letting $\beta = 1$, so $e(A,g_2)/e(B,C) = e(g_1,g_2)^{\alpha}$.

Because Alice is neither given g_1^{α} nor g_2^{α}, to split $e(g_1,g_2)^{\alpha}$ into A, B, C and g_2, Alice must use her sk through

$$\sum_{i\in I}\gamma_i\mathbf{M}_i \cdot ((\alpha+r)||\mathbf{v})^{\mathrm{T}} = \alpha + r. \tag{2}$$

To do so, she computes $A = \prod_{i\in I}(\mathsf{sk}_{2,i})^{\gamma_i} = g_1^{(\alpha+r)} \cdot \prod_{i\in I}\mathsf{H}_1(\pi(i))^{r\cdot\gamma_i}$. In order to get rid of $g_1^r \cdot \prod_{i\in I}\mathsf{H}_1(\pi(i))^{r\cdot\gamma_i}$, Alice lets it be B^r, where $B = g_1 \cdot \prod_{i\in I}\mathsf{H}_1(\pi(i))^{\gamma_i}$ can be computed by Alice.

Alice is not given r but she is given $\mathsf{sk}_1 = g_2^r$. To complete the commitment (A,B,C), she makes $C = \mathsf{sk}_1 = g_2^r$; therefore, $e(A,g_2)/e(B,C) = e(g_1,g_2)^{\alpha}$ holds.

The next step is to produce an NIZK proof (the syntax can be found in [36]) of the commitment. Now, we let Alice add the random masks as follows: choose two random values $k,t \in \mathbb{Z}_p$, let $\beta = k\cdot t$, $B := B^k$ and $\delta_i = \gamma_i \cdot k$, and get

$$A = \prod_{i\in I}(\mathsf{sk}_{2,i})^{\gamma_i\cdot k\cdot t},\ B = g_1^k \cdot \prod_{i\in I}\mathsf{H}_1(\pi(i))^{\gamma_i\cdot k},\ C = \mathsf{sk}_1^t.$$

In the verification process, Bob computes Equation 1 to obtain $Y = e(A,g_2)/e(B,C)$. The NIZK proof is a Schnorr-type signature proof of knowledge, SPK, which proves the structures of Y and B having three witnesses β, k and $(\delta_i)_{i\in I}$:

$$\mathsf{proof} = \mathsf{SPK}\{(\beta,k,(\delta_i)_{i\in I}) | Y = X^{\beta} \wedge B = g_1^k \cdot \prod_{i\in I}\mathsf{H}_1(\pi(i))^{\delta_i}\}(\mathsf{msg}). \tag{3}$$

From proof, the verifier Bob is convinced that Alice has correctly used her attribute set $\{\pi(i)\}_{i\in I}$. If this attribute set satisfies multiple policies, Bob cannot tell which one, so our scheme supports policy anonymity.

SP-ABS design. In our SP-ABS scheme, Alice's attribute secret key sk consists of

$$\mathsf{sk}_1 = g_1^{\alpha}\cdot g_3^r,\ \forall u \in \mathcal{S}\ \mathsf{sk}_{2,u} = \mathsf{H}_1(u)^r,\ \mathsf{sk}_3 = g_2^r,$$

where $u \in \mathcal{S}$ is a set of attributes and $g_3 \in \mathbb{G}_1$ is a public parameter. To sign a message msg, Alice demonstrates that she has a sufficient set of attributes $\{\pi(i)\}_{i\in I}$ satisfying the policy $(\mathbf{M},\pi)$, which is embedded in the signature. She first finds a vector $\{\gamma_i\}_{i\in I}$ such that $\sum_{i\in I}\gamma_i\mathbf{M}_i = (1,0,\ldots,0)$, meanwhile she generates a vector $\mathbf{a} = \{a_i\}_{i\in[n_2]}$ from $\mathbf{M}$ and uses them to embed the policy in a similar way as in the KP-ABS key, i.e., to show

$$\sum_{i\in I}\gamma_i\mathbf{M}_i \cdot (\mathbf{a})^{\mathrm{T}} = a_1. \tag{4}$$

She generates a batch commitment for her attributes $\mathsf{com} = (A,\ B,\ C)$,

$$A = \prod_{i\in I}(\mathsf{sk}_1^{\mathbf{M}_i\cdot(\mathbf{a})^{\mathrm{T}}} \cdot \mathsf{sk}_{2,\pi(i)})^{\gamma_i\cdot k\cdot t},$$

$$B = \prod_{i\in I}(g_3^{\mathbf{M}_i\cdot(\mathbf{a})^{\mathrm{T}}} \cdot \mathsf{H}_1(\pi(i))^{\gamma_i\cdot k},\ C = \mathsf{sk}_3^t,$$

where $k,t \in \mathbb{Z}_p$ are random masks. com satisfies the equation

$$e(A,g_2)/e(B,C) = X^{a_1\cdot\beta} = e(g_1,g_2)^{\alpha\cdot a_1\cdot\beta}, \tag{5}$$

where a_1 is used to enforce the policy and $\beta = k\cdot t$ is a random mask so com does not reveal the key or attributes. In the verification process, Bob computes Equation 5 to obtain $Y = e(A,g_2)/e(B,C)$.

For the same argument as we did for our KP-ABS scheme, the structure of the batch attribute commitment $\mathsf{com} = (A,B,C)$ is proved using a signature-based proof of knowledge SPK with two witnesses, β and $(\delta_i)_{i\in[n_1]}$:

$$\mathsf{proof} = \mathsf{SPK}\{(\beta,\{\delta_i\}_{i\in[n_1]}) |$$

$$Y = (X^{a_1})^{\beta} \wedge B = \prod_{i \in [n_1]} (g_3^{\mathbf{M}_i \cdot (\mathbf{a})^{\mathrm{T}}} \cdot \mathsf{H}_1(\pi(i))^{\delta_i}\}(\mathsf{msg}). \quad (6)$$

In the signing process, Alice uses the attribute set $\{\pi(i)\}_{i \in I}$, but in the verification process, Bob uses the attribute set $\{\pi(i)\}_{i \in [n_1]}$ to verify proof. To achieve this, we define $\forall i \in [n_1] \setminus I \ \gamma_i = 0$. As the values γ_i are hidden to Bob, our SP-ABS scheme supports attribute anonymity, i.e., Bob is convinced that Alice has correctly used a set of attributes satisfying a given policy. If the policy involved multiple attribute sets, Bob cannot tell which one.

If Alice does not have a sufficient set of attributes, can she not do what we have discussed here but anything else to create a valid batch commitment and its corresponding proof? In the security analysis of our KP-ABS scheme in Sections 5, and of our SP-ABS scheme in [36] , we will prove that if Alice can generate a signature without using her attribute key, she can be used to solve the Computational Bilinear Diffie-Hellman (CBDH) problem, which contradicts to the CBDH assumption defined in Sections 5.

2 A Summary of Related Work

This section provides a summary of related work on ABS. Due to page limitations, a detailed literature review will be presented in the full version of this paper [36].

In 2008, Maji *et al.* [23] introduced the notion of ABS and proposed a concrete scheme. They also proposed two security properties, *unforgeability* and *anonymity*. Since then, ABS has garnered significant research attention. An important part of ABS schemes is the policy. So far, several types of access policies have been used for ABS schemes, including MSP, non-MSP, threshold, circuits, Turing Machines (TM), Nondeterministic Finite Automata (NFA), Range of Inner Product (RIP), and Arithmetic Branching Programs (ABP). MSP uses a linear algebra model to compute a monotone access structure [2, 23]; non-MSP is MSP with not-gate [49]; a threshold policy ensures a user must possess a certain minimum number of specific attributes [50]; a circuit policy presents the policy in arbitrary Boolean circuits [25, 38, 39, 51, 52]; a TM policy presents the policy with any computable function [22, 53], and NFA is a subclass of TMs called read-only right-moving TMs [39]; a RIP policy lets a signature/secret key associated with a range of inner product [54, 55]; and ABP checks attributes based on arithmetic operations [56].

In general, based on the Attribute Authority (AA) settings, ABS schemes are divided as centralized ABS [1,2,2,19,22,23, 28,37,38,51,54,55,59,79,85–89] and decentralized ABS [90–99]. In addition, some ABS schemes outsource computation to other parties [98, 100–104]. Our focus in this paper is centralized ABS. The comparisons for selected important features of the classical ABS schemes are shown in Table 2, which indicate that our ABS schemes have the advantage of supporting the desirable features. The details for existing research on KP-ABS and SP-ABS will be introduced in [36].

Table 2: Comparisons for features of ABS schemes

Scheme[a]	Policy	Large universe[b]	Arbitrary attribute	Type-III pairings[c]	Open source	Adaptive unforge.[d]	Anon.[e]
KP-ABS							
[20](08)	Threshold	✓*	✓	×	×	×	✓
[20](08)	Threshold	✓*	×	×	×	×	✓
[1](09)	Threshold	×	×	×	×	×	✓
[19](10)	Threshold	×	✓	×	✓	×	✓
[19](10)	Threshold	×	×	×	✓	×	✓
[57](11)	MSP	×	✓	×	×	×	✓
[29] (12)	Threshold	×	×	×	×	✓	✓
[29] (12)	Threshold	✓*	×	×	×	✓	✓
[29] (12)	Threshold	×	×	×	×	✓	✓
[37](12)	Threshold	✓*	×	×	×	×	✓
[51](14)	Circuits	×	×	×	×	×	✓
[21](16)	MSP	×	×	×	×	×	✓
[21](16)	MSP	✓*	×	×	×	×	✓
[58](17)	MSP	×	×	×	×	†	†
[39](18)	NFA/TM	✓	✓	✓	×	✓	✓
[22](23)	TM	✓	✓	†	×	×	✓
[55](23)	RIP	×	×	×	×	†	†
Ours	MSP	✓	✓	✓	✓	✓	✓
SP-ABS							
[40](10)	MSP	✓	×	✓	×	†	†
[40](10)	MSP	×	×	✓	×	†	†
[41](10)	MSP	×	✓	×	×	✓	✓
[59](11)	Non-MSP	×	×	×	×	✓	✓
[60](11)	MSP	×	×	×	×	✓	†
[61](11)	Threshold	✓	✓	×	×	✓	✓
[61](11)	MSP	×	✓	×	×	†	†
[37](12)	Threshold	×	×	×	×	×	✓
[62](12)	MSP	×	×	×	×	†	×
[63](12)	MSP	×	✓	×	×	✓	✓
[64](12)	MSP	×	×	×	×	†	×
[65](13)	Threshold	×	×	×	×	†	†
[26](14)	MSP	×	✓	†	×	†	†
[28](14)	MSP	✓	×	✓	×	†	†
[66](14)	Threshold	×	✓	×	×	×	✓
[67](14)	Threshold	×	✓	×	×	✓	✓
[68](14)	MSP	×	×	×	×	×	✓
[3](14)	Threshold	×	✓	×	×	×	✓
[24](14)	Non-MSP	×	×	×	×	✓	✓
[69](14)	MSP	✓	✓	×	×	×	✓
[70](14)	Threshold	×	×	×	×	✓	✓
[71](15)	Threshold	×	×	×	✓	✓	✓
[72](15)	Threshold	×	×	×	×	✓	✓
[73](15)	Threshold	×	✓	×	×	✓	✓
[74](16)	MSP	✓	×	✓	×	✓	✓
[38](16)	Circuits	✓	×	✓	×	✓	✓
[75](16)	Threshold	×	×	×	×	×	✓
[76](17)	MSP	×	×	✓	×	✓	✓
[76](17)	Threshold	×	×	✓	×	✓	✓
[77](17)	MSP	×	×	×	×	✓	✓
[56](19)	ABP	×	×	✓	×	✓	✓
[30](19)	MSP	×	×	†	×	✓	✓
[78](19)	MSP	×	×	×	×	✓	✓
[79](21)	MSP	×	×	×	×	✓	✓
[54](22)	RIP	×	×	×	×	✓	✓
[80](22)	MSP	×	✓	×	×	×	✓
[52](22)	Circuits	✓	×	✓	×	†	†
[81](22)	Threshold	×	×	✓	✓	†	†
[82](23)	Threshold	×	×	×	✓	×	✓
Ours	MSP	✓	✓	✓	✓	✓	✓

a: The number in brackets denotes the year, such as "(08)" means 2008.
b: ✓ (or ×) indicates large (or small) universe, and ✓* claims large universe but it does not satisfy our large universe definition.
c: ✓ uses Type-III pairings, × uses Type-I or Type-II pairings, and † means that the scheme does not use pairings. Type-I pairings are showing serious security issues and Type-II pairings are inefficient [44, 83, 84].
d: ✓ (or ×) indicates adaptive (or selective) unforgeability, † indicates no proof.
e: ✓ indicates that anonymity has been proved, × indicates that anonymity is not held, and † indicates no proof of anonymity.

3 General ABS Model and Security

An ABS scheme involves three entities: (1) an attribute authority (AA) that generates attribute keys for a group of users, (2) a group of signers who receive the attribute keys from AA and use them to sign messages on some specified predicates, and (3) a verifier that checks the validity of message-signature pairs.

ABS algorithms. We provide a general syntax for both KP-ABS and SP-ABS schemes. An ABS scheme defined over attribute universe $\mathcal{U}$ and message space $\mathcal{M}$ consists of the following algorithms:

- $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$: The setup algorithm takes as input a security parameter λ, outputs a master public key mpk and a master secret key msk.
- $\mathsf{sk}_x \leftarrow \mathsf{KeyGen}(\mathsf{mpk}, \mathsf{msk}, x)$: The key generation algorithm takes as input a master key pair $(\mathsf{mpk}, \mathsf{msk})$, and a description x, outputs an attribute secret key sk_x.
- $\sigma \leftarrow \mathsf{Sign}(\mathsf{mpk}, \mathsf{sk}_x, y, \mathsf{msg})$: The signing algorithm takes as input a master public key mpk, a secret key sk_x, a description y, a message msg, outputs a signature σ.
- $0/1 \leftarrow \mathsf{Verify}(\mathsf{mpk}, \sigma, y, \mathsf{msg})$: The verification algorithm takes as input a master public key mpk, a signature σ, a description y, and a message msg, outputs 1 for accept or 0 for reject.

Remark 1. By *descriptions* x and y used as input in algorithms KeyGen, Sign and Verify we mean either an attribute set $\mathcal{S} \subseteq \mathcal{U}$ or an access policy $\mathcal{P}$. For KP-ABS, $x = \mathcal{P}$, $y = \mathcal{S}$; while for SP-ABS, $x = \mathcal{S}$, $y = \mathcal{P}$. We define $\mathsf{P}(x, y) = 1$ as $\mathcal{S}$ satisfies $\mathcal{P}$ and $\mathsf{P}(x, y) = 0$ as $\mathcal{S}$ does not satisfy $\mathcal{P}$, no matter which is associated with x or y.

Correctness. For any x, y with $\mathsf{P}(x, y) = 1$ and $\mathsf{msg} \in \mathcal{M}$, we require $\Pr[\mathsf{Verify}(\mathsf{mpk}, \sigma, y, \mathsf{msg}) = 1 : (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda), \mathsf{sk}_x \leftarrow \mathsf{KeyGen}(\mathsf{mpk}, \mathsf{msk}, x), \sigma \leftarrow \mathsf{Sign}(\mathsf{mpk}, \mathsf{sk}_x, y, \mathsf{msg})] = 1$.

Unforgeability. The standard notion of security for ABS is existential unforgeability under adaptive chosen message attack (EUF-CMA). This notion is defined using the following game between an adversary $\mathcal{A}$ and a challenger algorithm $\mathcal{B}$:

- **Setup.** $\mathcal{B}$ runs $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$, sends mpk to $\mathcal{A}$ and keeps msk secret.
- **Query.** $\mathcal{A}$ issues queries to a key generation oracle and a signing oracle for polynomial times:
  - Key generation oracle: Given a description x, the oracle returns $\mathsf{sk}_x \leftarrow \mathsf{KeyGen}(\mathsf{mpk}, \mathsf{msk}, x)$ to $\mathcal{A}$.
  - Signing oracle: Given descriptions (x, y) and a message msg, the oracle generates $\mathsf{sk}_x \leftarrow \mathsf{KeyGen}(\mathsf{mpk}, \mathsf{msk}, x)$, and returns $\sigma \leftarrow \mathsf{Sign}(\mathsf{mpk}, \mathsf{sk}_x, y, \mathsf{msg})$ to $\mathcal{A}$.
- **Forgery.** $\mathcal{A}$ outputs a tuple $(\mathsf{msg}^*, y^*, \sigma^*)$. The adversary wins the game if the following conditions hold:
  - $\mathsf{Verify}(\mathsf{mpk}, \sigma^*, y^*, \mathsf{msg}^*) = 1$.
  - $\mathcal{A}$ has not queried any description x such that $\mathsf{P}(x, y^*) = 1$ to the key generation oracle.
  - $\mathcal{A}$ has not queried the signing oracle with following types of inputs: (1) $(\cdot, y^*, \mathsf{msg}^*)$ for any description x, or (2) $(x, \cdot, \cdot)$ for any (y, msg) pair if $\mathsf{P}(x, y^*) = 1$.

Definition 1 *An ABS scheme is EUF-CMA if the advantage function refers to the above game*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{EUF-CMA}}(\lambda) = \Pr[\mathcal{A} \textit{ wins}]$$

is negligible in parameter λ for any PPT adversary $\mathcal{A}$.

Remark 2. There are several weaker notions of unforgeability [29], which includes (1) Existential unforgeability under selective chosen message attack (EUF-sCMA), in which the adversary commits to a chosen message at the Setup phase and the forgery must be produced on that message. (2) Selective-attribute unforgeability, in which the adversary commits to a chosen description y at the Setup phase and the forgery must be produced on y. It can be considered either under adaptive chosen message attack (s-EUF-CMA), or selective chosen message attack (s-EUF-sCMA). In this paper, we first prove the unforgeability of our ABS schemes under the s-EUF-CMA model, and then transform it into EUF-CMA by using the random oracle model.

Anonymity. An ABS scheme is anonymous if the distribution of signatures is independent of secret keys used to produce them, i.e., that signatures do not disclose any information about attributes or policies embedded into those keys. Specifically, one cannot deduce any information about the description x held by the signer from a signature on a description y, other than the fact that $\mathsf{P}(x, y) = 1$. The anonymity requirement is formally defined using the following game between an adversary $\mathcal{A}$ and a challenger algorithm $\mathcal{B}$:

- **Setup.** Same as the one defined in the EUF-CMA model.
- **Phase 1.** Same as the Query phase defined in EUF-CMA.
- **Challenge.** $\mathcal{A}$ outputs $(x_1^*, x_2^*, y^*, \mathsf{msg}^*)$ that satisfies $\mathsf{P}(x_1^*, y^*) = \mathsf{P}(x_2^*, y^*) = 1$. Then $\mathcal{B}$ chooses a bit $b \in \{0, 1\}$, generates a secret key $\mathsf{sk}_{x_b^*} \leftarrow \mathsf{KeyGen}(\mathsf{mpk}, \mathsf{msk}, x_b^*)$, and signature $\sigma_b^* \leftarrow \mathsf{Sign}(\mathsf{mpk}, \mathsf{sk}_{x_b^*}, y^*, \mathsf{msg}^*)$. Finally, $\mathcal{B}$ returns σ_b^* to $\mathcal{A}$.
- **Phase 2.** Same as Phase 1 defined above.
- **Guess.** $\mathcal{A}$ outputs $b' \in \{0, 1\}$. It wins if $b' = b$.

Definition 2 *An ABS scheme is anonymous if the advantage function refers to the above game*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Anon}}(\lambda) = \left|\Pr[b' = b] - \tfrac{1}{2}\right|$$

is negligible in parameter λ for any PPT adversary $\mathcal{A}$.

Remark 3. In the anonymity game, the adversary $\mathcal{A}$ is allowed to query the challenge descriptions x_1^* and x_2^* in the key generation oracle, and query the challenge tuple $(x_1^*, x_2^*, y^*, \mathsf{msg}^*)$ in the signing oracle. The rationale is that oracles always respond with a different *sk* or σ every time for the same inputs; hence, the challenge σ_b^* has up to negligible probability equal to any σ that has been queried from the signing oracle for the challenge tuple before. Thus, anonymity should be guaranteed even if $\mathcal{A}$ is given this capability.

4 Our ABS schemes

In this section, we provide a detailed specification of our proposed KP-ABS and SP-ABS schemes. We first define the bilinear maps and the algorithm for generating public parameters used in both schemes: Let GroupGen be a PPT algorithm that takes as input a security parameter 1^λ and outputs a set of group parameters $\mathsf{par} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$, where p is the prime order of $\Theta(\lambda)$ bits, $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ are cyclic groups of order p, g_1 and g_2 are the generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is an asymmetric Type-III pairing function where there exists no efficiently computable homomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$.

4.1 KP-ABS

Our Key-Policy Attribute-Based Signature (KP-ABS) scheme is presented in Fig. 1. To insert a policy into a secret key, we adopt a modified key structure from FABEO KP-ABE [47] by adding the randomness r into $\mathbf{M}_i \cdot ((\alpha + r)||\mathbf{v})^{\mathrm{T}}$ when computing $\mathsf{sk}_{2,i}$. This modification allows us to prove security of this KP-ABS scheme, which will be presented in Sections 5.

In this scheme, a KP-ABS signature $\sigma = (A,\ B,\ C,\ c,\ s_\alpha,\ s_k,\ \{s_i\}_{i\in I})$ includes a batch attribute commitment $\mathsf{com} = (A,\ B,\ C)$ and a signature-based proof of knowledge SPK, as shown in Equation 3:

$$\mathsf{proof} = \mathsf{SPK}\{(\beta, k, \{\delta_i\}_{i\in I}) | Y = X^\beta \wedge B = g_1^k \cdot \prod_{i\in I} h_i^{\delta_i}\}(\mathsf{msg}),$$

where $Y = e(A, g_2)/e(B, C)$. We use a Schnorr-type signature with multiple committed secrets as an instance of a SPK, denoted by $\mathsf{proof} = (c,\ s_\alpha,\ s_k,\ \{s_i\}_{i\in I})$.

Theorem 1 *Our KP-ABS scheme is correct.*

Proof 1 *We show that for* $x = (\mathbf{M},\ \pi,\ \{\pi(i)\}_{i\in[n_1]}), y = \{\pi(i)\}_{i\in I}$ *with* $\mathsf{P}(x,\ y) = 1$ *and* $\mathsf{msg} \in \mathcal{M}$, *then* $\Pr[\mathit{Verify}(\mathit{mpk}, \sigma, y, \mathit{msg}) = 1 : (\mathsf{mpk},\ \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda), \mathsf{sk}_x \leftarrow \mathit{KeyGen}(\mathit{mpk}, \mathit{msk}, x),\ \sigma \leftarrow \mathit{Sign}(\mathit{mpk}, \mathit{sk}_x, y, \mathit{msg})] = 1$. *Following the description of the scheme in Fig. 1, given a signature* $\sigma = (A, B, C, c, s_\alpha, s_k, \{s_i\}_{i\in I})$, *the verifier computes*

$$\begin{aligned} Y &= \frac{e(A,\ g_2)}{e(B,\ C)} \\ &= \frac{e(\prod_{i\in I}\ (g_1^{\mathbf{M}_i((\alpha+r)||\mathbf{v})^T} \cdot \mathsf{H}_1(\pi(i))^r)^{\gamma_i \cdot k \cdot t},\ g_2)}{e(g_1^k \cdot \prod_{i\in I}\ \mathsf{H}_1(\pi(i))^{\gamma_i \cdot k},\ g_2^{r\cdot t})} \\ &= \frac{e(g_1^{(\alpha+r)\cdot k\cdot t} \cdot \prod_{i\in I}\ (\mathsf{H}_1(\pi(i))^{r\cdot\gamma_i\cdot k\cdot t},\ g_2)}{e(g_1^k \cdot \prod_{i\in I}\ \mathsf{H}_1(\pi(i))^{\gamma_i\cdot k},\ g_2^{r\cdot t})} \\ &= e(g_1^{\alpha\cdot k\cdot t},\ g_2) = X^{k\cdot t} \end{aligned}$$

Then the verifier checks that $\mathsf{proof} = (c, s_\alpha, s_k, \{s_i\}_{i\in I})$ *is a Schnorr-type signature-based proof of knowledge SPK in Equation 3, proving three types of secrets:* β, k, δ_i, *where* $\beta = k \cdot t$ *and* $\delta_i = \gamma_i \cdot k$. *To verify proof, the verifier recomputes the value c by following the Verify algorithm in Figure 1 and compares it with the given c value in* σ. *For a valid Schnorr-type SPK, the comparison result must be positive.*

4.2 SP-ABS

Our Signature-Policy Attribute-Based Signature (SP-ABS) scheme is presented in Fig. 2. To insert a policy into a signature, we create a vector $\mathbf{a}$ from the policy matrix $\mathbf{M}$ with the dimensions $n_1 \times n_2$, and use this vector to test the policy. To do so, let $a_i = \mathsf{H}(i||\mathsf{H}_0(\mathbf{M}))$ for $i \in [n_2]$, and the vector $\mathbf{a} = \{a_i\}_{i\in[n_2]}$. Based on the subset $\{\pi(i)\}_{i\in I}$ under $(\mathbf{M}, \pi)$, the signer finds constants $\{\gamma_i\}_{i\in I}$. Following Equation 4, $\sum_{i\in I}\gamma_i \mathbf{M}_i(\mathbf{a})^{\mathrm{T}} = (a_1,\ 0,\ \ldots,\ 0)$. In our design, in order to achieve attribute anonymity, the signer uses a set of dummy attributes with null γ values, i.e., $\forall_{i\in[n_1]\setminus I}\ \gamma_i = 0$. In addition, we choose the base point g_3 at random; alternatively, we could also use FABEO's method by letting $g_3 = \mathsf{H}_1(|\mathcal{U}| + 1)$.

A signature $\sigma = (A, B, C, c, s_\alpha, \{s_i\}_{i\in[n_1]})$ includes a batch attribute commitment $\mathsf{com} = (A, B, C)$ and a signature-based proof of knowledge SPK, as shown in Equation 6:

$$\mathsf{proof} = \mathsf{SPK}\{(\beta, \{\delta_i\}_{i\in[n_1]}) |$$

$$Y = (X^{a_1})^\beta \wedge B = \prod_{i\in[n_1]} (g_3^{\mathbf{M}_i\cdot(\mathbf{a})^{\mathrm{T}}} \cdot \mathsf{H}_1(\pi(i)))^{\delta_i}\}(\mathsf{msg}),$$

where $Y = e(A, g_2)/e(B, C)$. We use a Schnorr-type signature with multiple committed secrets as an instance of a SPK, denoted by $\mathsf{proof} = (c, s_\alpha, \{s_i\}_{i\in[n_1]})$.

Theorem 2 *Our SP-ABS scheme is correct.*

Proof 2 *We show that for* $x = (\mathcal{S} \subseteq \mathcal{U})$, *in which an element in* $\mathcal{S}$ *is written as u in* KeyGen *or* $\pi(i)$ *in* Sign, *and* $y = (\mathbf{M}, \pi, \{\pi(i)\}_{i\in[n_1]})$ *with* $\mathsf{P}(x,\ y) = 1$ *and* $\mathsf{msg} \in \mathcal{M}$, *then* $\Pr[\mathit{Verify}(\mathit{mpk}, \sigma, y, \mathit{msg}) = 1 : (\mathsf{mpk},\ \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda), \mathsf{sk}_x \leftarrow \mathit{KeyGen}(\mathit{mpk}, \mathit{msk}, x), \sigma \leftarrow \mathit{Sign}(\mathit{mpk}, \mathit{sk}_x, y, \mathit{msg})] = 1$. *Following the description of the scheme in Fig. 2, given a signature* $\sigma = (A, B, C, c, s_\alpha, \{s_i\}_{i\in[n_1]})$, *the verifier computes*

$\underline{(\mathsf{mpk},\ \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)}$.

Run $\mathsf{GroupGen}(1^\lambda)$ to obtain the group parameters $\mathsf{par} = (p,\ \mathbb{G}_1,\ \mathbb{G}_2,\ \mathbb{G}_T,\ g_1,\ g_2,\ e)$. Pick $\alpha \leftarrow\!\$\ \mathbb{Z}_p^*$ and two hash functions $\mathsf{H}_0 : \{0,\ 1\}^* \to \mathbb{Z}_p$, $\mathsf{H}_1 : \{0,\ 1\}^* \to \mathbb{G}_1$. Output

$$\mathsf{mpk} = (\mathsf{H}_0,\ \mathsf{H}_1,\ \mathsf{par},\ X = e(g_1,\ g_2)^\alpha),\ \mathsf{msk} = \alpha.$$

$\underline{\mathsf{sk} \leftarrow \mathsf{KeyGen}(\mathsf{mpk},\ \mathsf{msk},\ x = (\mathbf{M},\ \pi,\ \{\pi(i)\}_{i\in[n_1]}))}$.

Pick $r \leftarrow\!\$\ \mathbb{Z}_p$, $\mathbf{v} \leftarrow\!\$\ \mathbb{Z}_p^{n_2-1}$. Compute

$$\begin{aligned} \mathsf{sk}_1 &= g_2^r, \\ \forall i \in [n_1]\ \mathsf{sk}_{2,i} &= g_1^{\mathbf{M}_i\cdot((\alpha+r)||\mathbf{v})^{\mathrm{T}}} \cdot \mathsf{H}_1(\pi(i))^r. \end{aligned}$$

Output $\mathsf{sk} = (\mathsf{sk}_1,\ \{\mathsf{sk}_{2,i}\}_{i\in[n_1]}, (\mathbf{M},\ \pi,\ \{\pi(i)\}_{i\in[n_1]}))$.

$\underline{\sigma \leftarrow \mathsf{Sign}(\mathsf{mpk},\ \mathsf{sk},\ y = \{\pi(i)\}_{i\in I},\ \mathsf{msg})}$.

Find $\{\gamma_i\}_{i\in I}$ s.t. $\sum_{i\in I}\gamma_i\mathbf{M}_i = (1,\ 0,\ \ldots,\ 0)$.
Pick $k, t, r_\alpha, r_k, \{r_i\}_{i\in I} \leftarrow\!\$\ \mathbb{Z}_p$. Compute

$$\begin{aligned} A &= \textstyle\prod_{i\in I}(\mathsf{sk}_{2,i})^{\gamma_i\cdot k\cdot t}, & C &= \mathsf{sk}_1^t, \\ B &= g_1^k \cdot \textstyle\prod_{i\in I}\mathsf{H}_1(\pi(i))^{\gamma_i\cdot k}, & Y &= X^{k\cdot t}, \\ W &= g_1^{r_k} \cdot \textstyle\prod_{i\in I}\mathsf{H}_1(\pi(i))^{r_i}, & Z &= X^{r_\alpha}, \\ c &= \mathsf{H}_0(A,\ B,\ C,\ Y,\ Z,\ W,\ \mathsf{msg}), \end{aligned}$$

$$s_\alpha = r_\alpha - k\cdot t\cdot c,\ s_k = r_k - k\cdot c,$$
$$\forall i \in I\ s_i = r_i - \gamma_i\cdot k\cdot c.$$

Output $\sigma = (A,\ B,\ C,\ c,\ \mathbf{s})$, where $\mathbf{s} = (s_\alpha,\ s_k,\ \{s_i\}_{i\in I})$.

$\underline{0/1 \leftarrow \mathsf{Verify}(\mathsf{mpk},\ \sigma,\ y = \{\pi(i)\}_{i\in I},\ \mathsf{msg})}$.

Compute

$$Y' = \frac{e(A,\ g_2)}{e(B,\ C)},\ Z' = X^{s_\alpha}\cdot Y'^c,\ W' = g_1^{s_k}\cdot\prod_{i\in I}\mathsf{H}_1(\pi(i))^{s_i}\cdot B^c,$$

$$c' = \mathsf{H}_0(A,\ B,\ C,\ Y',\ Z',\ W',\ \mathsf{msg}).$$

If $c' \neq c$, output 0; otherwise, output 1.

Figure 1: The construction of our KP-ABS scheme

$\underline{(\mathsf{mpk},\ \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)}$.

Run $\mathsf{GroupGen}(1^\lambda)$ to obtain $\mathsf{par} = (p,\ \mathbb{G}_1,\ \mathbb{G}_2,\ \mathbb{G}_T,\ g_1,\ g_2,\ e)$. Pick $\alpha \leftarrow\!\$\ \mathbb{Z}_p^*$, $g_3 \leftarrow\!\$\ \mathbb{G}_1$, and three hash functions $\mathsf{H}, \mathsf{H}_0 : \{0,\ 1\}^* \to \mathbb{Z}_p$, and $\mathsf{H}_1 : \{0,\ 1\}^* \to \mathbb{G}_1$. Output

$$\mathsf{mpk} = (\mathsf{H},\ \mathsf{H}_0,\ \mathsf{H}_1,\ \mathsf{par},\ g_3,\ X = e(g_1,\ g_2)^\alpha),\ \mathsf{msk} = \alpha.$$

$\underline{\mathsf{sk} \leftarrow \mathsf{KeyGen}(\mathsf{mpk},\ \mathsf{msk},\ x = \mathcal{S})}$.

Pick $r \leftarrow\!\$\ \mathbb{Z}_p$. Compute

$$\mathsf{sk}_1 = g_1^\alpha \cdot g_3^r,\ \forall u \in \mathcal{S}\ \mathsf{sk}_{2,u} = \mathsf{H}_1(u)^r,\ \mathsf{sk}_3 = g_2^r.$$

Output $\mathsf{sk} = (\mathsf{sk}_1,\ \{\mathsf{sk}_{2,u}\}_{u\in\mathcal{S}},\ \mathsf{sk}_3,\ \mathcal{S})$.

$\underline{\sigma \leftarrow \mathsf{Sign}(\mathsf{mpk},\ \mathsf{sk},\ y = (\mathbf{M},\pi,\{\pi(i)\}_{i\in[n_1]}),\ \mathsf{msg})}$.

Find $\{\gamma_i\}_{i\in I}$ s.t. $\sum_{i\in I}\gamma_i\mathbf{M}_i = (1,\ 0,\ \ldots,\ 0)$ with $\{\pi(i)\}_{i\in I} \subseteq \mathcal{S}$. Pick $k,\ t,\ r_\alpha,\ \{r_i\}_{i\in[n_1]} \leftarrow\!\$\ \mathbb{Z}_p$. Compute

$$\begin{aligned} a_i &= \mathsf{H}(i||\mathsf{H}_0(\mathbf{M})), \forall i \in [n_2], & \mathbf{a} &= \{a_i\}_{i\in[n_2]}, \\ A &= \textstyle\prod_{i\in I}(\mathsf{sk}_1^{\mathbf{M}_i\cdot(\mathbf{a})^{\mathrm{T}}} \cdot \mathsf{sk}_{2,\pi(i)})^{\gamma_i\cdot k\cdot t}, & C &= \mathsf{sk}_3^t, \\ B &= \textstyle\prod_{i\in I}(g_3^{\mathbf{M}_i\cdot(\mathbf{a})^{\mathrm{T}}} \cdot \mathsf{H}_1(\pi(i)))^{\gamma_i\cdot k}, & Y &= X^{a_1\cdot k\cdot t}, \\ W &= \textstyle\prod_{i\in[n_1]}(g_3^{\mathbf{M}_i\cdot(\mathbf{a})^{\mathrm{T}}} \cdot \mathsf{H}_1(\pi(i)))^{r_i}, & Z &= (X^{a_1})^{r_\alpha}, \\ c &= \mathsf{H}_0(A,\ B,\ C,\ Y,\ Z,\ W,\ \mathsf{msg}), \end{aligned}$$

$s_\alpha = r_\alpha - k\cdot t\cdot c,\ \forall i \in I\ s_i = r_i - \gamma_i\cdot k\cdot c,\ \forall i \in [n_1]\setminus I\ s_i = r_i.$

Output $\sigma = (A,\ B,\ C,\ c,\ \mathbf{s})$, where $\mathbf{s} = (s_\alpha,\ \{s_i\}_{i\in[n_1]})$.

$\underline{0/1 \leftarrow \mathsf{Verify}(\mathsf{mpk},\ \sigma,\ y = (\mathbf{M},\pi,\{\pi(i)\}_{i\in[n_1]}),\ \mathsf{msg})}$.

Compute

$$\forall i \in [n_2]\ a_i = \mathsf{H}(i||\mathsf{H}_0(\mathbf{M})),\ \mathbf{a} = \{a_i\}_{i\in[n_2]},$$

$$Y' = \frac{e(A,g_2)}{e(B,C)},\ Z' = (X^{a_1})^{s_\alpha}\cdot Y'^c,$$

$$W' = \prod_{i\in[n_1]}(g_3^{\mathbf{M}_i\cdot(\mathbf{a})^{\mathrm{T}}} \cdot \mathsf{H}_1(\pi(i)))^{s_i}\cdot B^c,$$

$$c' = \mathsf{H}_0(A,\ B,\ C,\ Y',\ Z',\ W',\ \mathsf{msg}).$$

If $c' \neq c$, output 0; otherwise output 1.

Figure 2: The construction of our SP-ABS scheme

$$\begin{aligned} Y &= \frac{e(A,\ g_2)}{e(B,\ C)} \\ &= \frac{e(\prod_{i\in I}\ ((g_1^\alpha\cdot g_3^r)^{\mathbf{M}_i(\mathbf{a})^T}\cdot\mathsf{H}_1(\pi(i))^r)^{\gamma_i\cdot k\cdot t},\ g_2)}{e(\prod_{i\in I}\ (g_3^{\mathbf{M}_i(\mathbf{a})^T}\cdot\mathsf{H}_1(\pi(i)))^{\gamma_i\cdot k},\ g_2^{r\cdot t})} \\ &= \frac{e(g_1^{\alpha\cdot a_1\cdot k\cdot t}\cdot\prod_{i\in I}\ (g_3^{a_1}\cdot\mathsf{H}_1(\pi(i))^{\gamma_i})^{r\cdot k\cdot t},\ g_2)}{e(\prod_{i\in I}\ (g_3^{a_1}\cdot\mathsf{H}_1(\pi(i)^{\gamma_i})^k,\ g_2^{r\cdot t})} \\ &= e(g_1^{\alpha\cdot a_1\cdot k\cdot t},\ g_2) = X^{a_1\cdot k\cdot t} \end{aligned}$$

Then the verifier checks that $\mathsf{proof} = (c, s_\alpha, \{s_i\}_{i\in I})$ *is a Schnorr-type signature-based proof of knowledge SPK in Equation 6, where* $\beta = k\cdot t$ *and* $\delta_i = \gamma_i\cdot k$. *The trick part is that in the signing process, the element B is computed with* $i \in I$, *but in the verification process, B is proved with* $i \in [n_1]$. *This difference does not stop passing the verification because* $\forall_{i\in[n_1]\setminus I}\ \gamma_i = 0$. *The verification of proof is a Schnorr-type signature-based proof of knowledge SPK in Equation 6, and its correctness is held.*

Remark 4. To achieve *large universe* and *arbitrary attributes*, we use an integer to point (I2P) conversion function H_1 that

maps any attribute string to a group element in $\mathbb{G}_1$. This eliminates the need to specify attributes during setup and allows attributes to be any type and size. This operation can be performed offline and the result can be reused over keys and signatures.

Remark 5. Attribute multi-use means that the scheme should still work even when the MSP handles complex policies, e.g., for threshold policies, the MSP matrix would need to map multiple rows to the same attribute. In FABEO ABE schemes [47], there are no interactions between the AA and the encryptors. Thus, they increase with $O(\tau)$ overhead on both the encryptor and decryptor, in which $\tau = \max_{i\in[n_1]}\rho(i)$ corresponding to maximum number of times an attribute is used in $\mathbf{M}$, and $\rho(i) = |\{z|\pi(z) = \pi(i), z \leq i\}|$ is the number of times an attribute $\pi(i)$ is used in $\mathbf{M}$. On the other hand, our ABS schemes can achieve this feature without increasing overhead by exploiting the fact that the AA always generates secret keys before the signer signs a message.

For our KP-ABS, the AA sets a policy in KeyGen stage. This means that the AA can map the same attribute from the matrix $\mathbf{M}$ with a different form if it is used multiple times. For instance, if an attribute a is used twice in $\mathbf{M}$ on row 2 and row 3, the AA can simply map $\pi(2) = a||1$ and $\pi(3) = a||2$ such that each attribute is treated uniquely in $\mathbf{M}$. Since $(\mathbf{M}, \pi)$ is known to the signer, the signer can calculate each $\mathsf{H}_1(\pi(i))$ in B and W by following the same π, and provide the verifier $\{\pi(i)\}_{i\in I}$ encoded by the same π. In the anonymity model, the adversary is given one challenge attribute set $\mathcal{S}^*$ that satisfies two policies $\mathcal{P}_0^*, \mathcal{P}_1^*$. This means that every attribute such as $a||2$ that appeared in $\mathcal{S}^*$ must appear in $\{\pi^*(i)\}_{i\in I^*}$. $\mathcal{P}_0^*, \mathcal{P}_1^*$ only differs in the part of attributes that are not in $\{\pi^*(i)\}_{i\in I^*}$. Thus, this technique does not affect anonymity.

In our SP-ABS, the AA does not know the policy at the KeyGen stage, so it can only calculate each $\mathsf{H}_1(u)$ as normal and cannot synchronize the encoding of attributes in the policy chosen by the signer. In this case, when the signer sets a policy with attributes that are used multiple times, it keeps π mapping the same attribute in the same form as in KeyGen stage, such that the matrix $\mathbf{M}$ can include multiple rows mapping to the same attribute. When the signer calculates A, B and W, some rows i in I and $[n_1]$ could be mapped to the same attribute, e.g., $\pi(i) = \pi(j)$ for $i \neq j$. For these attributes, the signer correspondingly multiplies the same $\mathsf{sk}_{2,\pi(i)}$, $\mathsf{H}_1(\pi(i))$ in $\rho(i)$ times, matching the same number of them appearing in $\mathbf{M}$. As $(\mathbf{M}, \pi, \{\pi(i)\}_{i\in[n_1]})$ is public to the verifier, the verifier can compute the $\prod_{i\in[n_1]}\mathsf{H}_1(\pi(i))$ term in the same way.

It is plausible that the above techniques can be applied to other ABS schemes, we remain this question as a future work.

5 Security analysis

We prove both our KP-ABS and SP-ABS schemes satisfy the EUF-CMA security and anonymity as defined in Sec. 3. The proofs are based on the following two mathematical problems. Due to page limitations, we only include the proofs of our KP-ABS scheme in this version and reserve the proofs of our SP-ABS scheme for the full version [36].

Computational Bilinear Diffie-Hellman (CBDH) assumption. The computational BDH problem in Type-III setting is that given a security parameter λ, $\mathsf{par} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2) \leftarrow \mathsf{GroupGen}(1^\lambda)$ and $\mathsf{D} = (g_1^\alpha, g_2^\alpha, g_1^\beta, g_2^\beta, g_1^\gamma, g_2^\gamma)$, where $\alpha, \beta, \gamma \leftarrow\$ \mathbb{Z}_p$, compute $\mathsf{T} = e(g_1, g_2)^{\alpha\beta\gamma}$. The advantage of a PPT algorithm $\mathcal{A}$ in solving this problem

$$\mathrm{Adv}_{\mathcal{A}}^{\mathrm{CBDH}}(\lambda) := \left|\Pr[\mathcal{A}(\mathsf{par}, \mathsf{D}, \mathsf{T}) = 1]\right|$$

is negligible in λ. The probability is over the random choice of the parameters and over the coin tosses of $\mathcal{A}$.

Definition 3 (CBDH assumption) The CBDH assumption holds in $\mathbb{G}_T$ if no PPT algorithm has advantage at least ε in solving the CBDH problem.

Decisional External Diffie-Hellman (DXDH) assumption [105]. The decisional XDH problem is that given a security parameter λ, $\mathsf{par} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2) \leftarrow \mathsf{GroupGen}(1^\lambda)$, $\mathsf{D} = (g_1^a, g_1^b)$, where $a, b, R \leftarrow\$ \mathbb{Z}_p$, $\mathsf{T}_0 = g_1^{ab}$ and $\mathsf{T}_1 = g_1^R$, decide which one from T_0 and T_1 is associated with D. The advantage of a PPT algorithm $\mathcal{A}$ in solving this problem

$$\mathrm{Adv}_{\mathcal{A}}^{\mathrm{DXDH}}(\lambda) := \left|\Pr[\mathcal{A}(\mathsf{par}, \mathsf{D}, \mathsf{T}_0) = 1] - \Pr[\mathcal{A}(\mathsf{par}, \mathsf{D}, \mathsf{T}_1) = 1]\right|$$

is negligible in λ. The probability is over the random choice of the parameters and over the coin tosses of $\mathcal{A}$.

Definition 4 (DXDH assumption) The DXDH assumption holds in $\mathbb{G}_1$ if no PPT algorithm has advantage at least ε in solving the DXDH problem.

Theorem 3 *Our KP-ABS scheme is EUF-CMA secure in the random oracle model under the CBDH assumption.*

We first prove our KP-ABS scheme in the selective attribute model (s-EUF-CMA) by defining the random oracle outputs for the challenge attributes that the adversary aims to forge in the setup phase. After that, we continue to transform the security into the adaptive attribute model (EUF-CMA) by relaxing this condition.

Proof sketch. We prove unforgeability via a reduction from the CBDH problem in the random oracle model. The simulator designates a special identity attribute. In the selective model, this attribute is fixed at setup and the random oracle programmed it differently from other attributes. During key generation and signing, the adversary is not allowed to query this identity attribute; such queries are safely rejected without information leakage. In the forgery phase, the adversary must output a signature on a policy or attribute set containing this identity attribute. Due to the special random oracle programming, any valid forgery embeds the CBDH challenge.

Thus, the simulator applies the forking lemma to extract the adversary's randomness and recover the CBDH solution. For adaptive security, all such rejections are replaced by abortion probabilities; we show that the overall abortion probability is negligible, yielding a sound selective-to-adaptive reduction.

Proof 3 *Suppose there exists a PPT adversary $\mathcal{A}$ that can break our KP-ABS scheme in the s-EUF-CMA game, then we can build an algorithm $\mathcal{B}$ that solves the CBDH problem. At first, $\mathcal{B}$ is given a BDH tuple $(\hat{g_1}, \hat{g_2}, \hat{g_1}^{\alpha}, \hat{g_2}^{\alpha}, \hat{g_1}^{\beta}, \hat{g_2}^{\beta}, \hat{g_1}^{\gamma}, \hat{g_2}^{\gamma})$. $\mathcal{B}$' s objective is to compute $e(\hat{g_1}, \hat{g_2})^{\alpha\beta\gamma}$, it interacts with $\mathcal{A}$ in a game as follows:*

Setup. *$\mathcal{A}$ picks an attribute set $\mathcal{S}^* = (\{u_i^*\}_{i\in[\ell^*-1]}, u_{id}^*)$ of size-ℓ^*, and a policy $\mathcal{P}^* = (\mathbf{M}^* \in \mathbb{Z}_p^{n_1^* \times n_2^*}, \pi^*, \{\pi^*(i)\}_{i\in[n_1^*]})$ with $\mathsf{P}(\mathcal{P}^*, \mathcal{S}^*) = 1$ that it intends to forge the signature. Assume u_{id}^* is an identity attribute that is only in the challenge attribute set $\mathcal{S}^*$, that is, it appears only in the forgery phase. Assume $\gamma^* = (\{\gamma_i^*\}_{i\in[\ell^*-1]}, \gamma_{id}^*)$ are the MSP recovery coefficients corresponding to $(\{u_i^*\}_{i\in[\ell^*-1]}, u_{id}^*)$ respectively. We denote the attribute universe as $\mathcal{U}$. Then $\mathcal{B}$ sets the random oracle $\mathsf{H}_0 : \{0,1\}^* \to \mathbb{Z}_p$, and sets the random oracle $\mathsf{H}_1 : \{0,1\}^* \to \mathbb{G}_1$ for different types of input as the following:*

1. *Identity attribute $u_{id}^* \in \mathcal{S}^*$: $\mathcal{B}$ picks $\delta_{id}^* \leftarrow\$ \mathbb{Z}_p$ and computes $\mathsf{H}_1(u_{id}^*) = h_{id}^* = (\hat{g_1}^{\beta})^{-\delta_{id}^*/\gamma_{id}^*}$.*
2. *Other challenge attribute $u_i^* \in \mathcal{S}^*$ $(i \in [\ell^* - 1])$: $\mathcal{B}$ picks $\delta_i^* \leftarrow\$ \mathbb{Z}_p$ and computes $\mathsf{H}_1(u_i^*) = h_i^* = \hat{g_1}^{\delta_i^*/\gamma_i^*}$.*
3. *Other attributes $u_i \in \{\mathcal{U}\backslash\mathcal{S}^*\}$: $\mathcal{B}$ picks $\delta_i \leftarrow\$ \mathbb{Z}_p$ and computes $\mathsf{H}_1(u_i) = h_i = \hat{g_1}^{\delta_i}$.*

Here h_{id}^*, h_i^* and h_i are uniform in $\mathbb{G}_1$ and independent of $\mathcal{A}$'s current view as required. Then $\mathcal{B}$ sets $g_1 \leftarrow \hat{g_1}^{\beta\cdot\delta_{id}^*}$, $g_2 \leftarrow \hat{g_2}$, and sets $X \leftarrow e(\hat{g_1}^{\beta\delta_{id}^*}, \hat{g_2}^{\alpha}) = e(\hat{g_1}, \hat{g_2})^{\alpha\beta\delta_{id}^*}$. In this way, the master secret key is implicitly set as α, and g_1^{α} is implicitly set as $\hat{g_1}^{\alpha\beta\delta_{id}^*}$. $\mathcal{B}$ generates $\mathsf{par} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \hat{g_1}^{\beta\delta_{id}^*}, \hat{g_2},)$ and sends $\mathsf{mpk} = \left(\mathsf{par}, e(\hat{g_1}, \hat{g_2})^{\alpha\beta\delta_{id}^*}\right)$ to $\mathcal{A}$.

Key generation oracle. *When $\mathcal{A}$ issues a secret key query with input an access policy $\mathcal{P} = (\mathbf{M} \in \mathbb{Z}_p^{n_1\times n_2}, \pi, \{\pi(i)\}_{i\in[n_1]})$, $\mathcal{B}$ checks if $\mathsf{P}(\mathcal{P}, \mathcal{S}^*) = 1$ or if $u_{id}^* \in \{\pi(i)\}_{i\in[n_1]}$. If any of them is true, reject the query. Otherwise, $\mathcal{B}$ runs the random oracle H_1 in Cases (2) or (3) as described before to obtain the corresponding $h_i \in \mathbb{G}_1$ for each attribute $\pi(i)$. There are three cases for the input of $\{\pi(i)\}_{i\in[n_1]}$: (1) $\{\pi(i)\}_{i\in[n_1]} \subseteq \{\mathcal{S}^*\backslash u_{id}^*\}$, (2) $\{\pi(i)\}_{i\in[n_1]} \cap \{\mathcal{S}^*\backslash u_{id}^*\} = \varnothing$, and (3) $\{\pi(i)\}_{i\in[n_1]} \cap \{\mathcal{S}^*\backslash u_{id}^*\} \neq \varnothing$. $\mathcal{B}$ chooses $r' \leftarrow\$ \mathbb{Z}_p$, $\mathbf{v} \leftarrow\$ \mathbb{Z}_p^{n_2-1}$, implicitly sets $r = -\alpha + r'$ and generates the first key element $\mathsf{sk}_1 = \hat{g_2}^{-\alpha+r'}$, which will be used for all three cases. The second key element is different for each case: Case 1:* $\mathsf{sk}_{2,i} = \hat{g_1}^{\beta\delta_{id}^*\cdot\mathbf{M}_{i,1}(\alpha-\alpha+r') + \sum_{j=1}^{n_2}\mathbf{M}_{i,j}\mathbf{v}_j} \cdot \hat{g_1}^{\frac{\delta_i^*}{\gamma_i^*}\cdot(-\alpha+r')}$ $= \hat{g_1}^{\beta\delta_{id}^*\cdot\mathbf{M}_{i,1}r' + \sum_{j=1}^{n_2}\mathbf{M}_{i,j}\mathbf{v}_j} \cdot \hat{g_1}^{(-\alpha+r')\cdot\frac{\delta_i^*}{\gamma_i^*}}$. *Case 2: the only difference is no γ values, so* $\mathsf{sk}_{2,i} = \hat{g_1}^{\beta\delta_{id}^*\cdot\mathbf{M}_{i,1}r' + \sum_{j=1}^{n_2}\mathbf{M}_{i,j}\mathbf{v}_j} \cdot \hat{g_1}^{(-\alpha+r')\cdot\delta_i}$. *Case 3: as $\mathcal{B}$ can simulate case 1 and case 2 with the input attributes both in $\{\mathcal{S}^*\backslash u_{id}^*\}$ and not in $\{\mathcal{S}^*\backslash u_{id}^*\}$, the $\mathsf{sk}_{2,i}$ in case 3 can be simulated by a hybrid of case 1 and case 2 dependent on if the attributes are in $\{\mathcal{S}^*\backslash u_{id}^*\}$ or not. Then $\mathcal{B}$ returns $\mathsf{sk} = (\mathsf{sk}_1, \{\mathsf{sk}_{2,i}\}_{i\in[n_1]})$ to $\mathcal{A}$.*

Signing oracle. *When $\mathcal{A}$ issues a signature query, with input a set of attributes $\mathcal{S} = \{u_i\}_{i\in I}$, an access policy $\mathcal{P} = (\mathbf{M} \in \mathbb{Z}_p^{n_1\times n_2}, \pi, \{\pi(i)\}_{i\in[n_1]})$ with $\mathsf{P}(\mathcal{P}, \mathcal{S}) = 1$, and a message msg. $\mathcal{B}$ checks if $\mathsf{P}(\mathcal{P}, \mathcal{S}^*) = 1$. If it is true, reject the query. Assume I is the index set that $\mathcal{S}$ matches the rows of $\mathbf{M}$, i.e., $\mathcal{S} = \{\pi(i)\}_{i\in I}$. $\mathcal{B}$ runs the random oracle to obtain the corresponding h_i for each attribute $\pi(i)$ in the same way as discussed before. We now describe the simulation of the attribute commitment (A, B, C) in the signature, which also has three different cases in terms of the type of attributes included in the subset: (1) $\{\pi(i)\}_{i\in I} \subseteq \{\mathcal{S}^*\backslash u_{id}^*\}$, (2) $\{\pi(i)\}_{i\in I} \cap \{\mathcal{S}^*\backslash u_{id}^*\} = \varnothing$, and (3) $\{\pi(i)\}_{i\in I} \cap \{\mathcal{S}^*\backslash u_{id}^*\} \neq \varnothing$. For all cases, $\mathcal{B}$ (1) generates a set of coefficients $\{\gamma_i\}_{i\in I}$ corresponding to $\{\pi(i)\}_{i\in I}$ respectively, (2) chooses $r' \leftarrow\$ \mathbb{Z}_p$ and implicitly sets $r = \alpha + r'$, (3) picks $k, t \leftarrow\$ \mathbb{Z}_p$. For Case 1, $\mathcal{B}$ generates* $A = \hat{g_1}^{\beta\delta_{id}^*\cdot(\alpha-\alpha+r')\cdot kt} \cdot \prod_{i\in I}\hat{g_1}^{\frac{\delta_i^*}{\gamma_i^*}\cdot(-\alpha+r')\cdot\gamma_i kt} = \hat{g_1}^{\beta\delta_{id}^* r' kt} \cdot \hat{g_1}^{(-\alpha+r')kt\cdot\sum_{i\in I}\frac{\delta_i^*\gamma_i}{\gamma_i^*}}$, $B = \hat{g_1}^{\beta\delta_{id}^*\cdot k} \cdot \hat{g_1}^{k\cdot\sum_{i\in I}\frac{\delta_i^*\gamma_i}{\gamma_i^*}}$, $C = \hat{g_2}^{(-\alpha+r')\cdot t}$. *For Case 2,* $A = \hat{g_1}^{\beta\delta_{id}^*\cdot(\alpha-\alpha+r')\cdot kt} \cdot \prod_{i\in I}\hat{g_1}^{\delta_i\cdot(-\alpha+r')\cdot\gamma_i kt}$ $= \hat{g_1}^{\beta\delta_{id}^* r' kt} \cdot \hat{g_1}^{(-\alpha+r')kt\cdot\sum_{i\in I}\delta_i\gamma_i}$, $B = \hat{g_1}^{\beta\delta_{id}^*\cdot k} \cdot \hat{g_1}^{k\cdot\sum_{i\in I}\delta_i\gamma_i}$, $C = \hat{g_2}^{(-\alpha+r')\cdot t}$. *For Case 3, $\mathcal{B}$ can simulate A, B, C through a hybrid of the previous two cases.*

Then $\mathcal{B}$ simulates the second part of signature by directly following the scheme. First, $\mathcal{B}$ picks $r_\alpha, r_k, \{r_i\}_{i\in[n_1]} \leftarrow\$ \mathbb{Z}_p$, then generates $Y = e(\hat{g_1}, \hat{g_2})^{\alpha\beta\delta_{id}^*\cdot kt}$, $Z = e(\hat{g_1}, \hat{g_2})^{\alpha\beta\delta_{id}^*\cdot r_\alpha}$, $W = \hat{g_1}^{\beta\delta_{id}^* r_k} \cdot \prod_{i\in I} h_i^{r_i}$, $c = \mathsf{H}_0(A, B, C, Y, Z, W, \mathsf{msg})$, $s_\alpha = r_\alpha - kt\cdot c$, $s_k = r_k - k\cdot c$, $\forall i \in I : s_i = r_i - \gamma_i k\cdot c$. Similar to the three cases analyzed for A, B, C, the random oracle H_1 can simulate each h_i value based on if each $\pi(i)$ is in the challenge attribute set $\mathcal{S}^*$ or not. It is straightforward to verify that $(c, s_\alpha, s_k, \{s_i\}_{i\in[n_1]})$ is valid by computing (1) $Y' = \frac{e(A,\, g_2)}{e(B,\, C)} = e(\hat{g_1}, \hat{g_2})^{\alpha\beta\delta_{id}^* kt}$, (2) $Z' = e(\hat{g_1}, \hat{g_2})^{\alpha\beta\delta_{id}^*\cdot s_\alpha} \cdot Y'^c$, (3) $W' = \hat{g_1}^{\beta\delta_{id}^* s_k} \cdot \prod_{i\in I} h_i^{s_i} \cdot B^c$, (4) $c' = \mathsf{H}_0(A, B, C, Y', Z', W', \mathsf{msg}^*)$, and obtaining $c = c'$. This indicates that $\mathcal{B}$ can simulate a valid signature $\sigma = (A, B, C, c, s_\alpha, s_k, \{s_i\}_{i\in I})$ for $\mathcal{A}$'s query without knowing the answer of the CBDH problem.

Forgery. *When $\mathcal{A}$ outputs a message msg^* and a valid signature $\sigma^* = (A^*, B^*, C^*, c^*, s_k^*, s_\alpha^*, \{s_i^*\}_{i\in[\ell^*]})$, there exists $r^*, k^*, t^* \in \mathbb{Z}_p$ such that*

$$A^* = \hat{g_1}^{\beta\delta_{id}^*\cdot(\alpha k^* t^* + r^* k^* t^*)} \cdot \hat{g_1}^{-\beta\cdot\frac{\delta_{id}^*}{\gamma_{id}^*}\cdot\gamma_{id}^* r^* k^* t^*} \prod_{i\in[\ell^*-1]} \hat{g_1}^{\frac{\delta_i^*}{\gamma_i^*}\cdot\gamma_i^* r^* k^* t^*}$$
$$= \hat{g_1}^{\alpha\beta\delta_{id}^* k^* t^*} \cdot \hat{g_1}^{r^* k^* t^*\cdot\sum_{i\in[\ell^*-1]}\delta_i^*}$$
$$B^* = \hat{g_1}^{\beta\delta_{id}^* k} \cdot \hat{g_1}^{-\beta\cdot\frac{\delta_{id}^*}{\gamma_{id}^*}\cdot\gamma_{id}^* k^*} \prod_{i\in[\ell^*-1]} \hat{g_1}^{\frac{\delta_i^*}{\gamma_i^*}\cdot\gamma_i^* k} = \hat{g_1}^{k\cdot\sum_{i\in[\ell^*-1]}\delta_i^*}$$

$$C^* = \hat{g_2}^{r^* t^*}, Y^* = e(\hat{g_1}, \hat{g_2})^{\alpha\beta\delta^*_{id} k^* t^*},\ Z^* = e(\hat{g_1}, \hat{g_2})^{\alpha\beta\delta^*_{id} r^*_\alpha}$$

$$W^* = \hat{g_1}^{\beta\delta^*_{id} r^*_k} \cdot \hat{g_1}^{-\beta \cdot \frac{\delta^*_{id}}{\gamma^*_{id}} \cdot r^*_{id}} \hat{g_1}^{\sum_{i\in[\ell^*-1]} \frac{\delta^*_i}{\gamma^*_i} \cdot r^*_i}$$

$$c^* = \mathsf{H}_0(A^*, B^*, C^*, Y^*, Z^*, W^*, \mathsf{msg}^*)$$

$$s^*_\alpha = r^*_\alpha - k^* t^* \cdot c^*,\ s^*_k = r^*_k - k^* \cdot c^*,\ \forall i \in [\ell^*] : s^*_i = r^*_i - \gamma^*_i k^* \cdot c^*$$

Since the tuple $(c^*, s^*_\alpha, s^*_k, \{s^*_i\}_{i\in l^*})$ in our signature is a Schnorr type of signature which satisfies the "generic digital signature" notion defined in [106], $\mathcal{B}$ could use the forking lemma to extract the value k^*t^* hidden in s^*_α, and the value k^* hidden in s^*_k: After a polynomial time of queries to the random oracle H_0, $\mathcal{A}$ obtains c^*_1, c^*_2 at some point where $c^*_1 \neq c^*_2$ and generates two valid signatures $(s^*_{\alpha_1}, s^*_{k,1}, c^*_1)$ and $(s^*_{\alpha_2}, s^*_{k,2}, c^*_2)$. Then $\mathcal{A}$ sends these signatures to $\mathcal{B}$. According to the scheme, $s^*_{\alpha_1} = r^*_\alpha - k^* t^* \cdot c^*_1$, $s^*_{\alpha_2} = r^*_\alpha - k^* t^* \cdot c^*_2$. Then $\mathcal{B}$ can get $s^*_{\alpha_1} - s^*_{\alpha_2} = k^* t^* \cdot (c^*_2 - c^*_1)$. Therefore, $k^* t^* = \frac{s^*_{\alpha_1} - s^*_{\alpha_2}}{c^*_2 - c^*_1}$. Similarly, $\mathcal{B}$ can obtain $s^*_{k,1} - s^*_{k,2} = k^* \cdot (c^*_2 - c^*_1)$, so $k^* = \frac{s^*_{k,1} - s^*_{k,2}}{c^*_2 - c^*_1}$. By leveraging the knowledge of k^*t^* and k^*, $\mathcal{B}$ can first calculate the value of t^*, and then compute $\hat{g_2}^{r^*} = C^{*\frac{1}{t^*}}$. Finally, $\mathcal{B}$ can extract the answer of the CBDH problem from the forgery by the following:

$$\mathsf{T} = \left(\frac{e(A^*, \hat{g_2}^\gamma)}{e(\prod_{i\in[\ell^*-1]} \hat{g_1}^{\gamma \cdot \delta^*_i}, \hat{g_2}^{r^*})^{k^* t^*}} \right)^{\frac{1}{\delta^*_{id} k^* t^*}}.$$

In conclusion, by assuming $\mathcal{B}$ breaks the CBDH assumption with advantage $\mathrm{Adv}^{\mathrm{CBDH}}_{\mathcal{B}}(\lambda)$ over the choice of λ, the advantage $\mathcal{A}$ wins the s-EUF-CMA game can be reduced to [3]

$$\mathrm{Adv}^{\text{s-EUF-CMA}}_{\mathcal{A}}(\lambda) \leq \mathrm{Adv}^{\mathrm{CBDH}}_{\mathcal{B}}(\lambda).$$

The above analysis for s-EUF-CMA has several constraints defined in the setup phase: $\mathcal{B}$ has to set the challenge policy, challenge attribute subset, and an identity attribute, which implicitly requires $\mathcal{B}$ to know the values $(\{\delta^*_i\}_{i\in[\ell^*-1]}, \delta^*_{id})$, and $(\{\gamma^*_i\}_{i\in[\ell^*-1]}, \gamma^*_{id})$ at the beginning of the game. To remove these constraints and transform the unforgeability into adaptive attribute model, we modify the game into the EUF-CMA game as follows:

- *We assume that the MSP encoding technique supports that secret recovery coefficients γ_i are equal to some predefined values if the attribute sets satisfy the access policy. For example, we can convert an access policy into a Boolean formula and then to an MSP using the Lewko-Waters method [46] so that a matrix $\mathbf{M}$ has only entries in $\{0, 1, -1\}$ and $\{\gamma_i\}$ that are always 1. In this case, while $\mathcal{B}$ does not know the challenge attribute set at setup, it fixes its size l^* and knows the set will satisfy a challenge policy chosen later by the adversary. Thus, it can predefine l^* outputs of H_1, i.e., $(\{\gamma^*_i\}_{i\in[\ell^*-1]}, \gamma^*_{id})$ all equal to 1 in setup.*
- *$\mathcal{B}$ chooses $\ell^* \leftarrow\$ \mathbb{Z}_p$ and $\delta^*_{id}, \{\delta^*_i\}_{i\in[\ell^*-1]} \leftarrow\$ \mathbb{Z}_p$, and then sets ℓ^* outputs of H_1 as $\left\{\hat{g_1}^{\delta^*_i/\gamma^*_i}\right\}_{i\in[\ell^*-1]}$ and $\hat{g_1}^{\beta(-\delta^*_{id}/\gamma^*_{id})}$ respectively, without knowing the corresponding ℓ^* inputs to the random oracle.*
- *Apart from the above changes, the game runs in the same way as the s-EUF-CMA game. When $\mathcal{A}$ queries random oracle H_1, $\mathcal{B}$ randomly sets ℓ^* $\mathcal{A}$'s inputs as $(\{u^*_i\}_{i\in[\ell^*-1]}, u^*_{id})$ and correspondingly returns to $\mathcal{A}$ the pre-determined outputs $\left\{\hat{g_1}^{\delta^*_i/\gamma^*_i}\right\}_{i\in[\ell^*-1]}$ and $\hat{g_1}^{\beta(-\delta^*_{id}/\gamma^*_{id})}$ respectively, and answer other queries u_i by picking $\delta_i \leftarrow\$ \mathbb{Z}_p$ and returning $\mathsf{H}_1(u_i) = h_i = \hat{g_1}^{\delta_i}$. When $\mathcal{A}$ queries the random oracle H_0, $\mathcal{B}$ answers the queries with random numbers in $\mathbb{Z}_p$.*

Assume $\mathcal{A}$ has queried the random oracles $\mathsf{H}_0, \mathsf{H}_1$ for $q_{\mathsf{H}_0}, q_{\mathsf{H}_1}$ times and performed the key generation query and signing oracle query for q_{key}, q_{sign} times, respectively. Assume the output message space for $\mathsf{H}_0, \mathsf{H}_1$ are s_0, s_1 respectively, and the attribute universe $\mathcal{U}$ has size n. Then we analyze the cases that $\mathcal{B}$ does not abort in the adaptive game as follows:

- *$q_{\mathsf{H}_1} > \ell^*$ with probability p_1.*
- *$\mathcal{A}$ has never queried a secret key for u^*_{id} in q_{key} key generation queries, with probability $1 - \frac{1}{q_{\mathsf{key}}}$.*
- *$\mathcal{A}$ has never queried a signature for u^*_{id} in q_{sign} signing queries, with probability $1 - \frac{1}{q_{\mathsf{sign}}}$.*
- *$\mathcal{A}$ chooses $(\{u^*_i\}_{i\in[\ell^*-1]}, u^*_{id})$ as the challenge attribute set with probability $\frac{(n-l^*)! \cdot q_{\mathsf{H}_1}!}{(q_{\mathsf{H}_1} - l^*)! \cdot n!}$. This is the probability that l^* fixed elements appear in a size-q_{H_1} subset drawn from the size-n attribute universe. As n is polynomially bounded and l^* is small, this probability is non-negligible.*
- *Both random oracles $\mathsf{H}_0, \mathsf{H}_1$ has no collisions for $\mathcal{A}$'s queries, with probability $\frac{q^2_{\mathsf{H}_0}}{2^{s_0}}$ and $\frac{q^2_{\mathsf{H}_1} \cdot \ell^*}{2^{s_1}}$ respectively.*

$\mathcal{B}$ will pass all the simulations and find the forgery result only when all the above cases happen at the same time. Therefore, the advantage that $\mathcal{A}$ wins the adaptive security game is:

$$\mathrm{Adv}^{\text{EUF-CMA}}_{\mathcal{A}}(\lambda) \leq \mathrm{Adv}^{\text{s-EUF-CMA}}_{\mathcal{A}}(\lambda) \cdot p_1 \cdot \left(1 - \frac{1}{q_{\mathsf{key}}}\right) \cdot \left(1 - \frac{1}{q_{\mathsf{sign}}}\right) \cdot \frac{(n-l^*)! \cdot q_{\mathsf{H}_1}!}{(q_{\mathsf{H}_1} - l^*)! \cdot n!} \cdot \frac{q^2_{\mathsf{H}_0}}{2^{s_0}} \cdot \frac{q^2_{\mathsf{H}_1} \cdot \ell^*}{2^{s_1}}.$$

Theorem 4 *Our KP-ABS scheme is anonymous in the random oracle model under the DXDH assumption.*

The proof embbeds the DXDH challenge term in the commitment part (A, B) of the simulated signature. Together with the randomness masking from the Schnorr signature, it ensures indistinguishability of signatures so that any adversary

[3] Note that this is only a qualitative security bound for simplicity. There should be an asymptotic square root loss for the probability on $\mathrm{Adv}^{\mathrm{CBDH}}_{\mathcal{B}}(\lambda)$. Readers may refer to [107] for more details.

breaking anonymity with advantage ε yields a DXDH distinguisher with advantage $\varepsilon/2$.

Proof 4 *Suppose there exists a PPT adversary $\mathcal{A}$ that can win the anonymity game with advantage ε, we can build an algorithm $\mathcal{B}$ that solves the DXDH problem by advantage $\varepsilon/2$. Algorithm $\mathcal{B}$ is given an DXDH tuple, where we replace the variable names in the DXDH tuple by following the ABS scheme: $a=\tau$, $b=k^*$. The DXDH tuple $(\hat{g_1}, \hat{g_2}, \hat{g_1}^{\tau}, \hat{g_1}^{k^*}, T)$ is given to $\mathcal{B}$.*

Let $z \leftarrow\$ \mathbb{Z}_p$. The challenger flips random coin μ outside of $\mathcal{B}$'s view. If $\mu=0$, the challenger sets $T=\hat{g_1}^{\tau k^*}$. If $\mu=1$, the challenger sets $T=\hat{g_1}^{z}$. $\mathcal{B}$'s goal is to output 0 if $T=\hat{g_1}^{\tau k^*}$, or output 1 otherwise. $\mathcal{B}$ works by interacting with $\mathcal{A}$ in a game as follows:

Setup. *$\mathcal{B}$ sets $g_1 \leftarrow \hat{g_1}^{\tau}$, $g_2 \leftarrow \hat{g_2}$, α is implicitly set to $\frac{1}{\tau}$, so $X=e(\hat{g_1}^{\tau}, \hat{g_2})^{\frac{1}{\tau}}=e(\hat{g_1}, \hat{g_2})$. Then $\mathcal{B}$ generates $\mathsf{par}=(p,\ \mathbb{G}_1,\ \mathbb{G}_2,\ \mathbb{G}_T,\ e, \hat{g_1}^{\tau}, \hat{g_2})$ and sends $\mathcal{A}$ $\mathsf{mpk}=(\mathsf{H}_0, \mathsf{H}_1, \mathsf{par}, e(\hat{g_1}, \hat{g_2}))$.*

Phase 1. *$\mathcal{A}$ can query two random oracles H_0 and H_1, a key generation oracle, and a signing oracle. They are defined as follows:*

Random oracle H_0: $\mathcal{B}$ maintains a list R with entries of the form $\langle x, h_x \rangle$, which is initially empty. When $\mathcal{A}$ inputs a value x, $\mathcal{B}$ checks if h_x already appears on R in a tuple $\langle x, h_x \rangle$. If yes, then $\mathcal{B}$ responds with $\mathsf{H}_0(x)=h_x$. Otherwise, $\mathcal{B}$ picks $h_x \leftarrow\$ \mathbb{Z}_p$ and adds the tuple $\langle x, h_x \rangle$ to R and responds to $\mathcal{A}$ by setting $\mathsf{H}_0(x)=h_x$. h_x is uniform in $\mathbb{Z}_p$ and is independent of $\mathcal{A}$'s current view as required.

Random oracle H_1: $\mathcal{B}$ maintains a list L with entries of the form $\langle u_i, h_i, \delta_i \rangle$, which is initially empty. When $\mathcal{A}$ inputs a value u_i, $\mathcal{B}$ checks if u_i already appears on L in a tuple $\langle u_i, h_i, \delta_i \rangle$. If yes, then $\mathcal{B}$ responds with $\mathsf{H}_1(u_i)=h_i \in \mathbb{G}_1$. Otherwise, $\mathcal{B}$ picks $\delta_i \leftarrow\$ \mathbb{Z}_p$ and computes $h_i \leftarrow \hat{g_1}^{\delta_i} \in \mathbb{G}_1$. Then $\mathcal{B}$ adds the tuple $\langle u_i, h_i, \delta_i \rangle$ to L and responds to $\mathcal{A}$ by setting $\mathsf{H}_1(u_i)=h_i$. h_i is uniform in $\mathbb{G}_1$ and is independent of $\mathcal{A}$'s current view as required.

Key generation oracle: When $\mathcal{A}$ issues a query for generating a secret key on an access policy $\mathcal{P}=(\mathbf{M} \in \mathbb{Z}_p^{n_1 \times n_2}, \pi, \{\pi(i)\}_{i\in[n_1]})$, $\mathcal{B}$ runs the random oracle H_1 to obtain $h_i \in \mathbb{G}_1$ for each attribute $\pi(i)$ such that $\mathsf{H}_1(\pi(i))=h_i$. Let $\langle \pi(i), h_i, \delta_i \rangle$ be the corresponding tuple on list L, then $\mathcal{B}$ picks $r \leftarrow\$ \mathbb{Z}_p$, $\mathbf{v} \leftarrow\$ \mathbb{Z}_p^{n_2-1}$ and generates the key as follows: $\mathsf{sk}_1=\hat{g_2}^{r}$, $\mathsf{sk}_{2,i}=\hat{g_1}^{\tau\cdot(\mathbf{M}_{i,1}\cdot(\frac{1}{\tau}+r)+\sum_{j=2}^{n_2}\mathbf{M}_{i,j}\mathbf{v}_j)} \cdot \hat{g_1}^{\delta_i r}=\hat{g_1}^{(1+\tau r)\cdot\mathbf{M}_{i,1}+\tau\cdot\sum_{j=2}^{n_2}\mathbf{M}_{i,j}\mathbf{v}_j+\delta_i r}$. Then $\mathcal{B}$ returns the secret key $\mathsf{sk}=(\mathsf{sk}_1, \{\mathsf{sk}_{2,i}\}_{i\in[n_1]})$.

Signing oracle: When $\mathcal{A}$ issues a query to sign a message msg with an attribute set $\mathcal{S}=\{u_i\}_{i\in[m]}$ under an access policy $\mathcal{P}=(\mathbf{M} \in \mathbb{Z}_p^{n_1 \times n_2}, \pi, \{\pi(i)\}_{i\in[n_1]})$ with $\mathsf{P}(\mathcal{P}, \mathcal{S})=1$. Assume I is the index set that $\mathcal{S}$ matches the rows of $\mathbf{M}$, i.e., $\mathcal{S}=\{\pi(i)\}_{i\in I}$. $\mathcal{B}$ calculates the secret recovery coefficients $\{\gamma_i\}_{i\in I}$ and runs the random oracle H_1 to obtain $h_i \in \mathbb{G}_1$ for each attribute $\pi(i)$ such that $\mathsf{H}_1(\pi(i))=h_i$. Let $\langle \pi(i), h_i, \delta_i \rangle$ be the corresponding tuple on list L, then $\mathcal{B}$ picks $r, k, t, r_\alpha, r_k, \{r_i\}_{i\in[n_1]} \leftarrow\$ \mathbb{Z}_p$, and simulates the signature: $A=\hat{g_1}^{\tau\cdot\frac{1}{\tau}\cdot kt}\cdot(\hat{g_1}^{\tau})^{rkt}\cdot\prod_{i\in I}\hat{g_1}^{\delta_i\gamma_i rkt}=\hat{g_1}^{kt}\cdot\hat{g_1}^{\tau rkt}\cdot\hat{g_1}^{rkt\cdot\sum_{i\in I}\delta_i\gamma_i}$, $B=\hat{g_1}^{\tau\cdot k+k\cdot\sum_{i\in I}\delta_i\gamma_i}$, $C=\hat{g_2}^{rt}$, $Y=e(\hat{g_1}, \hat{g_2})^{kt}$, $Z=e(\hat{g_1}, \hat{g_2})^{r\alpha}$, $W=\hat{g_1}^{\tau\cdot r_k+\sum_{i\in I}\delta_i r_i}$, $c=\mathsf{H}_0(A, B, C, Y, Z, W, \mathsf{msg})$, $s_\alpha=r_\alpha-kt\cdot c$, $s_k=r_k-k\cdot c$, $\forall_{i\in I}: s_i=r_i-\gamma_i k\cdot c$.

It is easy to verify that the above signature is valid by computing (1) $Y'=\frac{e(A,\ g_2)}{e(B,\ C)}=e(\hat{g_1}, \hat{g_2})^{kt}$, (2) $Z'=e(\hat{g_1}, \hat{g_2})^{s_\alpha}\cdot Y'^{c}$, (3) $W'=\hat{g_1}^{\tau s_k}\cdot\prod_{i\in I}\hat{g_1}^{\delta_i s_i}\cdot B^{c}$, (4) $c'=\mathsf{H}_0(A, B, C, Y', Z', W', \mathsf{msg}^*)$, and obtaining $c=c'$. Thus, the signature $\sigma=(A, B, C, c, s_\alpha, s_k, \{s_i\}_{i\in I})$ is a valid signature for $(\mathcal{P}, \mathcal{S}, \mathsf{msg})$. Then $\mathcal{B}$ sends σ to $\mathcal{A}$.

Challenge. *$\mathcal{A}$ outputs a challenge tuple $(\mathcal{P}_0^*, \mathcal{P}_1^*, \mathcal{S}^*, \mathsf{msg}^*)$, in which the policies $\mathcal{P}_0^*=\left(\mathbf{M}_0^* \in \mathbb{Z}_p^{n_1^* \times n_2^*},\ \pi_0^*,\ \{\pi_0^*(i)\}_{i\in[n_1^*]}\right)$, $\mathcal{P}_1^*=\left(\mathbf{M}_1^* \in \mathbb{Z}_p^{n_3^* \times n_4^*},\ \pi_1^*,\ \{\pi_1^*(i)\}_{i\in[n_3^*]}\right)$, and challenge attribute set $\mathcal{S}^*$ satisfying $\mathsf{P}(\mathcal{P}_0^*, \mathcal{S}^*)=\mathsf{P}(\mathcal{P}_1^*, \mathcal{S}^*)=1$. $\mathcal{B}$ randomly picks $b \in \{0,\ 1\}$ and signs msg^* with $\mathcal{S}^*$ under $\mathcal{P}_b^*$. Assume I_b^* is the index set that $\mathcal{S}^*$ matches the rows of $\mathbf{M}_b^*$, $\mathcal{B}$ first calculates the secret recovery coefficients $\{\gamma_{i,b}^*\}_{i\in I_b^*}$, then $\mathcal{B}$ chooses $r^*, t^* \leftarrow\$ \mathbb{Z}_p$ and runs the random oracle H_1 to obtain $h_i^* \in \mathbb{G}_1$ for each attribute value $\pi(i)^*$ such that $\mathsf{H}_1(\pi(i)^*)=h_i^*$. Let $\langle \pi(i)^*,\ h_i^*,\ \delta_i^* \rangle$ be the corresponding tuple on list L, $\mathcal{B}$ simulates the challenge signature as follows: $A^*=\hat{g_1}^{\tau\cdot\frac{1}{\tau}\cdot k^*t^*}\cdot T^{r^*t^*}\cdot\prod_{i\in I_b^*}\hat{g_1}^{k^*\delta_i^*\gamma_{i,b}^* r^*t^*}=\hat{g_1}^{k^*t^*+k^*r^*t^*\cdot\sum_{i\in I_b^*}\delta_i^*\gamma_{i,b}^*}\cdot T^{r^*t^*}$, $B^*=T\cdot\hat{g_1}^{k^*\cdot\sum_{i\in I_b^*}\delta_i^*\gamma_{i,b}^*}$, $C^*=\hat{g_2}^{r^*t^*}$, $c^*, s_\alpha^*, s_k^*, \{s_i^*\}_{i\in I_b^*} \leftarrow\$ \mathbb{Z}_p$, $Y^*=e(\hat{g_1}^{k^*}, \hat{g_2})^{t^*}=e(\hat{g_1}, \hat{g_2})^{k^*t^*}$, $W^*=\left(\hat{g_1}^{\tau s_k^*}\prod_{i\in I_b^*}\hat{g_1}^{\delta_i^* s_i^*}\right)\cdot(B^*)^{c^*}$. Then $\mathcal{B}$ stores $\langle (A^*, B^*, C^*, Y^*, Z^*, W^*, \mathsf{msg}), c^* \rangle$ as an input-output pair for H_0.*

If $\mu=0$, $T=\hat{g_1}^{\tau k^*}$, $A^*=\hat{g_1}^{k^*t^*+\tau k^*r^*t^*+k^*r^*t^*\cdot\sum_{i\in I_b^*}\delta_i^*\gamma_{i,b}^*}$, $B^*=\hat{g_1}^{\tau k^*+k^*\sum_{i\in I_b^*}\delta_i^*\gamma_{i,b}^*}$. σ_b^* is a valid signature for the challenge tuple $(\mathcal{P}_b^*, \mathcal{S}^*, \mathsf{msg}^*)$ instantiated with valid DXDH elements. If $\mu=1$, $T=\hat{g_1}^{z}$, $A^*=\hat{g_1}^{k^*t^*+k^*r^*t^*\cdot\sum_{i\in I_b^*}\delta_i^*\gamma_{i,b}^*+zr^*t^*}$, $B^*=\hat{g_1}^{z+k^*\cdot\sum_{i\in I_b^*}\delta_i^*\gamma_{i,b}^*}$. σ_b^* is also a valid signature for the challenge tuple $(\mathcal{P}_b^*, \mathcal{S}^*, \mathsf{msg}^*)$ with masking the randomness z chosen by the challenger. Thus, $\sigma_b^*=(A^*, B^*, C^*, c^*, s_\alpha^*, s_k^*, \{s_i^*\}_{i\in[n_1^*]})$ is a valid challenge signature for $(\mathcal{P}_b^*, \mathcal{S}^*, \mathsf{msg}^*)$ from $\mathcal{A}$'s view.

Phase 2. *Same as Phase 1.*

Guess. *$\mathcal{A}$ outputs a bit $b' \in \{0,\ 1\}$. If $b'=b$, $\mathcal{B}$ outputs $\mu'=0$ indicating T is a valid DXDH element (i.e., $T=\hat{g_1}^{\tau k^*}$). Otherwise, it outputs $\mu'=1$ indicating T is given as a random element. In the case of $\mu=1$, as z is random, A^*, B^* are random elements on $\mathbb{G}_1$ hence (A^*, B^*, C^*) contains no information about $\mathcal{P}_b^*$ and $\gamma_{i,b}^*$ respectively. Besides, there is no information about k^*t^*, k^*, and $\gamma_{i,b}^* k^*$ as the randomness r_α, r_k and $\{r_i\}_{i\in I_b^*}$ are set implicitly in s_α^*, s_k^*, and $\{s_i^*\}_{i\in I_b^*}$ respectively and are unknown to $\mathcal{A}$. Thus, $\mathcal{A}$ gains no information about b from $(c^*, s_\alpha^*, s_k^*, \{s_i^*\}_{i\in I_b^*})$. Then we have $\Pr[b \neq b' | \mu=1]=\frac{1}{2}$. As $\mathcal{B}$ guesses $\mu=1$ when $b \neq b'$, we*

Table 3: Time (ms) for Setup and operations on BN254 curve

Scheme	Setup
RD16 [21]	170
Our KP-ABS	28
KCGD14 [28]	275
Our SP-ABS	29

Group	Sample	Mul.	Exp.	I2P	Pairing
$\mathbb{G}_1$	0.79	0.0018	0.79	0.03	
$\mathbb{G}_2$	1.81	0.0040	1.24	0.07	25.4
$\mathbb{G}_T$	-	0.0288	6.09	-	

- The setup time of RD16 is measured when the maximum bound n is set to 100.
- The setup time of KCGD14 is measured when the attribute universe $|\mathcal{U}_s|$ is set to 100.
- Our ABS schemes support a large universe and have constant setup time.

have $\Pr[\mu' = \mu | \mu = 1] = \frac{1}{2}$. *If* $\mu = 0$, *then* $\mathcal{A}$ *sees a signature for* I^*_b, *with an advantage* ε *of breaking the scheme. Thus, we have* $\Pr[b = b' | \mu = 0] = \frac{1}{2} + \varepsilon$. *As* $\mathcal{B}$ *guesses* $\mu = 0$ *when* $b = b'$, *we have* $\Pr[\mu' = \mu | \mu = 0] = \frac{1}{2} + \varepsilon$. *Therefore, the overall advantage of* $\mathcal{B}$ *solves the decisional XDH problem is* $\frac{1}{2}\Pr[\mu' = \mu | \mu = 0] + \frac{1}{2}\Pr[\mu' = \mu | \mu = 1] - \frac{1}{2} = \frac{1}{2} \cdot (\frac{1}{2} + \varepsilon) + \frac{1}{2} \cdot \frac{1}{2} - \frac{1}{2} = \frac{\varepsilon}{2}$. *Therefore, we complete the proof of Theorem 4.*

Theorem 5 *Our SP-ABS scheme is EUF-CMA secure in the random oracle model under the CBDH assumption.*

This proof uses techniques similar to the proof of theorem 3. We first prove our SP-ABS scheme in the sEUF-CMA model, then we relax the restrictions at the setup phase by using the random oracle, and transform it into the EUF-CMA model. The proof of this theorem is given in the full version of this paper [36].

Theorem 6 *Our SP-ABS scheme is anonymous in the random oracle model under the DXDH assumption.*

This proof uses techniques similar to the proof of theorem 4. The proof of this theorem is also given in the full version of this paper [36].

6 Implementation

We implemented our ABS schemes and compared them with the state-of-the-art. Our implementation is the first in the MSP-based ABS field. Our experiments are run in Python 3.9.16 using the Charm 0.5 framework [108] and the BN254 curve for pairings, which is considered secure based on current knowledge and is widely used in commercial applications. All running times below were measured on a PC with a 3.59 GHz AMD Ryzen 5 3600 6-Core Processor and 16GB RAM. Our code is available on Github [109].

Comparison rationale. Our goal is to compare the efficiency of our ABS schemes with state-of-the-art constructions listed in Table 1. For KP-ABS, RD16 [21] proposes two schemes: one for a small universe and one for a large universe. Both achieve constant-size signatures and require a constant number of pairings in verification, but are asymptotically slower than ours in key generation and signing time. The key difference between their schemes is that the small-universe scheme fixes all attributes as group generators at setup, whereas the large-universe scheme only specifies an upper bound on the attribute set size, making it more practical. Therefore, we implemented our scheme alongside the large-universe scheme of RD16. Since RD16 is built over Type-I pairings, we converted their large-universe construction to Type-III pairings for a fair comparison; the transformation details are given in Appendix A.

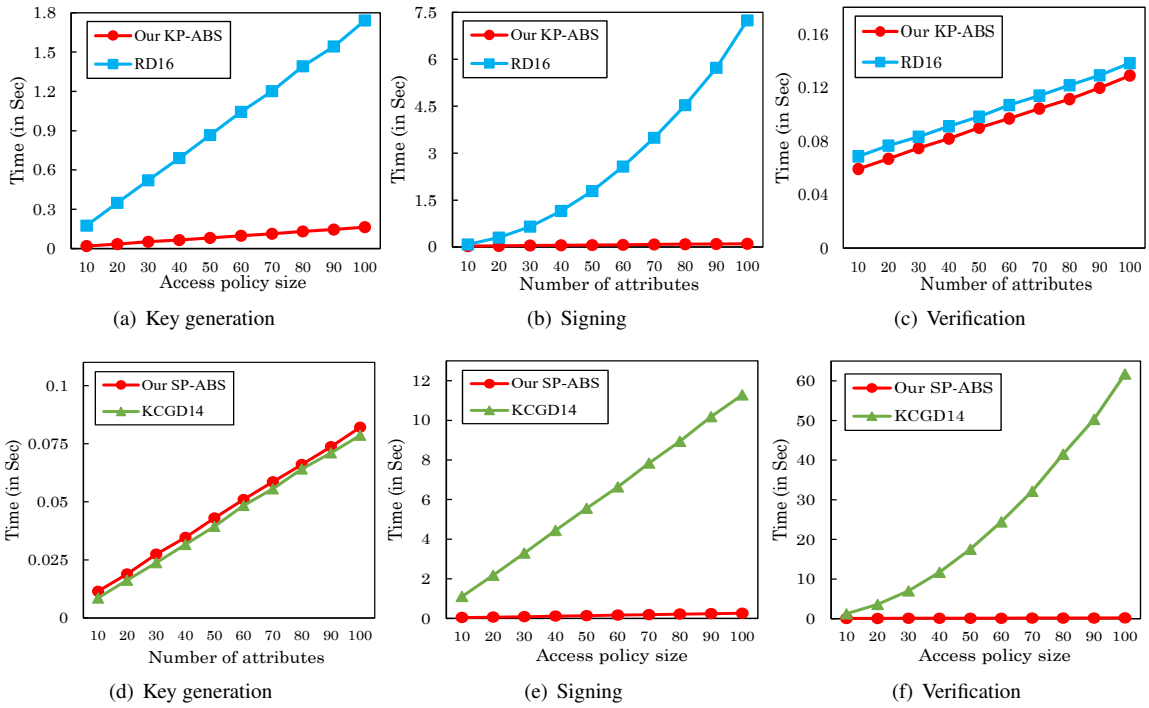
For SP-ABS, the schemes in MPR10 [40] lack sufficient concrete details, making implementation infeasible. The scheme in WZFY10 [41] is based on Type-II pairings, which we avoid because: (1) they are less efficient in practice due to costly operations in $\mathbb{G}_2$; (2) they limit protocol design since many modern assumptions are not realizable; and (3) they are incompatible with modern cryptographic libraries and pairing-friendly curves. KCGD14 [28] has key generation efficiency comparable to ours but slower asymptotic efficiency in signing and verification. Nonetheless, implementing it alongside our scheme highlights the concrete efficiency gap. Therefore, for SP-ABS we implemented our scheme and KCGD14 [28].

Experiment setting. In our experiments, we measure the running time in two types of parameter setting: (1) We fix the attribute set size to 10, and increase the policy size from 10, 20, ... to 100. (2) We fix the policy size as 100, and increase the size of attribute set from 10, 20, ..., 100. For the KP-ABS scheme in RD16, an extra parameter n is defined as the maximum bound size of the attribute set, in which we always set it equal to the size of the attribute set. For the SP-ABS scheme in KCGD14, we define an additional parameter $\mathcal{U}_s$ as the attribute universe, and set its size $|\mathcal{U}_s|$ to 100 to include all possible attributes in any attribute set or policy. To ensure that the attribute set always satisfies the policy, we form AND gates between attributes in the policy when the policy size is equal to the attribute set size. For the case where the policy size is larger than the attribute set size, we form the policy with one OR gate right after one subset of the policy is equal to the attribute set. For example, for an size-2 attribute set [A, B], a size-4 policy should be set to (A AND B) OR (C AND D). Besides, we convert an access policy into a Boolean formula and then to an MSP using Lewko-Waters' method [46] so that the matrix $\mathbf{M}$ has only entries in $\{0, 1, -1\}$ and the reconstruction coefficients $\{\gamma_i\}$ are always 1, which reduces the number of exponentiations for all ABS schemes implemented.

Table 3 lists the average time taken by various operations on BN254 and the setup times for the ABS schemes. Then we show the running times for the ABS schemes in Fig. 3. Table 4 and Table 5 compares the computational and storage overhead, respectively. The computational overhead lists the number of exponentiations, I2P operations and pairings. The storage overhead shows the size of the master public key (mpk), a secret key (sk), and a signature (σ).

Basic operations. According to the right-hand side of Table 3, we can see that the operations in group $\mathbb{G}_2$ are more expensive than on $\mathbb{G}_1$, in which it is nearly 1.3 times slower

Figure 3: Running times for KP-ABS (top) and SP-ABS (bottom) schemes



(a) Key generation (b) Signing (c) Verification

(d) Key generation (e) Signing (f) Verification

to choose an element, 0.6 times slower for exponentiation, 1.2 times slower for multiplication, and 1.3 times slower for I2P functions. Pairing is the most expensive operation that is nearly 32 and 20 times slower than exponentiation on $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. In addition, the size of an element in $\mathbb{G}_2$ is 1.5 times that of $\mathbb{G}_1$.

On the left-hand side of Table 3, we show the setup time of the schemes listed in our evaluation. Since both our ABS schemes support large universes and do not have to define parameters related to any attribute, both our schemes achieve a constant setup time. Despite RD16 claims it supports a large universe, the number of public parameters in their setup algorithm is linear to the bound n. Thus, when n is set to 100, their setup time (170 ms) is 6 times slower than our KP-ABS (28 ms). Since KCGD14 is a small universe scheme, when the attribute universe size $|\mathcal{U}_s|$ is set to 100, their setup time (275ms) is 9.5 times slower than our SP-ABS scheme.

KP-ABS. The running times and computational overhead for KP-ABE schemes are shown in the upper part of Fig. 3 and the upper part of Table 4 respectively. For key generation, when the policy contains 100 attributes, our scheme (0.16s) outperforms RD16 (1.74s). This is because our scheme has $2n_1$ exponentiations and n_1 I2P operations on $\mathbb{G}_1$, and only 1 exponentiation on $\mathbb{G}_2$. On the other hand, RD16 has $2nn_1$ exponentiations on $\mathbb{G}_1$ and n_1 exponentiations on $\mathbb{G}_2$. In these experiments, we measure the running time when the policy size increases, so the size of the attribute set and the maximum bound n in RD16 are fixed to 10. This makes RD16's $\mathbb{G}_1$ exponentiations equal to $20n_1$ instead of a superlinear complexity. Nevertheless, our scheme is still 10 times faster.

For signing, when the size of the attribute set is set to 100, RD16 runs 7.2s for signing, which is 72 times slower than our scheme (0.1s). This is because the signing time of RD16 is related to the product of the size of the attribute set $|I|$ and the maximum bound n. When $|I|$ increases, n must increase at least equal to $|I|$ since the maximum bound should at least equal to the number of attributes used in the signing algorithm. This makes RD16 presenting a superlinear blowup in the signing time when the size of attribute set is increasing, while our KP-ABS only needs a linear $3|I|+2$ exponentiations and $|I|$ I2P operations, as well as a small number of $\mathbb{G}_2$ and $\mathbb{G}_T$ exponentiations.

For verification, when the size of the attribute set is set to 100, RD16 requires 0.14s while our scheme is a little bit faster (0.13s). The reason is that, RD16 has $n+l$ $\mathbb{G}_1$ exponentiations, in which l is the output bit number of the hash function. In the charm library, the hash function maps the hash digest into a large prime-order subgroup (typically 256-bit), and the resulting decimal representation spans approximately 77 digits. Thus, RD16 requires $n+77$ $\mathbb{G}_1$ exponentiations[4] and needs 3 pairings. While our KP-ABS requires less $\mathbb{G}_1$ exponentiations ($|I|+2$) and only 2 pairings, it also needs $|I|$ I2P operations, and 2 $\mathbb{G}_T$ exponentiations. As a result, our scheme is faster than RD16 by a minor advantage.

[4]We let $n = |I|$ for the best case of RD16. In real applications, $n \geq |I|$.

Table 4: Computational overhead for key generation, signing and verification for KP-ABS (top) and SP-ABS schemes (bottom)

Schemes	KeyGen			Sign					Verify				
	$\mathbb{G}_1$		$\mathbb{G}_2$	$\mathbb{G}_1$		$\mathbb{G}_2$	$\mathbb{G}_T$	pairing	$\mathbb{G}_1$		$\mathbb{G}_2$	$\mathbb{G}_T$	pairing
	exp	I2P*	exp	exp	I2P	exp	exp		exp	I2P	exp	exp	
RD16 [21]	$2nn_1$	-	n_1	$(\lvert I\rvert+1)n+l-\lvert I\rvert+2$	-	$\lvert I\rvert+2$	-	-	$n+l$	-	-	-	3
Our KP-ABS	$2n_1$	n_1	1	$3\lvert I\rvert+2$	$\lvert I\rvert$	1	2	-	$\lvert I\rvert+2$	$\lvert I\rvert$	-	2	2
KCGD14 [28]	$\lvert\mathcal{S}\rvert+1$	-	-	$2n_1(n_2+1)+2\lvert I\rvert+4$	-	$6n_1+4$	$4n_1n_2$	$3n_1+1$	$n_1n_2+3n_1+n_2+3$	-	$n_1n_2+3+3n_1$	$(4n_1+1)n_2$	n_1n_2+2
Our SP-ABS	$\lvert\mathcal{S}\rvert+2$	$\lvert\mathcal{S}\rvert$	1	$3\lvert I\rvert+2n_1$	$\lvert I\rvert$	1	2	-	$2n_1+1$	n_1	-	2	2

Table 5: Comparison of storage overhead for KP-ABS schemes (top) and SP-ABE schemes (bottom)

Schemes	*mpk*			*sk*		σ		
	$\mathbb{G}_1$	$\mathbb{G}_2$	$\mathbb{G}_T$	$\mathbb{G}_1$	$\mathbb{G}_2$	$\mathbb{Z}_p$	$\mathbb{G}_1$	$\mathbb{G}_2$
RD16 [21]	$n+l+3$	1	1	nn_1	n_1	-	1	2
Our KP-ABS	1	1	1	n_1	1	$\lvert I\rvert+3$	2	1
KCGD14 [28]	$2\lvert\mathcal{U}_s\rvert+4$	$2\lvert\mathcal{U}_s\rvert+2$	-	$\lvert\mathcal{S}\rvert+1$	-	$8n_1+5$	$2n_1+n_2+1$	n_1+1
Our SP-ABS	2	1	1	$\lvert\mathcal{S}\rvert+1$	1	n_1+2	2	1

The storage overhead of the KP-ABS schemes is shown on the upper part of Table 5. For the master pubic key, RD16 has a linear number of $\mathbb{G}_1$ elements related to $n+l$ while our elements in $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are all constant 1.For secret key, RD16 has superlinear nn_1 size in $\mathbb{G}_1$ elements and n_1 size in $\mathbb{G}_2$ elements. Our scheme only has n_1 $\mathbb{G}_1$ elements and 1 $\mathbb{G}_2$ elements. For signature size, RD16 only needs constant 1 $\mathbb{G}_1$ element and 2 $\mathbb{G}_2$ elements in $\mathbb{G}_2$. Our scheme requires a linear $|I|+3$ elements in $\mathbb{Z}_p$ for a Schnorr-type signature. This shows that the design goal of RD16 is to achieve a constant signature size while we aim at faster running times.

SP-ABS. The running times and computational overhead for SP-ABS scheme are presented in the lower part of Fig. 3 and lower part of Table 4, respectively. For key generation, when the size of attribute set $|\mathcal{S}|$ is set to 100, our scheme runs comparable time (0.082s) as KCGD14 (0.078s). Our scheme is slightly slower, mainly because our scheme has one more exponent on $\mathbb{G}_2$. For signing, when the policy size is set to 100 and the size of the attribute subset $|I|$ is set to 10, our SP-ABS runs 0.26s, which is 43 times faster than KCGD14 (11.28s). This is because our scheme only has a linear number of exponentiations in $\mathbb{G}_1$ $(3|I|+2n_1)$ where KCGD14 has $O(n_1n_2)$ $\mathbb{G}_1, \mathbb{G}_T$ exponentiations and $3n_1+1$ pairings. For verification, our SP-ABS runs in 0.21s when the policy size is set to 100, which is 294 times faster than KCGD14 (61.7s). The reason is that our scheme only requires $2n_1+1$ exponentiations in $\mathbb{G}_1$, 2 exponentiations in $\mathbb{G}_T$, and 2 pairings, while KCGD14 requires $O(n_1n_2)$ operations on $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ and pairings.

The storage overhead of our SP-ABS is on the lower part of Table 5. For the master pubic key, our SP-ABS only needs constant number of elements (2 in $\mathbb{G}_1$, 1 in $\mathbb{G}_2$ and $\mathbb{G}_T$ each), while KCGD14 has $O(|\mathcal{U}_s|)$ elements in $\mathbb{G}_1, \mathbb{G}_2$ as their scheme is small universe. For secret key, KCGD14 only needs $|\mathcal{S}|+1$ elements, our scheme requires 1 more element in $\mathbb{G}_2$. For signature size, our scheme requires n_1+2 elements in $\mathbb{Z}_p$, 2 elements in $\mathbb{G}_1$, and 1 element in $\mathbb{G}_2$. KCGD14 has more elements than ours, $8n_1+5$ in $\mathbb{Z}_p$, $2n_1+n_2+1$ in $\mathbb{G}_1$, and n_1+1 in $\mathbb{G}_2$.

7 Conclusions

This paper proposed fast key-policy ABS and signature-policy ABS schemes. These schemes support expressive policies, large universes, and arbitrary attributes. Both KP-ABS and SP-ABS schemes demonstrate adaptive unforgeability under the CBDH assumption and anonymity under the DXDH assumption. As the first implementation of MSP-based ABS schemes, the results indicate that our schemes achieve overall the fastest running time compared to the state-of-the-art ABS schemes. On the other hand, the sizes of our signatures are linear to the attribute size because we use a Schnorr-type signature for anonymity. Considering that RD16 [21] ensures anonymity while keeping their signature size constant, it will be engaging in future work to replace the Schnorr signature with different techniques in our schemes for shortening the signature size, while keeping the advantage of fast operations.

Acknowledgments

This work is supported by the European Union's Horizon research and innovation program under grant agreement numbers: 101069688 (CONNECT), 101070627 (REWIRE), 101095634 (ENTRUST), 101167904 (CASTOR), and 101190366 (PiQASO). The first four projects are funded by the UK government's Horizon Europe guarantee and administered by UKRI. Additionally, C. C. Drăgan is partially supported by EPSRC grants EP/Y020529/1 and EP/W032473/1. M. Manulis acknowledges funding for Project LIONS under dtec.bw (via NextGenerationEU).

Ethical Considerations

We conducted a stakeholder-based ethics analysis following the Menlo Report principles.

Stakeholders and Impacts: Our research on attribute-based signatures (ABS) affects multiple stakeholder groups. As the research team, we bear responsibility for the correctness and security of our proposed methods. Future cryptography researchers who may analyze, implement, or extend our KP-ABS and SP-ABS schemes are primary stakeholders who will benefit from our efficiency improvements but could be

impacted by any undiscovered flaws in our constructions. Cryptographic standard bodies and developers implementing identity management systems represent indirect stakeholders who may incorporate our techniques. The broader cryptographic community is affected through the research directions and resource allocations our work may influence. Given the theoretical nature of our contribution, immediate risks are limited. Potential negative impacts include propagation of any undetected errors in our security proofs, computational resources consumed in verification or extension of our work, and opportunity costs if our approach proves less fruitful than alternative research directions.

Beneficence: Our research provides concrete benefits to the cryptographic community by introducing more efficient ABS schemes. Specifically, our KP-ABS and SP-ABS constructions demonstrate improvements over existing schemes in computational and communication complexity, potentially enabling practical deployment of ABS in real-world applications where previous schemes were too resource-intensive.

Respect for Persons: Our research involved no human subjects or personal data.

Justice: We ensure equitable distribution of our research benefits through open publication, making our advances accessible to researchers globally regardless of institutional resources or geographic location. Our efficiency improvements particularly benefit resource-constrained environments where previous ABS schemes were computationally prohibitive. We provide detailed complexity analyses and implementation guidance to reduce barriers to adoption.

Respect for Law and Public Interest: Our research advances the public interest in privacy-preserving authentication systems while complying with applicable regulations. ABS enables fine-grained access control and anonymous credentials, serving legitimate privacy needs in digital identity systems. As fundamental cryptographic research intended for publication, our work falls under academic research exemptions to export control regulations. We make no patent claims, ensuring these techniques remain freely available to the cybersecurity community.

Mitigations: We implemented several harm-reduction measures:

- Rigorous verification of all security proofs through multiple independent reviews by team members
- Comprehensive complexity analysis with concrete parameter comparisons to existing schemes
- Clear documentation of security assumptions and implementation considerations
- Explicit notation of any hardness assumptions

Unmitigated Risks: We acknowledge that researchers implementing or extending our work will need to invest computational resources for verification. Additionally, subtle implementation challenges inherent to attribute-based cryptography remain, despite our efforts to provide clear guidance.

Moreover, if a real-world application is concerned that a centralized attribute authority could become a bottleneck for exclusion of users from services, or that aggregated logs of signed attributes could enable covert surveillance of groups sharing those attributes, the centralized attribute authority in the proposed ABS schemes can be replaced with decentralized or distributed attribute authorities. This can be achieved using threshold cryptography or distributed key generation as a safeguard against a single point of failure. During verification and auditing, privacy-enhancing technologies such as zero-knowledge proofs can be integrated into the decentralized design.

Decision: After careful consideration of the benefits and risks, our efficiency improvements represent meaningful progress toward practical ABS research. The theoretical advances are rigorously verified, and open publication ensures broad benefit to the cryptographic community.

Open Science

We embrace open science principles to ensure reproducibility and facilitate future research on attribute-based signatures.

Code Release: During submission, we will release complete implementations of both our KP-ABS and SP-ABS schemes at [109], including:

- Optimized implementations with extensive documentation
- Benchmarking framework comparisons against prior ABS schemes

License: All code and implementations will be released under the MIT License, allowing unrestricted use, modification, and distribution. Documentation will be available under Creative Commons CC-BY 4.0. We make no patent claims on our ABS constructions.

Reproducibility: All experiments can be reproduced using our provided code.

References

[1] S. F. Shahandashti and R. Safavi-Naini, "Threshold attribute-based signatures and their application to anonymous credential systems," in AFRICACRYPT, 2009.

[2] H. K. Maji, M. Prabhakaran, and M. Rosulek, "Attribute-based signatures," in CT-RSA, 2011.

[3] Z. Xu and K. M. Martin, "Anonymous user revocation for using attribute-based signature in cloud computing," in IEEE International Conference on Cloud Computing Technology and Science, 2014.

[4] R. Ch, "The use of attribute-based signature in a blockchain for health records system storage," Journal of Computer Engineering & Information Technology, 2021.

[5] L. Liu et al., "A revocable and comparable attribute-based signature scheme from lattices for IoMT," Journal of Systems Architecture, 2024.

[6] H. Hong, B. Hu, and Z. Sun, "An efficient and secure attribute-based online/offline signature scheme for mobile crowdsensing," Humman-Centric Computing and Information Sciences, 2021.

[7] B. Chen et al., "Efficient attribute-based signature with collusion resistance for internet of vehicles," IEEE Transactions on Vehicular Technology, 2023.

[8] L. Chen and R. Urian, "DAA-A: Direct anonymous attestation with attributes," in TRUST, 2015.

[9] S. K. Ramani et al., "NDN-ABS: Attribute-based signature scheme for named data networking," in The ACM Conference on Information-Centric Networking, 2019.

[10] B. Hampiholi et al., "Towards practical attribute-based signatures," in Security, Privacy, and Applied Cryptography Engineering, 2015.

[11] W. Xiangyu et al., "Dynamic fine-grained access control for smart contracts based on improved attribute-based signature," The Journal of Supercomputing, 2025.

[12] Security Team I.R.Z., "Specification of the identity mixer cryptographic library." 2012.

[13] C. Paquin and G. Zaverucha, "U-Prove cryptographic specification." 2023.

[14] ISO/IEC WD 24843. Information security — Attribute-Based Credentials.

[15] N. Kaaniche and M. Laurent, "Attribute-based signatures for supporting anonymous certification," in ESORICS, 2016.

[16] Microsoft, "What is Azure attribute-based access control (Azure ABAC)?." https://learn.microsoft.com/en-us/azure/role-based-access-control/conditions-overview.

[17] Amazon Web Services, "AWS identity and access management." https://docs.aws.amazon.com/pdfs/IAM/latest/UserGuide/iam-ug.pdf#introduction_attribute-based-access-control.

[18] NIST, "Guide to attribute based access control (ABAC) definition and considerations." NIST Special Publication 800-162.

[19] J. Li et al., "Attribute-based signature and its applications," in ASIACCS, 2010.

[20] J. Li and K. Kim, "Attribute-based ring signatures." Cryptology ePrint Archive, 2008.

[21] Y. S. Rao and R. Dutta, "Efficient attribute-based signature and signcryption realizing expressive access structures," International Journal of Information Security, 2016.

[22] P. Datta, R. Dutta, and S. Mukhopadhyay, "Short attribute-based signatures for arbitrary turing machines from standard assumptions," Designs, Codes and Cryptography, 2023.

[23] H. Maji, M. Prabhakaran, and M. Rosulek, "Attribute-based signatures: Achieving attribute-privacy and collusion-resistance." Cryptology ePrint Archive, 2008.

[24] T. Okamoto and K. Takashima, "Efficient attribute-based signatures for non-monotone predicates in the standard model," IEEE Transactions on Cloud Computing, 2014.

[25] R. Hayashi, Y. Sakai, and S. Yamada, "Attribute-based signatures for circuits with optimal parameter size from standard assumptions." Cryptology ePrint Archive, 2024.

[26] H. Anada, S. Arita, and K. Sakurai, "Attribute-based signatures without pairings via the fiat-shamir paradigm," in ASIAPKC, 2014.

[27] J. Herranz, "Attribute-based signatures from RSA," Theoretical Computer Science, 2014.

[28] A. El Kaafarani et al., "Attribute-based signatures with user-controlled linkability," in CANS, 2014.

[29] M. Gagné, S. Narayan, and R. Safavi-Naini, "Short pairing-efficient threshold-attribute-based signature," in International conference on pairing-based cryptography, 2012.

[30] G. Lin et al., "F2P-ABS: A fast and secure attribute-based signature for mobile platforms," Security and Communication Networks, 2019.

[31] X. P. Mao et al., "Attribute-based signature on lattices," Journal of Shanghai Jiaotong University, 2014.

[32] F. Luo and S. Al-Kuwari, "Attribute-based signatures from lattices: Unbounded attributes and semi-adaptive security," Designs, Codes and Cryptography, 2022.

[33] J. Dey and R. Dutta, "Privacy enhanced secure compact attribute-based signature from MQ problem for monotone span program," Theoretical Computer Science, 2024.

[34] S. Ling et al., "Fully dynamic attribute-based signatures for circuits from codes," in PKC, 2024.

[35] IETF, "Terminology for post-quantum traditional hybrid schemes," 2025.

[36] L. Chen et al., "FABS: Fast attribute-based signatures." Cryptology ePrint Archive, Paper 2026/022, 2026.

[37] J. Herranz et al., "Short attribute-based signatures for threshold predicates," in CT-RSA, 2012.

[38] Y. Sakai, N. Attrapadung, and G. Hanaoka, "Attribute-based signatures for circuits from bilinear map," in PKC, 2016.

[39] Y. Sakai et al., "Attribute-based signatures for unbounded languages from standard assumptions," in ASIACRYPT, 2018.

[40] H. K. Maji, M. Prabhakaran, and M. Rosulek, "Attribute-based signatures." Cryptology ePrint Archive, 2010.

[41] H.-X. Wang et al., "Attribute-based signature with policy-and-endorsement mechanism," Journal of Computer Science and Technology, 2010.

[42] L. Meng et al., "FABESA: Fast (and anonymous) attribute-based encryption under standard assumption," in ACM CCS, 2024.

[43] Y. Rouselakis and B. Waters, "Practical constructions and new proof methods for large universe attribute-based encryption," in ACM CCS, 2013.

[44] S. D. Galbraith, K. G. Paterson, and N. P. Smart, "Pairings for cryptographers," Discrete Applied Mathematics, 2008.

[45] A. Beimel, "Secure schemes for secret sharing and key distribution," PhD thesis, Israel Institute of Technology, Technion, 1996.

[46] A. Lewko and B. Waters, "Unbounded HIBE and Attribute-Based Encryption," in EUROCRYPT, 2011.

[47] D. Riepel and H. Wee, "FABEO: Fast attribute-based encryption with optimal security," in ACM CCS, 2022.

[48] C. P. Schnorr, "Efficient identification and signatures for smart cards," in Crypto – 89, 1990.

[49] T. Okamoto, "Efficent attribute-based signatures for non-monotone predicates in the standard model," in PKC, 2011.

[50] S. F. Shahandashti and R. Safavi-Naini, "Threshold attribute-based signatures and their application to anonymous credential systems," in AFRICACRYPT, 2009.

[51] F. Tang, H. Li, and B. Liang, "Attribute-based signatures for circuits from multilinear maps," in ISC, 2014.

[52] Y. Sakai, "Succinct attribute-based signatures for bounded-size circuits by combining algebraic and arithmetic proofs," in SCN, 2022.

[53] P. Datta, R. Dutta, and S. Mukhopadhyay, "Attribute-based signatures for turing machines.," IACR Cryptol. ePrint Arch., 2017.

[54] M. Ishizaka and K. Fukushima, "Attribute-based signatures for range of inner product and its applications," in ICISC, 2022.

[55] M. Ishizaka, "Key-range attribute-based signatures for range of inner product and its applications," in ProvSec, 2023.

[56] P. Datta, T. Okamoto, and K. Takashima, "Efficient attribute-based signatures for unbounded arithmetic branching programs," in PKC, 2019.

[57] S. Kumar et al., "Attribute based signatures for bounded multi-level threshold circuits," in EuroPKI, 2011.

[58] S. Rani and S. T. Ali, "Expressive key-policy attribute-based constant-size signature," in ISEA Asia Security and Privacy, 2017.

[59] T. Okamoto and K. Takashima, "Efficient attribute-based signatures for non-monotone predicates in the standard model," in PKC, 2011.

[60] P. Yang et al., "Efficient and expressive fully secure attribute-based signature in the standard model," in Australian Information Security Management Conference, 2011.

[61] A. Escala, J. Herranz, and P. Morillo, "Revocable attribute-based signatures with adaptive security in the standard model," in AFRICACRYPT, 2011.

[62] D. Cao et al., "An expressive attribute-based signature scheme without random oracles," in International Conference on Computer Application and System Modeling, 2012.

[63] C. I. Fan et al., "Attribute-based strong designated-verifier signature scheme," Journal of Systems and Software, 2012.

[64] D. Cao et al., "Mediated attribute based signature scheme supporting key revocation," in ICIDT, 2012.

[65] C. Chen et al., "Fully secure attribute-based systems with short ciphertexts/signatures and threshold access structures," in CT-RSA, 2013.

[66] J. Chen et al., "Attribute-based key-insulated signature and its applications," Information Sciences, 2014.

[67] K. Wang et al., "Attribute-based signature with message recovery," in ISPEC, 2014.

[68] S. Ding, Y. Zhao, and Y. Liu, "Efficient traceable attribute-based signature," in TrustCom, 2014.

[69] J. Su et al., "ePASS: An expressive attribute-based signature scheme with privacy and an unforgeability guarantee for the internet of things," Future Generation Computer Systems, 2014.

[70] F. Cai, W. Guo, and X. Liu, "Threshold attribute based universal designated verifier signature scheme in the standard model," WSEAS Transaction on Communications, 2014.

[71] J. Wei, W. Liu, and X. Hu, "Forward-secure threshold attribute-based signature scheme," The Computer Journal, 2015.

[72] J. Wei et al., "Revocable threshold attribute-based signature against signing key exposure," in ISPEC, 2015.

[73] Y. Ren et al., "Attribute–based signature schemes with accountability," International Journal of Information and Communication Technology, 2015.

[74] M. Urquidi et al., "Attribute-based signatures with controllable linkability," in INTRUST, 2016.

[75] J. Wei et al., "Practical attribute-based signature: Traceability and revocability," The Computer Journal, 2016.

[76] A. El Kaafarani and E. Ghadafi, "Attribute-based signatures with user-controlled linkability without random oracles," in Cryptography and Coding: IMA International Conference, 2017.

[77] K. Gu et al., "Efficient and secure attribute-based signature for monotone predicates," Acta Informatica, 2017.

[78] K. Gu, K. Wang, and L. Yang, "Traceable attribute-based signature," Journal of Information Security and Applications, 2019.

[79] S. T. Ali, "Traceable and verifier-local revocable attribute-based signature with constant length," in Security and Privacy, 2021.

[80] H. Chen et al., "SPCABS: Signature-policy comparable attribute-based signatures," Security and Communication Networks, 2022.

[81] H. Anada, K. Anzai, and M. Fukumitsu, "Attribute-based signatures of Fiat-shamir type in bilinear groups: Scheme and performance," in International Conference on Platform Technology and Service, 2022.

[82] Z. Kang et al., "TFS-ABS: Traceable and forward-secure attribute-based signature scheme with constant-size," IEEE Transactions on Knowledge and Data Engineering, 2023.

[83] S. Galbraith, "New discrete logarithm records, and the death of type 1 pairings," 2014.

[84] A. Joux, "A new index calculus algorithm with complexity in small characteristic," in SAC, 2013.

[85] J. Li and K. Kim, "Hidden attribute-based signatures without anonymity revocation," Information Sciences, 2010.

[86] T. H. Yuen et al., "Forward secure attribute-based signatures," in ICICS, 2012.

[87] H. Cui et al., "Escrow free attribute-based signature with self-revealability," Information Sciences, 2016.

[88] J. Zhang, J. Chen, and W. Meng, "Efficient attribute-based signature for monotone predicates," in ProvSec, 2021.

[89] Y. Zhang et al., "Registered attribute-based signature," in PKC, 2024.

[90] T. Okamoto and K. Takashima, "Decentralized attribute-based signatures," in PKC, 2013.

[91] Y. Sun et al., "A decentralizing attribute-based signature for healthcare blockchain," in International conference on computer communication and networks, 2018.

[92] E. Ghadafi, "Stronger security notions for decentralized traceable attribute-based signatures and more efficient constructions," in CT-RSA, 2015.

[93] Q. Tao, X. Cui, and A. Iftekhar, "A novel lightweight decentralized attribute-based signature scheme for social co-governance," Information Sciences, 2024.

[94] J. Sun et al., "Outsourced decentralized multi-authority attribute based signature and its application in IoT," IEEE Transactions on Cloud Computing, 2019.

[95] T. Okamoto and K. Takashima, "Decentralized attribute-based encryption and signatures," IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, 2020.

[96] R. Guo et al., "Secure attribute-based signature scheme with multiple authorities for blockchain in electronic health records systems," IEEE Access, 2018.

[97] J. Sun, J. Qin, and J. Ma, "Securely outsourcing decentralized multi-authority attribute based signature," in Cyberspace Safety and Security International Symposium, 2017.

[98] Y. Chen et al., "Efficient attribute based server-aided verification signature," IEEE Transactions on Services Computing, 2021.

[99] J. Li et al., "Decentralized attribute-based server-aid signature in the internet of things," IEEE Internet of Things Journal, 2021.

[100] Z. Huang et al., "Secure outsourced attribute-based signatures with perfect anonymity in the standard model," Security and Communication Networks, 2021.

[101] Y. Wang et al., "Attribute-based user revocable data integrity audit for internet-of-things devices in cloud storage," Security and Communication Networks, 2020.

[102] C. Huang et al., "Toward security as a service: A trusted cloud service architecture with policy customization," Journal of Parallel and Distributed Computing, 2021.

[103] F. Luo et al., "Attribute-based proxy re-signature from standard lattices and its applications," Computer Standards & Interfaces, 2021.

[104] Z. Huang et al., "Outsourced attribute-based signatures with perfect privacy for circuits in cloud computing," Concurrency and Computation: Practice and Experience, 2021.

[105] D. Boneh and X. Boyen, "Short signatures without random oracles," in EUROCRYPT, 2004.

[106] D. Pointcheval and J. Stern, "Security arguments for digital signatures and blind signatures," Journal of cryptology, 2000.

[107] M. Bellare and G. Neven, "Multi-signatures in the plain public-key model and a general forking lemma," in ACM CCS, 2006.

[108] J. A. Akinyele et al., "Charm: a framework for rapidly prototyping cryptosystems," Journal of Cryptographic Engineering, 2013.

[109] "FABS: Fast Attribute-Based Signatures." https://github.com/Anonymous-Mystery/FABS, 2025.

A RD16 [21] KP-ABS with Type-III pairings

$(\mathsf{mpk},\ \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$.

Run $\mathsf{GroupGen}(1^\lambda)$ to obtain the group parameters $\mathsf{par} = (p,\ \mathbb{G}_1,\ \mathbb{G}_2,\ \mathbb{G}_T,\ g_1,\ g_2,\ e)$. Pick $\alpha \leftarrow_\$ \mathbb{Z}_p^*$ and a hash function $\mathcal{H}: \{0,\ 1\}^* \rightarrow \{0,1\}^l$. Let n be a maximum bound on the size of signing attribute set used in signing algorithm.

Pick $V_0, V_1, ..., V_n, u_0, u_1, ..., u_l \leftarrow_\$ \mathbb{G}_1$, output

$\mathsf{mpk} = (\mathcal{H},\ \mathsf{par},\ Y = e(g_1,g_2)^\alpha, V_0, ..., V_n, u_0, ..., u_l),\ \mathsf{msk} = \alpha$.

$\mathsf{sk} \leftarrow \mathsf{KeyGen}(\mathsf{mpk},\ \mathsf{msk},\ (\mathbf{M},\ \pi,\ \{\pi(i)\}_{i\in[n_1]}))$.

Pick $\{r_i\}_{i\in[n_1]} \leftarrow_\$ \mathbb{Z}_p$, $\mathbf{v} \leftarrow_\$ \mathbb{Z}_p^{n_2-1}$, Compute

$$D_i = g_1^{\mathbf{M}_i(\alpha||\mathbf{v})^\top} V_0^{r_i}, D_i^{'} = g_2^{r_i}$$
$$D_i^{''} = \{D_{i,x}^{''} : D_{i,x}^{''} = (V_1^{-\rho(i)^{x-1}} V_x)^{r_i}, \forall x = 2, ..., n\}$$

Output $\mathsf{sk} = ((\mathbf{M},\ \pi,\ \{\pi(i)\}_{i\in[n_1]}), \{D_i, D_i^{'}, D_i^{''}\}_{i\in[n_1]})$.

$\sigma \leftarrow \mathsf{Sign}(\mathsf{mpk}, \mathsf{sk}, W = \{\pi(i)_i\}_{i\in I}, \mathsf{msg})$.

Find $\{\gamma_i\}_{i\in I}$ s.t. $\sum_{i\in I}\gamma_i\mathbf{M}_i = (1,\ 0,\ \ldots,\ 0)$.

Compute $(y_1, y_2, ..., y_n)$ s.t. $P_W(X) = \sum_{j=1}^{|W|+1} y_j \cdot X^{j-1}$. If $|W|+1 < n$, set $y_{|W|+2} = \cdots = y_n = 0$. So, $P_W(X) = \sum_{j=1}^{n} y_j \cdot X^{j-1}$.

Pick $\theta, \varepsilon \leftarrow_\$ \mathbb{Z}_p$, compute

$\sigma_1 = g_2^\theta, \sigma_2 = g_2^\varepsilon \prod_{i\in I}(D_i^{'})^{\gamma_i}, \mathcal{H}(\mathsf{msg}||\sigma_2||W) = (m_1, m_2, ..., m_l)$,

$$\sigma_3 = \left(\prod_{i\in I}\left(D_i \prod_{x=2}^{n}(D_{i,x}^{''})^{y_x}\right)^{\gamma_i}\right) \cdot \left(V_0 \prod_{k\in[n]} V_k^{y_k}\right)^{\varepsilon} \cdot \left(u_0 \prod_{j\in[l]} u_j^{m_j}\right)^{\theta}$$

Output $\sigma := (\sigma_1, \sigma_2, \sigma_3)$.

$0/1 \leftarrow \mathsf{Verify}(\mathsf{mpk}, \sigma, W = \{\pi(i)\}_{i\in I}, \mathsf{msg})$.

Compute $\mathcal{H}(\mathsf{msg}||\sigma_2||W) = (m_1, m_2, ..., m_l) \in \{0,1\}^l$

Compute $(y_1, y_2, ..., y_n)$ s.t. $P_W(X) = \sum_{j=1}^{n} y_j \cdot X^{j-1}$ as above.

Verify if $e(\sigma_3, g_2) = Y \cdot e(V_0 \prod_{k\in[n]} V_k^{y_k}, \sigma_2) \cdot e(u_0 \prod_{j\in[l]} u_j^{m_j}, \sigma_1)$.

If yes, output 1; otherwise, output 0.

Figure 4: The large universe KP-ABS in RD16 [21] in the Type-III pairing