Bilateral Access Control ABE from Lattices and Application to AB-ME

Huige Wang[1], Kefei Chen[1], Shuai Han[2], Jian Weng[3], and Qi Xie[1]

[1] Hangzhou Normal University, Hangzhou 311121, China
[2] Shanghai Jiao Tong University, Shanghai 200240, China
[3] Jinan University, Guangzhou, 510632, China
{whgexf,kfchen,qixie,dalen17,cryptjweng}@hznu.edu.cn,sjtu.edu.cn,gmail.com

Abstract. Attribute-Based Encryption (ABE) is a cryptosystem that can provide flexible access control, which has important application value in the field of fine-grained access control to encrypted data. However, current ABEs only support unilateral access control from the sender to the receiver, and fail to provide data source authentication, which violate the principle of data source identification for sensitive information as mandated by the General Data Protection Regulation (GDPR). To address this problem, the notion of bilateral access control Attribute-Based Encryption (bilateral access control ABE) is introduced, but the existing schemes are constructed under the classical assumption. In order to resist the quantum attacks, in this paper, we propose a post-quantum secure bilateral access control ABE scheme based on the Learning With Errors (LWE) and Short Integer Solution (SIS) assumptions on lattices. In comparison to the existing ABEs based on lattice assumptions, our approach not only enables access control from sender to receiver, but also from receiver to sender. Particularly, our scheme also captures the data source authentication. Furthermore, we prove that our scheme achieves selective Indistinguishable Chosen-Plaintext-Attacks (sIND-CPA) security and authentication under the LWE and SIS assumptions. Beyond that, we also capture the attribute-hiding property to obtain a post-quantum secure Attribute-Based Matchmaking Encryption (AB-ME) scheme by using a lockable-obfuscation-based transformation. Compared to the existing AB-ME schemes, our scheme not only achieves privacy and authentication, but also can resist quantum attacks.

1 Introduction

Attribute-Based Encryption (ABE) [1], initially proposed by Sahai and Waters [2], is a cryptosystem that can support flexible access control on users to encrypted data. ABE is a generalization of Identity-Based Encryption (IBE) by extending the user's "identity" to expressive and flexible attributes rather than a single string. ABE is widely used in many applications, such as cloud computing, big data and Internet of Things (IoT) due to its flexible access control on users to encrypted data. According to the policy-embedded mode, ABE is classified into Ciphertext-Policy ABE (CP-ABE) and Key-Policy ABE (KP-ABE).

For CP-ABE, secret key is associated with user's attributes, and ciphertext is bounded with access control policy. For KP-ABE, secret key is associated with access control policy, and ciphertext is connected to user's attributes.

Bilateral Access Control Attribute-Based Encryption. In contrast to traditional Attribute-Based Encryption (ABE), which only permits sender to impose access control on receivers, bilateral access control ABE empowers both the sender and the receiver to define access control policies for each other. Given ciphertext, message can be recovered with decryption key when the attributes of the sender and receiver match with their respective access control policy. When decryption fails, the encrypted message is invisible to the adversary. Bilateral access control ABE not only realizes data confidentiality, but also achieves authentication. This capability aligns with the principle of data source identification for sensitive information as required by the General Data Protection Regulation (GDPR) [3]. For this, bilateral access control ABE attracts widespread attention among researchers who, based on different assumptions, have proposed various bilateral access control ABE schemes in a variety of applications. For example, Li et al. [4] used the bilinear pairing to present an efficient privacy-preserving bilateral friendly ABE scheme in mobile social networks. Based on the non-standard q-1 assumption, Xu et al. [5] proposed a server-aided bilateral access control ABE and designed a dynamic user group data sharing system within the Internet of Things (IoT) environment. Furthermore, Xu et al. [6] presented a fine-grained bilateral access control data sharing system in the cloud computing environment using their proposed ABE scheme based on the standard Decisional Bilinear Diffie-Hellman (DBDH) and Computational Diffie-Hellman (CDH) assumptions. Additionally, a lightweight bilateral access control ABE [7] is introduced and used for designing data sharing system in the cloud IoT applications.

AB-ME [15, 16] is similar to bilateral access control ABE except that the former is additionally required to satisfy user-privacy (i.e., the sender attributes and receiver attributes are also hidden in the ciphertext). More specifically, the security guarantees of a standard AB-ME scheme should satisfy the following properties: If a mismatch happens, nothing is leaked except the fact that a match did not occur; If a match happens, nothing is leaked except for the encrypted message and the fact that a match occurred; A valid ciphertext can not be forged under an unauthorized encryption key (not generated by the key generation center), i.e, the AB-ME should satisfy authentication security. The weak security of AB-ME is the same as the above security requirement except that when a match occurs, not only the encrypted message is allowed to leak, but the sender and the receiver's attributes are also allowed to leak. That is, weak security only guarantees the user's privacy when match does not occur. Signcryption is a cryptosystem that combines digital signature and encryption, primarily designed to ensure data authenticity, integrity, and non-repudiation. While bilateral access control ABE shares similarities with signcryption, its core purpose is to address the challenge of biliteral matchmaking–i.e., enabling both communicating parties to enforce access control over private data simultaneously. In contrast, Attribute-Based Matchmaking Encryption (AB-ME) not only fulfills the functionalities of bilateral access control ABE but also provides additional protection for user-identity privacy.

Due to these characteristics, matchmaking encryption holds significant application value in bilateral access control contexts. For example, in mobile client advertising push scenarios, certain merchants want to send private service advertisements to specific users, while those specific users only wish to receive advertisements from particular merchants. Using matchmaking encryption, this private service advertisement push functionality can be effectively implemented. Specifically, merchants (i.e., data providers) first encrypt private service advertisement data along with designated target policies using attribute-related encryption keys and push it out. On the receiving end, clients (i.e., advertisement recipients) decrypt the received ciphertext under their own specified target policies using attribute-related decryption keys. When the attributes of both the client and the merchant match each other's specified target policies, the decryption process correctly recovers the private service advertisement information. Another example involves a CIA agent from the Central Intelligence Agency encrypting an email to specify that only FBI agents residing in New York City (NYC) can read his intelligence. Simultaneously, FBI agents in NYC can set their decryption policies to receive intelligence exclusively from CIA agents. Similarly, matchmaking encryption can be applied in bidding systems: a bidder can encrypt their private bid information by specifying which collectors are authorized to view it, while collectors can define decryption rules to access bids only from designated bidders. As in ABE, according to policy-embedded mode, AB-ME can be classified into key-policy AB-ME and ciphertext-policy AB-ME, where in the former, the receiver specified policies are embedded into the decryption key and the sender specified attributes are encrypted into the ciphertext, while in the latter case, it is the opposite.

Motivations and Challenges. Although bilateral access control ABE has got widely developed, the existing schemes rely on classical number-theoretical assumptions, which are vulnerable to quantum attacks. Among all known assumptions, the lattice-based assumptions are considered currently the most efficient for designing cryptographic algorithms that can resist quantum attacks. Based on this, in this paper, we aim to leveraging the Learning With Errors (LWE) and Short Integer Solution (SIS) assumptions over lattices to develop bilateral access control ABE scheme that can resist quantum attacks. To design such schemes, there are at least two challenges. On the one hand, it is necessary to consider how to establish the effective link between the master secret key and user's attributes during the encryption key generation and the effective link between the encryption key and the encrypted message during encryption to realize the data confidentiality and the source authentication. On the other hand, it is necessary to consider how to match the sender access policy specified by the receiver in the decryption with the sender attribute embedded in the ciphertext, and the receiver attribute specified by the sender during the encryption with the access policy embedded in the receiver private key to recover the encrypted message from the ciphertext. In comparison, designing a ciphertext-policy bilateral access control ABE scheme will be a larger challenge. This is because, in such scheme, we should not only consider how to associate the receivers' attributes with their decryption private key, but also how to match the receivers' attributes with their access control policy specified by the sender during the encryption. However, how to link the receivers' attributes with their private key through the master secret key (i.e., lattice trapdoor), and how to embed a general access control policy specified by the sender during encryption into the ciphertext remain a challenge. Based on this, we will focus on researching efficient construction of lattice-based key-policy bilateral access control ABE scheme, then employing universal circuit technique to transform it into a ciphertext-policy bilateral access control ABE scheme.

Overview of Definition for Key-Policy Bilateral Access Control ABE. Roughly speaking, a bilateral access control ABE scheme consists of six probabilistic polynomial time (ppt) algorithms. The setup algorithm, run by the Key Generation Center (KGC), takes as input a security parameter 1^λ, the maximum length of the attributes ℓ and the maximum length of encrypted message $N \in \mathbb{N}$, and outputs a public parameter pp and a master secret key msk. The sender encryption key generation algorithm, run by the KGC, takes as input the public parameter pp, the master secret key msk and a sender's attribute $satt$ and returns an attribute-related sender encryption key ek_{satt}. The receiver private key generation algorithm run by the KGC, takes as input the public parameter pp, the master secret key msk and a receiver's access control policy f_r, and outputs a policy-related receiver private key rk_{f_r}. The policy key generation algorithm, run by the KGC, takes as input the public parameter pp, the master secret key msk and an access control policy f_s (specified by the receiver), and returns a policy key pok_{f_s}. The encryption algorithm, run by a sender, takes as input the public parameter pp, the sender encryption key ek_{satt}, a message μ and a specified receiver attribute $ratt$, and outputs a ciphertext ct. The decryption algorithm run by a receiver, takes as input the public parameter pp, the receiver's private key rk_{f_r}, a policy key pok_{f_s} and a ciphertext $(ct, ratt, satt)$, and returns a message μ or $\perp$. The correctness requires that if the ciphertext is generated according to the definition, the encrypted message can be recovered with overwhelming probability when the sender attribute $satt$ and the receiver attribute $ratt$ match their policies f_s and f_r respectively. To better understand the above definition, we show the system model in Fig. 1.

The security of bilateral access control ABE requires to satisfy (selective) IND-CPA and authentication securities. The (selective) IND-CPA security guarantees the confidentiality of encrypted data, and the authentication ensures the authentication of data source. More specifically, the former captures that any ppt adversary cannot distinguish the encryptions of any two equal-length messages chosen by the adversary adaptively when allowing the adversary to make polynomial times of sender encryption key queries, receiver private key queries and policy-key queries subject to the constraints that the policies are not satisfied by their respective attributes. The latter captures that a valid ciphertext can not be forged under a sender encryption key that is not authorized by the

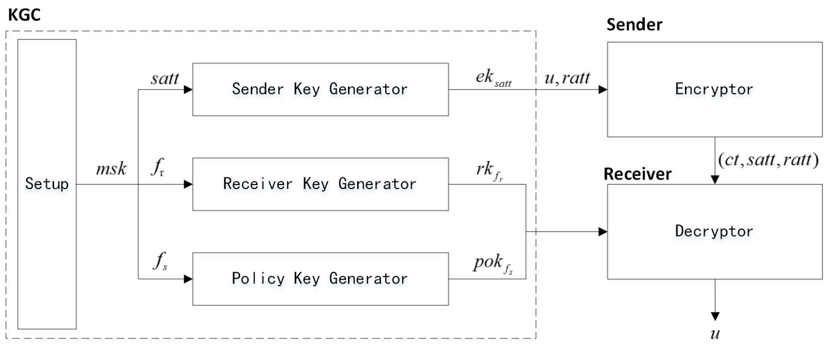

Fig. 1. Key-policy bilateral access control ABE system model

KGC when allowing the adversary to also make polynomial times of queries as the former.

Our Contributions and Techniques. In this paper, we propose a key-policy bilateral access control ABE scheme based on the LWE and SIS assumptions. We emphasize that this is the first lattice-based bilateral access control ABE scheme for generic policy representable by constant-depth circuits. Furthermore, using the transformation technique [32], the proposed bilateral access control ABE scheme can be transformed into one with attribute-hiding, resulting in the first Attribute-Based Matchmaking Encryption (AB-ME) from lattice assumptions. Now, we outline the construction of the proposed scheme. First, we run two times of lattice trapdoor generation algorithm $\mathsf{TrapGen}$ to generate two lattice trapdoors $\mathbf{T}_{\mathbf{A}}$, $\mathbf{T}_{\mathbf{A}'}$ as master secret key, where $\mathbf{T}_{\mathbf{A}}$, $\mathbf{T}_{\mathbf{A}'}$ are respectively used to compute the policy-related keys (rk_{f_r}, pok_{f_s}) and the attribute-related key ek_{satt}. More precisely, the receiver private key $rk_{f_r} = \{\mathbf{r}^i_{f_r}\}_{i\in[N]}$ consists of N low-norm vectors $\mathbf{r}^i_{f_r}$, which is generated using the left sampling algorithm $\mathbf{r}^i_{f_r} \leftarrow \mathsf{SampleLeft}\,(\mathbf{A}, \mathbf{B}_{f_r}, \mathbf{T}_{\mathbf{A}}, \mathbf{u}_{1,i}, \delta)$ under the master secret key component $\mathbf{T}_{\mathbf{A}}$ such that $(\mathbf{A}|\mathbf{B}_{f_r}) \cdot \mathbf{r}^i_{f_r} = \mathbf{u}_{1,i}$ for each $i \in [N]$. Similarly, the sender-policy-related policy key component $\mathbf{r}^i_{f_s}$ in the policy key $pok_{f_s} = \{\mathbf{r}^i_{f_s}\}_{i\in[N]}$ is also computed using the left sampling algorithm $\mathbf{r}^i_{f_s} = \mathsf{SampleLeft}\,(\mathbf{A}, \mathbf{D}_{f_s}, \mathbf{T}_{\mathbf{A}}, \mathbf{u}_{2,i}, \delta)$ under the key $\mathbf{T}_{\mathbf{A}}$ such that $(\mathbf{A}|\mathbf{D}_{f_s}) \cdot \mathbf{r}^i_{f_s} = \mathbf{u}_{2,i}$, where $\mathbf{u}_{2,i} = \mathbf{u}_i - \mathbf{u}_{1,i}$. Note that the homomorphic matrices $\mathbf{B}_{f_r}$, $\mathbf{D}_{f_s}$ are computed using the matrix embedding technique via $\mathbf{B}_{f_r} = \mathsf{Eval}_{pp}\,(f_r, (\mathbf{B}_1, \cdots, \mathbf{B}_\ell))$, $\mathbf{D}_{f_s} = \mathsf{Eval}_{pp}(f_s, (\mathbf{D}_1, \cdots, \mathbf{D}_\ell))$. In particular, for each $i \in [N]$, the above computations satisfy

$$(\mathbf{A}|\mathbf{B}_{f_r}) \cdot \mathbf{r}^i_{f_r} + (\mathbf{A}|\mathbf{D}_{f_s}) \cdot \mathbf{r}^i_{f_s} = \mathbf{u}_i,$$

which is crucial in decryption, where $\{\mathbf{u}_i\}, \{\mathbf{B}_i\}, \{\mathbf{D}_i\}$ are the public parameter components generated in the setup phase.

To realize authentication, the sender encryption key needs to be bounded together with the encrypted message in the encryption process, and meanwhile the resulted intermediate component (in which the sender attribute and the encrypted message are embedded) must be encapsulated in the ciphertext to realize data confidentiality. To achieve this, we invoke the lattice trapdoor delegation algorithm $\mathsf{DelTrap}(\mathbf{F}_{satt}, \mathbf{T}_{\mathbf{A}'}, \delta)$ under the trapdoor $\mathbf{T}_{\mathbf{A}'}$ (for lattice $\Lambda_q^{\perp}(\mathbf{A}')$) to generate a delegated trapdoor $\mathbf{T}$ (for lattice $\Lambda_q^{\perp}(\mathbf{F}_{satt})$) as the signature key of a SIS-based signature scheme and include $\mathbf{T}$ into the encryption key $ek_{satt} = (\mathbf{T}, \mathbf{D}_{satt}, satt)$, where $\mathbf{D}_{satt} = (\mathbf{D}_1 + satt_1\mathbf{G}|\cdots|\mathbf{D}_\ell + satt_\ell\mathbf{G})$, $\mathbf{H}_{satt} = H_1(pp, \mathbf{D}_{satt})$, $\mathbf{F}_{satt} = [\mathbf{A}'|\mathbf{H}_{satt}]$ and $\mathbf{G}$ is the gadget matrix.

During the encryption, for a message μ and a receiver attribute $ratt$ specified by the sender, we first execute a SIS-based signature algorithm $\mathsf{PreSamp}$ ($\mathbf{F}_{satt}$, $\mathbf{T}$, $\mathbf{h}_\mu$, δ) under the signature key $\mathbf{T}$ (which is a part of the encryption key $ek_{sattr} = (\mathbf{T}, \mathbf{D}_{satt}, satt)$) to generate a signature $\boldsymbol{\sigma}$ on the encrypted message μ such that $\mathbf{F}_{satt}.\boldsymbol{\sigma} = \mathbf{h}_\mu$, where $\mathbf{h}_\mu = H_2(pp, satt, \mu|ratt)$, $\mathbf{H}_{satt} = H_1(pp, \mathbf{D}_{satt})$, $\mathbf{F}_{satt} = [\mathbf{A}'|\mathbf{H}_{satt}]$. Then a bare ciphertext component $\mathbf{c}_0 = \mathbf{A}^T\mathbf{s}_0 + \mathbf{e}_0$ and two attribute-related ciphertext components $\mathbf{c}_1 = \mathbf{D}_{ratt}^T\mathbf{s}_0 + \mathbf{e}_1$, $\mathbf{c}_2 = \mathbf{D}_{satt}^T\mathbf{s}_0 + \mathbf{e}_2$ are calculated under the secret $\mathbf{s}_0$ and noisy $\mathbf{e}_0$, $\mathbf{e}_1$, $\mathbf{e}_2$. Next, we use N fresh LWE tuples to conceal the message μ and signature $\boldsymbol{\sigma}$ bit-by-bit, and get N message-related ciphertext components $\{c_{3,i}\}_{i\in[N]}$:

$$c_{3,1} = \mathbf{u}_1^T\mathbf{s}_0 + x_1 + \mu_1\lceil\frac{q}{2}\rceil, \cdots, c_{3,\ell_0} = \mathbf{u}_{\ell_0}^T\mathbf{s}_0 + x_{\ell_0} + \mu_{\ell_0}\lceil\frac{q}{2}\rceil,$$

$$c_{3,\ell_0+1} = \mathbf{u}_{\ell_0+1}^T\mathbf{s}_0 + x_{\ell_0+1} + \boldsymbol{\sigma}_1\lceil\frac{q}{2}\rceil, \cdots, c_{3,N} = \mathbf{u}_N^T\mathbf{s}_0 + x_N + \boldsymbol{\sigma}_{\ell_1}\lceil\frac{q}{2}\rceil,$$

where $|\mu| = \ell_0$, $|\sigma'| = \ell_1$, and $\ell_0 + \ell_1 = N$. The resulted ciphertext consists of $ct = (\mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_2, \{c_{3,i}\}_{i\in[N]})$.

To decrypt a ciphertext $ct = (\mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_2, \{c_{3,i}\}_{i\in[N]})$ under the decryption keys (rk_{f_r}, pok_{f_s}), we first parse the ciphertext components $\mathbf{c}_1$, $\mathbf{c}_2$ into $\{\mathbf{c}_{1,i}\}_{i\in[\ell]}$, $\{\mathbf{c}_{2,i}\}_{i\in[\ell]}$, then use the matrix embedding technique to compute the homomorphic ciphertexts

$$\mathbf{c}_{f_r} = \mathsf{Eval}_{ct}(f_r, \{(\mathbf{B}_i, ratt_i\mathbf{G}, \mathbf{c}_{1,i})\}_{i\in[\ell]}),$$

$$\mathbf{c}_{f_s} = \mathsf{Eval}_{ct}(f_s, \{(\mathbf{D}_i, satt_i\mathbf{G}, \mathbf{c}_{2,i})\}_{i\in[\ell]}),$$

where $ratt \in \{0,1\}^\ell$, $satt \in \{0,1\}^\ell$, and $ratt_i$ and $satt_i \in \{0,1\}$ are respectively the i-th bit of the attributes $ratt$, $satt$. Next compute

$$w_i = c_{3,i} - (\mathbf{r}_{f_r}^i)^T \begin{bmatrix} \mathbf{c}_0 \\ \mathbf{c}_{f_r} \end{bmatrix} - (\mathbf{r}_{f_s}^i)^T \begin{bmatrix} \mathbf{c}_0 \\ \mathbf{c}_{f_s} \end{bmatrix},$$

and check whether $|w_i - \lceil\frac{q}{2}\rceil| < \frac{q}{4}$ holds, if yes, set $m_i' = 1$, otherwise set $m_i' = 0$. Note that since

$$(\mathbf{r}_{f_r}^i)^T \begin{bmatrix} \mathbf{c}_0 \\ \mathbf{c}_{f_r} \end{bmatrix} = (\mathbf{u}_{1,i})^T\mathbf{s}_0 + (\mathbf{r}_{f_r}^i)^T \begin{bmatrix} \mathbf{e}_0 \\ \mathbf{e}_{f_r} \end{bmatrix},$$

$$(\mathbf{r}_{f_s}^i)^T \begin{bmatrix} \mathbf{c}_0 \\ \mathbf{c}_{f_s} \end{bmatrix} = (\mathbf{u}_{2,i})^T\mathbf{s}_0 + (\mathbf{r}_{f_s}^i)^T \begin{bmatrix} \mathbf{e}_0 \\ \mathbf{e}_{f_s} \end{bmatrix},$$

we have

$$|w_i - \lceil \frac{q}{2} \rceil| \leq |x_i| + \| (\mathbf{r}_{f_r})^T \| \cdot \left\| \begin{bmatrix} \mathbf{e}_0 \\ \mathbf{e}_{f_r} \end{bmatrix} \right\| + \|(\mathbf{r}_{f_s})^T\| \cdot \left\| \begin{bmatrix} \mathbf{e}_0 \\ \mathbf{e}_{f_s} \end{bmatrix} \right\|$$
$$\leq O(Qm^2(m+1)^{2d}\sqrt{\log m}) < \frac{q}{4}.$$

By setting appropriate system parameters, we can ensure that the inequality holds (refer to section 5.2 for detailed parameter settings), and thus the encrypted messages can be recovered correctly. For the formal description of the construction, see section 5.1 for details.

Furthermore, we prove that our scheme achieves selective IND-CPA (sIND-CPA) security and authentication via formal reduction proofs from the sIND-CPA security to the LWE assumption, and from the authentication to the SIS assumption, where "selective" means that in the sIND-CPA security experiment, a challenge receiver attribute and sender attribute pair $(ratt^*, satt^*)$ is announced before the Setup by the adversary.

Application to Attribute-Based Matchmaking Encryption. Finally, we explore the application of bilateral access control ABE and use it to yield an Attribute-Based Matchmaking Encryption (AB-ME) scheme from the LWE and SIS assumptions. AB-ME [15, 16] is similar to bilateral access control ABE except that the former is additionally required to satisfy user-privacy (i.e., the sender attributes and receiver attributes are also hidden). More specifically, the security of a standard AB-ME scheme should satisfy the following properties: If a mismatch happens, nothing is leaked except the fact that a match did not occur; If a match happens, nothing is leaked except for the encrypted message and the fact that a match occurred; A valid ciphertext can not be forged under an unauthorized encryption key (not generated by the key generation center), i.e, the AB-ME should satisfy authentication.

In a nutshell, for a (key-policy) AB-ME, our construction works as follows. To encrypt a message μ under the sender attribute $satt$ and receiver attribute $ratt$, the AB-ME encryption algorithm first computes a signature σ on $(\alpha|ratt)$ and a ciphertext ct_α on the randomly selected lock string α according to the encryption algorithm $\mathsf{BACABE.Enc}$. Then it constructs a decryption circuit

$$P \leftarrow \mathsf{BACABE.Dec}(pp, \cdot, \cdot, (ct_\alpha, ratt, satt))$$

using the bilateral access control ABE decryption algorithm $\mathsf{BACABE.Dec}$, next it masks the message μ, the signature σ, the attributes $ratt$, $satt$ and the lock α using the obfuscation algorithm of the lockable obfuscation scheme [32], i.e.,

$$\widetilde{P} \leftarrow \mathsf{Obf}(1^\lambda, P, \mu||\sigma||ratt||satt||\alpha||ct_\alpha, \alpha),$$

where the intermediate signature σ is generated as that in the encryption algorithm $\mathsf{BACABE.Enc}(pp, ek_{satt}, \mu, satt)$. Analogously, the sender key ek_{satt}, the receiver key rk_{f_r} and the policy key pok_{f_s} are built with the sender key generation algorithm, the receiver key generation algorithm and the policy key generation algorithm of the AB-ME scheme. By executing the evaluation algorithm of the lockable obfuscation scheme on the obfuscated circuit $\widetilde{P}$ under the keys rk_{f_r}, pok_{f_s}, the message $\mu||\sigma||ratt||satt||\alpha||ct_\alpha$ can be recovered when the attributes $satt$, $ratt$ respectively match with their polices f_s, f_r, i.e. $\mu||\sigma||ratt||satt||\alpha||ct_\alpha \leftarrow \mathsf{Eval}(\widetilde{P}, rk_{f_r}, pok_{f_s})$ when $f_s(satt) = 0$ and $f_r(ratt) = 0$ hold.

We sketch the security reductions as follows. If the bilateral access control ABE scheme satisfies sIND-CPA security and the lockable obfuscation scheme is secure, then the AB-ME scheme satisfies privacy, and if the the bilateral access control ABE scheme satisfies authentication, then the AB-ME scheme also satisfies authentication. We stress that we only achieve the weaker privacy security notion in case of mismatch, but the seminal works of ME [15, 16] defined the privacy security of ME in case of match. Still, considering the weaker privacy security is meaningful and non-trivial to be achieved. We provide more details in section 6.

Previous works [20, 21] consider AB-ME schemes with unbounded collusion for either very restricted policies (i.e., for identity matching) under the LWE and SIS assumptions, or for arbitrary policies from classical assumptions such as bilinear maps [5–7]. In particular, we stress that while the ME scheme proposed in [30, 31] is also constructed using the lockable obfuscation scheme and for arbitrary policies from the LWE assumption (subexponential LWE), it fails to provide authentication (or the authentication can only be implemented using Non-Interactive Zero-Knowledge proof system (NIZK)).

Related Works. Shamir [9] proposed the notion of Identity-Based Encryption (IBE) to simplify key management in the traditional PKE setting. In IBE, a message is encrypted by a sender under a user's identity. If the decryptor has the same identity as the identity embedded in the ciphertext, he/she can recover the encrypted message from the ciphertext. According to the decryption control mode, IBE can be viewed as the access control cryptosystem supporting equality policy. To overcome the single access control (i.e., equality policy) in IBE, the notion of Attribute-Based Encryption (ABE) [2] is proposed, which extends the user's "identity" to some expressive attributes and supports more flexible access control. However, the current ABEs [11, 12, 8, 13, 14, 10] cannot support bilateral access control that allows both parties to specify access control policy for each other.

Matchmaking encryption, proposed by Ateniese et al. [15, 16], supports bilateral access control, in which when the attributes of the sender and receiver match the access control policies they specify for each other, the message can be correctly recovered from the ciphertext. According to the matching rules, matchmaking encryption is categorized into many-to-many ME and one-to-one ME where the former corresponds to ME in the ABE settings (dubbed as AB-ME) and the latter corresponds to ME in the IBE settings (dubbed as IB-ME). Researchers have paid widespread attention to matchmaking encryptions in different applications based on different assumptions. For example, in AB-ME, Li et al. [4] proposed an efficient privacy-protecting bilateral friendly AB-ME scheme under bilinear pairing assumption in mobile social networks. Xu et al. [5] put forward a server-aided bilateral access control AB-ME with user revocation property under the non-standard q-1 assumption in the Internet of Things (IoT) applications. Xu et al. [6] proposed an AB-ME scheme under the standard DBDH and CDH assumptions and used it to build a data sharing system in the cloud computing environment. Additionally, a lightweight bilateral access control ABE was introduced in [7] and used for constructing data sharing system capable of bilateral access control in cloud IoT.

In IBE settings, the first IB-ME scheme was proposed by Ateniese et al. [15, 16] under the BDH assumption in the random oracle model. Later, Francati et al. [17] proposed another IB-ME based on the non-standard q-ABDHE assumption in the standard model. In addition, a generic transformation from private IB-ME to plain IB-ME was presented in [17]. To remove the non-standard assumption in [17], Chen et al. [18] presented an IB-ME scheme under the standard SXDH assumption in the standard model. Next, Wu et al. [19] put forward a fuzzy IB-ME under the DBDH and CDH assumptions in the standard model which addressed an opening problem proposed by Ateniese et al. [15]. By integrating two-level HIBE with ME, Wang et al. [20] proposed an IB-ME generic construction. Recently, Wang et al. [21] proposed a concrete construction for IB-ME under the standard lattice assumptions. In order to resolve the key exposure in IB-ME, Jiang et al. [22] presented a revocable IB-ME scheme under the standard DBDH and CDH assumptions. In this paper, we propose a bilateral access control ABE scheme based on the standard LWE and SIS assumptions for the first time, and explore its application to the construction of attribute-based matchmaking encryption (AB-ME) scheme with post-quantum security.

2 Preliminary

In this section, we give some notions and preliminaries used in the proposed scheme.

2.1 Notations

In this paper, we use $negl(\lambda)$ to denote a negligible function in the security parameter λ and use *ppt* to denote probabilistic polynomial time. For convenience, let $[N]$ denote the set $\{1, 2, \cdots, N\}$. We use bold uppercase letters to represent matrices and bold lowercase letters to represent vectors. For a matrix $\mathbf{A}$ and a vector $\mathbf{x}$, we use $\mathbf{A}^T$ and $\mathbf{x}^T$ to denote their transposes. In particular, we use $||\mathbf{b}||$ to denote the ℓ_2 norm of a vector $\mathbf{b} \in \mathbb{Z}_q^n$, which is defined as $||\mathbf{b}|| := \sqrt{\sum_{i=1}^n \mathbf{b}_i^2}$, where $\mathbf{b}_i$ denotes the *i-th* component of $\mathbf{b}$. Correspondingly, we use $||\mathbf{A}||$ to denote the ℓ_2 norm of a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times n}$, which is defined as the ℓ_2 norm of its longest column vector. We use $||\widetilde{\mathbf{A}}||$ to denote the ℓ_2 norm of its Gram-Schmidt (GS) orthogonalization. Moreover, the largest singular value of the matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ is defined as $||\mathbf{A}||_2 := \sup_{||\mathbf{e}||=1} ||\mathbf{Ae}||$, where $\mathbf{e} \in \mathbb{Z}_q^m$.

Thus we have $||\widetilde{\mathbf{A}}|| \leq ||\mathbf{A}|| \leq ||\mathbf{A}||_2 \leq \sqrt{m} \cdot ||\mathbf{A}||$ and $||\mathbf{AB}||_2 = ||\mathbf{A}||_2||\mathbf{B}||_2$ for any $\mathbf{B} \in \mathbb{Z}_q^{m\times k}$.

2.2 Lattice Definitions and Assumptions

In this section, we will give some lattice definitions and lattice assumptions used in our scheme.

Definition 1 (Lattice [23]). *For a prime number q, matrices $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$, $\boldsymbol{D} \in \mathbb{Z}_q^{n\times k}$ and a vector $\boldsymbol{u} \in \mathbb{Z}_q^n$, we give several lattice definitions:*

$$\Lambda_q(\boldsymbol{A}) := \{\boldsymbol{e} \in \mathbb{Z}^m \text{ s.t. } \exists \boldsymbol{s} \in \mathbb{Z}_q^n \text{ where } \boldsymbol{A}^T\boldsymbol{s} = \boldsymbol{e} \bmod q\}$$
$$\Lambda_q^{\perp}(\boldsymbol{A}) := \{\boldsymbol{e} \in \mathbb{Z}^m \text{ s.t. } \boldsymbol{A}\boldsymbol{e} = \boldsymbol{0} \bmod q\}$$
$$\Lambda_q^{\boldsymbol{u}}(\boldsymbol{A}) := \{\boldsymbol{e} \in \mathbb{Z}^m \text{ s.t. } \boldsymbol{A}\boldsymbol{e} = \boldsymbol{u} \bmod q\}$$
$$\Lambda_q^{\boldsymbol{D}}(\boldsymbol{A}) := \{\boldsymbol{R} \in \mathbb{Z}^{m\times k} \text{ s.t. } \boldsymbol{A}\boldsymbol{R} = \boldsymbol{D} \bmod q\}$$

Definition 2 (Discrete Gaussians [23]). *For $L \subset \mathbb{Z}^m$, $\boldsymbol{c} \in \mathbb{R}^m$ and $\delta \in \mathbb{R}$, we have the following definition:*

$$\rho_{\delta,\boldsymbol{c}}(\boldsymbol{x}) = \exp(-\pi\frac{||\boldsymbol{x}-\boldsymbol{c}||^2}{\delta^2}) \quad , \quad \rho_{\delta,\boldsymbol{c}}(L) = \sum_{\boldsymbol{x}\in L}\rho_{\delta,\boldsymbol{c}}(\boldsymbol{x})$$

Again, we define:

$$\forall \boldsymbol{y} \in L, \quad \mathcal{D}_{L,\delta,\boldsymbol{c}}(\boldsymbol{y}) = \frac{\rho_{\delta,\boldsymbol{c}}(\boldsymbol{y})}{\rho_{\delta,\boldsymbol{c}}(L)}$$

In what follows, we write $\rho_{\delta,\boldsymbol{0}}$ as ρ_δ and $\mathcal{D}_{L,\delta,\boldsymbol{0}}$ as $\mathcal{D}_{L,\delta}$ for simplicity.

Definition 3 (Gadget Matrix [14]). *Let $q \geq 2$ and $n \geq 1$, we review the gadget matrix defined in [14]. A gadget matrix is defined as $\boldsymbol{G} := \boldsymbol{I}_n \otimes \boldsymbol{g} \in \mathbb{Z}_q^{n\times m}$ for $m = n\lceil\log q\rceil$ where $\boldsymbol{g} := (1, 2, \cdots, 2^{\lceil\log q\rceil-1}) \in \mathbb{Z}_q^{\lceil\log q\rceil}$. Its inversion function is defined as $\boldsymbol{G}^{-1} : \mathbb{Z}_q^{n\times m} \to \{0,1\}^{m\times m}$ such as $\boldsymbol{G} \cdot \boldsymbol{G}^{-1}(\boldsymbol{A}) = \boldsymbol{A}$ for $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$.*

Lemma 1 (Trapdoor Generation [24]). *For $n \geq 1$, $q \geq 2$ and $m = \Theta(n\log q)$, there exists a ppt algorithm $\mathsf{TrapGen}(q,n)$ that samples $(\boldsymbol{A}, \boldsymbol{T_A})$, where $\boldsymbol{A} \overset{s}{\cong} \mathcal{U}$ and $\boldsymbol{T_A}$ is a short basis (called trapdoor) for lattice $\Lambda_q^{\perp}(\boldsymbol{A})$ which satisfies:*

$$||\widetilde{\boldsymbol{T_A}}||_2 \leq O(\sqrt{n\log q}) \quad \text{and} \quad ||\boldsymbol{T_A}||_2 \leq O(n\log q)$$

where $\mathcal{U} \leftarrow_{\$} \mathbb{Z}_q^{n\times m}$.

Lemma 2 ([23]). *Let $n, q, k > 2$, $m > n$. Given $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$ and a trapdoor $\boldsymbol{T_A} \in \mathbb{Z}_q^{m\times m}$ for lattice $\Lambda_q^{\perp}(\boldsymbol{A})$, for any $\delta \geq ||\widetilde{\boldsymbol{T_A}}|| \cdot \omega(\sqrt{\log m})$, $\boldsymbol{u} \in \mathbb{Z}_q^n$ and $\boldsymbol{D} \in \mathbb{Z}_q^{n\times k}$, we have*

$$\mathsf{Pr}[\boldsymbol{x} \leftarrow \mathcal{D}_{\Lambda_q^{\boldsymbol{u}}(\boldsymbol{A}),\delta} \mid ||\boldsymbol{x}|| > \sqrt{m}\delta] = negl(n).$$

$$\mathsf{Pr}[\boldsymbol{R} \leftarrow \mathcal{D}_{\Lambda_q^{\boldsymbol{D}}(\boldsymbol{A}),\delta} \mid ||\boldsymbol{R}||_2 > \sqrt{mk}\delta] = negl(n).$$

Lemma 3 ([24]). *For* $\boldsymbol{R} \leftarrow_{\$} \{-1,1\}^{m\times m}$*, we have* $\mathsf{Pr}[||\boldsymbol{R}||_2 > 20\sqrt{m}] = negl(n)$ *hold.*

Definition 4 (Q-bounded Distribution [10]). *An ensemble distribution* $\{\chi_n\}_{n\in\mathbb{N}}$ *is called* Q*-bounded distribution over integer, if* $\underset{e\leftarrow\chi_n}{\mathsf{Pr}}[|e|>Q] \le 2^{-\widetilde{\Omega}(n)}$.

Lemma 4 (Leftover Hash Lemma [25]). *Let* $\boldsymbol{A}$, $\boldsymbol{B} \in \mathbb{Z}_q^{n\times m}$ *be two uniformly random matrices,* $\boldsymbol{R}$ *be chosen uniformly from* $\{-1,1\}^{m\times m}$*. Then* $(\boldsymbol{A}, \boldsymbol{AR}, \boldsymbol{R}^T\boldsymbol{v}) \overset{s}{\cong} (\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{R}^T\boldsymbol{v})$ *for all* $\boldsymbol{v}\in\mathbb{Z}_q^m$*, prime* $q>2$ *and* $m > (n+1)\log q + \omega(\log n)$.

Lemma 5 (Preimage Sampling [23]). *Given a prime* $q \ge 2$*, a matrix* $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$*, a trapdoor* $\boldsymbol{T_A} \in \mathbb{Z}_q^{m\times m}$ *and a parameter* $\delta \ge ||\widetilde{\boldsymbol{T_A}}||\omega(\sqrt{\log m})$*, there exists a ppt algorithm* $\mathsf{PreSam}(\boldsymbol{A}, \boldsymbol{T_A}, \boldsymbol{u}, \delta)$ *that samples a vector* $\boldsymbol{v} \leftarrow \mathcal{D}_{\Lambda_q^{\boldsymbol{u}}(\boldsymbol{A}),\delta}$ *such that* $\boldsymbol{Av} = \boldsymbol{u} \mod q$*, where* $\boldsymbol{u}\in\mathbb{Z}_q^n$*,* $\boldsymbol{v}\in\mathbb{Z}_q^m$*.*

Lemma 6 (Delegation Trapdoor Generation [26]). *For* $q\ge 2$*,* $m>n$*, there exists a ppt algorithm* $\mathsf{DelTrap}(\boldsymbol{F}, \boldsymbol{T_A}, \delta)$ *that inputs a matrix* $\boldsymbol{F} = [\boldsymbol{A}|\boldsymbol{C}] \in \mathbb{Z}_q^{n\times(m+k)}$*, a trapdoor* $\boldsymbol{T_A} \in \mathbb{Z}_q^{m\times m}$ *and a parameter* $\delta \ge \omega(\sqrt{\log m})$*, and outputs a trapdoor* $\boldsymbol{T_F} \in \mathbb{Z}_q^{(m+k)\times(m+k)}$ *with* $s_1(\boldsymbol{T_F}) \le \delta\cdot O(\sqrt{m}+\sqrt{k})$*, where* $s_1(\boldsymbol{T_F})$ *is the spectral norm of* $\boldsymbol{T_F}$*,* $(\boldsymbol{A}, \boldsymbol{T_A}) \leftarrow_{\$} \mathsf{TrapGen}(q,n)$*,* $\boldsymbol{A}\in\mathbb{Z}_q^{n\times m}$ *and* $\boldsymbol{C}\in\mathbb{Z}_q^{n\times k}$*.*

Definition 5 (Learning with Errors (LWE) [24]). *For a prime* $q>1$*, an integer* $n>1$ *and* $m \ge n\log(q)$ *and a* Q*-bounded noisy distribution* $\chi = \chi(n)$ *over* $\mathbb{Z}_q$*, the* (q,n,χ)*-LWE problem is to distinguish two distributions with one distribution being associated with the outputs of a noisy pseudorandom sampling oracle* $\mathcal{O}_s$ *for which a random secret vector* $\boldsymbol{s}\in\mathbb{Z}_q^n$ *is fixed, and the other distribution being associated with the outputs of a truly random sampling oracle* $\mathcal{O}_{\$}$*. The two oracles are defined as follows.*

$\mathcal{O}_{\boldsymbol{s}}$*: This oracle outputs tuples* $(\boldsymbol{u}_i, \boldsymbol{v}_i) = (\boldsymbol{u}_i, \boldsymbol{u}_i^T\boldsymbol{s} + x_i)$ *with a fixed secret key* $\boldsymbol{s}$*, where* $x_i\in\mathbb{Z}_q$ *is sampled according to* χ*, and* $\boldsymbol{u}_i$ *is chosen uniformly at random from* $\mathbb{Z}_q^n$*.*

$\mathcal{O}_{\$}$*: It returns uniformly random tuples* $(\boldsymbol{u}_i, \boldsymbol{v}_i)$ *from* $\mathbb{Z}_q^n \times \mathbb{Z}_q$*.*

We define a general oracle $\mathcal{O}$ which could be $\mathcal{O}_{\mathbf{s}}$ or $\mathcal{O}_{\$}$. Furthermore, the (q,n,χ)-LWE problem is allowed to query the oracle $\mathcal{O}$ multiple times. The (q,n,χ)-LWE assumption holds if for any ppt adversary $\mathcal{A}$, the advantage of $\mathcal{A}$ distinguishing the outputs of $\mathcal{O}_{\mathbf{s}}$ and $\mathcal{O}_{\$}$ is negligible, i.e.,

$$\mathsf{Adv}_{\mathcal{A}}^{lwe}(\lambda) = |\mathsf{Pr}[\mathcal{A}^{\mathcal{O}_{\mathbf{s}}}(\lambda)=1] - \mathsf{Pr}[\mathcal{A}^{\mathcal{O}_{\$}}(\lambda)=1]| \le negl(\lambda)$$

As shown in [27], for $q = q(n) \le 2^n$, Q-bounded distribution $\chi = \chi(n)$ with $Q = Q(n) \ge \omega(\log n)\sqrt{n}$ and $q/Q \ge 2^{n^\epsilon}$ where $\epsilon > 0$, if there exists a ppt algorithm that can solve the (q,n,χ)-LWE problem, then there exists a quantum algorithm that can approximately resolve the SIVP and GapSVP problems, where the approximate factor is $2^{\Omega(n^\epsilon)}$.

Definition 6 (Short Integer Solution (SIS) [28]). *A short integer solution* $\mathsf{SIS}_{n,m,q,\beta}$ *problem is hard if for integers* n*,* m*,* q *and* β *and* $\boldsymbol{A} \leftarrow_{\$} \mathbb{Z}_q^{n\times m}$*, there does not exist a ppt adversary* $\mathcal{A}$ *which could come up with an integer solution* $\boldsymbol{z} \in \mathbb{Z}_q^m$ *s.t.* $\boldsymbol{z} \neq 0$, $||\boldsymbol{z}|| \leq \beta$ *and* $\boldsymbol{Az} = 0 \bmod q$ *with non-negligible probability.*

Theorem 1 (SampleLeft [23]). *The ppt sampling algorithm* $\mathsf{SampleLeft}(\boldsymbol{A}, \boldsymbol{C}, \boldsymbol{T_A}, \boldsymbol{u}, \delta)$ *is given inputs* $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$*,* $\boldsymbol{C} \in \mathbb{Z}_q^{n\times m_1}$*,* $\boldsymbol{T_A} \in \mathbb{Z}_q^{m\times m}$*,* $\boldsymbol{u} \in \mathbb{Z}_q^n$ *and* $\delta > ||\widetilde{\boldsymbol{T_A}}||\omega(\sqrt{\log(m+m_1)})$*, and returns a short* $\boldsymbol{e} \in \mathbb{Z}^{m+m_1}$ *s.t.* $\boldsymbol{u} = [\boldsymbol{A}|\boldsymbol{C}]\boldsymbol{e}$*, where* $\boldsymbol{e} \leftarrow \mathcal{D}_{\Lambda_q^{\boldsymbol{u}}(\boldsymbol{F}_1),\delta}$*,* $q > 2$*,* $m > n$*,* $\boldsymbol{F}_1 = [\boldsymbol{A}|\boldsymbol{C}]$*.*

Theorem 2 (SampleRight [23]). *The ppt sampling algorithm* $\mathsf{SampleRight}(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{R}, \boldsymbol{T_B}, \boldsymbol{u}, \delta)$ *is given inputs* $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$*,* $\boldsymbol{B} \in \mathbb{Z}_q^{n\times m}$ *with rank* n*,* $\boldsymbol{R} \in \mathbb{Z}_q^{m\times m}$*,* $\boldsymbol{T_B} \in \mathbb{Z}_q^{m\times m}$*,* $\boldsymbol{u} \in \mathbb{Z}_q^n$ *and* δ*, and returns a short* $\boldsymbol{e} \in \mathbb{Z}^{2m}$ *s.t.* $\boldsymbol{u} = [\boldsymbol{A}|\boldsymbol{AR}+\boldsymbol{B}]\boldsymbol{e}$*, where* $\boldsymbol{e} \leftarrow \mathcal{D}_{\Lambda_q^{\boldsymbol{u}}(\boldsymbol{F}_2),\delta}$*,* $\boldsymbol{F}_2 = [\boldsymbol{A}|\boldsymbol{AR}+\boldsymbol{B}]$*,* $q > 2$*,* $m > n$*, and* $\delta > ||\widetilde{\boldsymbol{T_B}}|| \cdot s_R\omega(\sqrt{\log m})$*,* $s_R := ||\boldsymbol{R}||_2 = \sup_{||\boldsymbol{x}||=1} ||\boldsymbol{Rx}||$*. In particular, if* $\boldsymbol{B}$ *is the gadget matrix* $\boldsymbol{G}$*, we have* $\delta > \sqrt{5} \cdot (||\boldsymbol{R}||_2 + 1)\omega(\sqrt{\log m})$*.*

2.3 Matrix Embedding

Now we review the matrix embedding techniques proposed by Boneh et al. [8] from the fully homomorphic encryption scheme [29], which is an approach that embed circuits into LWE matrices.

Let $\mathbf{B}_1, \cdots, \mathbf{B}_\ell \in Z_q^{n\times m}$, $f : \{0,1\}^\ell \to \{0,1\}$ be a circuit of depth d and χ be a Q-bounded distribution. For any $attr \in \{0,1\}^\ell$ and $i \in [\ell]$, $\mathbf{c}_i = (\mathbf{B}_i + attr_i\mathbf{G})^T\mathbf{s} + \mathbf{e}_i$, there exist three homomorphic algorithms defined as below, where $\mathbf{s} \leftarrow_{\$} Z_q^n$, $\mathbf{e}_i \leftarrow_{\$} \chi^m$, and $attr_i$ is the i-th bit of $attr$.

– $\mathsf{Eval}_{pp}(f, (\mathbf{B}_1, \cdots, \mathbf{B}_\ell)) \to \mathbf{B}_f$: On input a policy function f, and ℓ matrices $(\mathbf{B}_1, \cdots, \mathbf{B}_\ell)$, the algorithm outputs a homomorphic matrix $\mathbf{B}_f \in Z_q^{n\times m}$.
– $\mathsf{Eval}_{ct}(f, \{\mathbf{B}_i, attr_i\mathbf{G}, \mathbf{c}_i\}_{i\in[\ell]}) \to \mathbf{c}_f$: On input a policy function f, ℓ matrices $(\mathbf{B}_1, \cdots, \mathbf{B}_\ell)$, length-$\ell$ attribute $attr \in \{0,1\}^\ell$ and ℓ vectors $\mathbf{c}_1, \cdots, \mathbf{c}_\ell \in Z_q^m$, the algorithm outputs a homomorphic vector $\mathbf{c}_f \in Z_q^m$ such that

$$\mathbf{c}_f = (\mathbf{B}_f + f(attr)\mathbf{G})^T\mathbf{s} + \mathbf{e}_f,$$

where $\mathbf{B}_f = \mathsf{Eval}_{pp}(f, (\mathbf{B}_1, \cdots, \mathbf{B}_\ell))$, $\mathbf{e}_f \in Z_q^m$, $||\mathbf{e}_f|| \leq Q\sqrt{m}(m+1)^d$ with all but a negligible probability.
– $\mathsf{Eval}_{sim}(f, \{(\mathbf{S}_i^*, attr_i^*)\}_{i\in[\ell]}, \mathbf{A}) \to \mathbf{S}_f^*$: On input a policy function f, ℓ short matrices $(\mathbf{S}_1^*, \cdots, \mathbf{S}_\ell^*) \in \{-1,1\}^{m\times m}$, length-$\ell$ attribute $attr \in \{0,1\}^\ell$ and a matrix $\mathbf{A} \in Z_q^{n\times m}$, the algorithm outputs a homomorphic matrix $\mathbf{S}_f^* \in \{-1,1\}^{m\times m}$ such that

$$\mathbf{AS}_f^* - f(attr^*)\mathbf{G} = \mathbf{B}_f$$

where $\mathbf{B}_f = \mathsf{Eval}_{pp}(f, (\mathbf{AS}_1^* - attr_1^*\mathbf{G}, \cdots, \mathbf{AS}_\ell^* - attr_\ell^*\mathbf{G}))$, $||\mathbf{S}_f^*||_2 \leq 20\sqrt{m}(m+1)^d$ with all but a negligible probability.

2.4 Lockable Obfuscation Scheme

In this section, we review the definition of lockable obfuscation scheme [32]. Let $\mathcal{C}_{n,s,d}(\lambda)$ be a circuit family with depth d, input size n and output size s, where $n(\cdot)$, $s(\cdot)$ and $d(\cdot)$ are polynomials in the security parameter λ. A lockable obfuscation scheme Π for the circuit family $\mathcal{C}_{n,s,d}(\lambda)$ and message space $\mathcal{M}$ consists of the following two ppt algorithms.

– $\mathsf{Obf}(1^\lambda, \mathbb{C}, m, y)$: The lockable obfuscator is a randomized algorithm, which takes as input a security parameter 1^λ, a circuit $\mathbb{C} \in \mathcal{C}_{n,s,d}(\lambda)$, a message $m \in \mathcal{M}$ and a lock $y \in \{0,1\}^s$, and outputs an obfuscated circuit $\widetilde{\mathbb{C}}$.
– $\mathsf{Eval}(\widetilde{\mathbb{C}}, x)$: The evaluation algorithm is a deterministic algorithm, which takes as input an obfuscated circuit $\widetilde{\mathbb{C}}$ and an input $x \in \{0,1\}^n$, and outputs a message $m \in \mathcal{M} \cup \{\perp\}$.

Correctness. For all $\mathbb{C} \in \mathcal{C}_{n,s,d}(\lambda)$, message $m \in \mathcal{M}$ and lock $y \in \{0,1\}^s$, and $\widetilde{\mathbb{C}} \leftarrow \mathsf{Obf}(1^\lambda, \mathbb{C}, m, y)$:

1. If $\mathbb{C}(x) = y$, then $\mathsf{Pr}[\mathsf{Eval}(\mathsf{Obf}(1^\lambda, \mathbb{C}, m, y), x) = m] = 1$.
2. If $\mathbb{C}(x) \neq y$, then $\mathsf{Pr}[\mathsf{Eval}(\mathsf{Obf}(1^\lambda, \mathbb{C}, m, y), x) = \perp] \geq 1 - negl(\lambda)$.

In fact, according to [32], the above definition is called the semi-statistical correctness of the lockable obfuscation scheme.

Security. The security of the lockable obfuscation scheme Π must hide any information about the circuit $\mathbb{C}$, the message m and the lock y when the lock y is chosen randomly. This follows that there exists a ppt simulator $\mathcal{S}$ which can simulate the obfuscated circuit $\widetilde{\mathbb{C}}$. First we give the simulation game played between a ppt adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ and a challenger as follows.

$\underline{\mathsf{EXP}^{lsim}_{\Pi,\mathcal{A}}(\lambda)}$:

– The adversary $\mathcal{A}_0(1^\lambda)$ first outputs a circuit $\mathbb{C}$, a message m and an auxiliary value α.
– The challenger picks a random bit $b \in \{0,1\}$ and a random lock $y \in \{0,1\}^s$. It computes an obfuscated circuit $\widetilde{\mathbb{C}_0} \leftarrow \mathsf{Obf}(1^\lambda, \mathbb{C}, m, y)$ and a simulated obfuscation circuit $\widetilde{\mathbb{C}_1} \leftarrow \mathcal{S}(1^\lambda, 1^{|\mathcal{C}|}, 1^{|m|})$, where $|m|$ denotes the length of the message m.
– The challenger sends $\widetilde{\mathbb{C}_b}$ to $\mathcal{A}_1$. The adversary $\mathcal{A}_1$ takes as the security parameter 1^λ, the challenge obfuscation circuit $\widetilde{\mathbb{C}_b}$ and the auxiliary value α as inputs and outputs a guess b'.
– If $b' = b$, the experiment outputs 1, otherwise, it outputs 0.

Definition 7. *The lockable obfuscation scheme $\Pi = (\mathsf{Obf}, \mathsf{Eval})$ is secure, if the following advantage function $\mathsf{Adv}^{lsim}_{\Pi,\mathcal{A}}(\lambda)$ is negligible in security parameter λ. Namely*

$$\mathsf{Adv}^{lsim}_{\Pi,\mathcal{A}}(\lambda) = |\mathsf{Pr}[\mathsf{EXP}^{lsim}_{\Pi,\mathcal{A}}(\lambda) = 1] - \frac{1}{2}| \leq negl(\lambda).$$

Note that the definition above is reviewed according to [32]. Wichs and Zirdelis [33] proposed a slightly more general notion of obfuscation for multi-bit compute-and-compare circuits where the lock is only required to be unpredictable. They also give a LWE-based instantiation for multi-bit compute-and-compare circuits.

2.5 Identity-Based Signature

An Identity-Based Signature (IBS) allows a signer to execute signature on a message m using a signature key associated with his/her identity. Any receiver can verify the received signature with the corresponding verification key (i.e., the signer's identity).

Generally speaking, an IBS scheme consists of four ppt algorithms: (1) The parameter generation algorithm $\mathsf{SSetup}(\lambda)$ takes as input a security parameter λ and returns a public parameter pp and a master secret key msk. Note that pp is taken implicitly as the input in the following algorithms; (2) The signature key generation algorithm $\mathsf{SKGen}(msk, id)$ takes as input the master secret key msk and a signer's identity id, and outputs a signature key ssk_{id}; (3) The signature algorithm $\mathsf{SSign}(ssk_{id}, id, m)$ takes as input a signature key ssk_{id}, the signer's identity id and a message m, and returns a signature σ; (4) The verification algorithm $\mathsf{SVer}(id, m, \sigma)$ takes as input an identity id, a message m and a signature σ, and outputs "0" or "1" indicating that the signature is rejected or received.

The security requires that the IBS scheme satisfies UF-CMA. Roughly speaking, the security model involves four phases. **Setup:** In this phase, the challenger first generates the public parameter pp and the master secret key msk, and sends pp to the adversary. **Key Query:** In this phase, the adversary is allowed to make a series of signature key queries with his/her chosen identity id, and the challenger invokes the signature key generation algorithm to generate a signature key and sends it to the adversary. **Signature Query.** In this phase, the adversary can query the signature on his/her chosen identity/message pair (id, m), and the challenger runs the signature algorithm to generate a signature and sends it to the adversary. Note that in this phase, the identity id in (id, m) is not allowed to be that submitted in the key query phase. **Signature Forge:** In this phase, the adversary outputs a signature forgery σ^* on the identity/message pair (id^*, m^*) with $(id^*, m^*) \neq (id, m)$ and $id^* \neq id'$, where (id, m) denotes the queries in the signature query phase, and id' denotes the queries in the key query phase. The experiment outputs 1 if the forged signature σ^* on (id^*, m^*) satisfies $\mathsf{SVer}(id^*, m^*, \sigma^*) = 1$. If the success probability that the adversary outputs a valid forgery in the above UF-CMA experiment is negligible, we say that the IBS scheme is UF-CMA secure.

Wang et al. [20] proposed an instantiation of IBS scheme with UF-CMA security. Note that as indicated in [20], the IBS scheme captures the selective UF-CMA security, where the "selective" means that the adversary should submit all the signing key queries and signature queries before setup phase. In addition, we emphasize that the SIS-based instantiation of IBS scheme is applied in a semi-black-box way into our scheme.

3 Syntax and security definition of Key-Policy Bilateral Access Control ABE

In this section, we give the definition and security model of key-policy bilateral access control attribute-based encryption (bilateral access control ABE).

3.1 Definition

A key-policy bilateral access control ABE scheme consists of the following six ppt algorithms.

$\mathsf{Setup}(1^\lambda, \ell, N) \to (pp, msk)$. The setup algorithm $\mathsf{Setup}(1^\lambda, \ell, N)$ is executed by the Key Generation Center (KGC) which takes as input the security parameter 1^λ, the maximum length of the attributes ℓ and the maximum length of encrypted message $N \in \mathbb{N}$. It outputs a public parameter pp and a master secret key msk.

$\mathsf{SKGen}(pp, msk, satt) \to ek_{satt}$: The sender encryption key generation algorithm SKGen $(pp, msk,$ $satt)$ is run by the KGC which takes as input the public parameter pp, the master secret key msk and a sender's attribute $satt$, and outputs a sender encryption key ek_{satt}.

$\mathsf{RKGen}(pp, msk, f_r) \to rk_{f_r}$: The receiver key generation algorithm $\mathsf{RKGen}(pp,$ $msk,$ $f_r)$ is executed by the KGC which takes as input the public parameter pp, the master secret key msk and a receiver policy f_r, and outputs the receiver private key rk_{f_r}.

$\mathsf{PoKGen}(pp, msk, f_s) \to pok_{f_s}$: The policy key generation algorithm $\mathsf{PoKGen}(pp,$ $msk,$ $f_s)$ is run the KGC which takes as input the public parameter pp, the master secret key msk and a specified sender policy f_s, and outputs a policy key pok_{f_s}.

$\mathsf{Enc}(pp, ek_{satt}, \mu, ratt) \to ct$: The encryption algorithm $\mathsf{Enc}(pp, ek_{satt}, \mu, ratt)$ is executed by a sender which takes as input the public parameter pp, the sender's encryption key ek_{satt}, a message μ and a specified receiver's attribute $ratt$, and outputs a ciphertext $ct_{ratt,satt}$.

$\mathsf{Dec}(pp, rk_{f_r}, pok_{f_s}, (ct_{ratt,satt}, ratt, satt)) \to \mu/\perp$: The decryption algorithm $\mathsf{Dec}(pp, rk_{f_r},$ $pok_{f_s},$ $(ct_{ratt,satt},$ $ratt, satt))$ takes as input the public parameter pp, a receiver private key rk_{f_r}, a policy key pok_{f_s} and a ciphertext $ct_{ratt,satt}$, and outputs a message μ or $\perp$.

Correctness. For all $(pp, msk) \leftarrow \mathsf{Setup}(1^\lambda, \ell, N)$, $ek_{satt} \leftarrow \mathsf{SKGen}$ $(pp,$ $msk,$ $satt)$, $rk_{f_r} \leftarrow \mathsf{RKGen}(pp, msk, f_r)$, $pok_{f_s} \leftarrow \mathsf{PoKGen}(pp, msk, f_s)$, we have

(1) If $f_r(ratt) = 0 \wedge f_s(satt) = 0$, then

$$\Pr[\mathsf{Dec}(pp, rk_{f_r}, pok_{f_s}, \mathsf{Enc}(pp, ek_{satt}, \mu, ratt), ratt, satt) = \mu] = 1 - negl(\lambda).$$

(2) Otherwise,

$$\Pr[\mathsf{Dec}(pp, rk_{f_r}, pok_{f_s}, \mathsf{Enc}(pp, ek_{satt}, \mu, ratt), ratt, satt) = \perp] = 1 - negl(\lambda).$$

3.2 Security Model

Since our scheme achieves a selective security, in this section, we only give the selective IND-CPA (sIND-CPA) security definition and authentication security definition of the key-policy bilateral access control ABE scheme. The sIND-CPA security captures the confidentiality of the message μ encapsulated in a ciphertext. The authentication security guarantees that a valid ciphertext can not be forged under a sender encryption key that is not generated via the KGC, that is, an uncertified sender encryption key cannot be used to produce a valid ciphertext.

Let Π be a key-policy bilateral access control ABE scheme, $\mathcal{A}$ a ppt adversary, and $\mathcal{C}$ a challenger, the selective IND-CPA security experiment $\mathsf{sIND\text{-}CPA}_{\mathcal{A}}^{\Pi}(\lambda)$ played between $\mathcal{A}$ and $\mathcal{C}$ is described as follows.

The sIND-CPA experiment $\mathsf{sIND\text{-}CPA}_{\mathcal{A}}^{\Pi}(\lambda)$.

The sIND-CPA security experiment consists of the following several phases.

Setup. In this phase, the adversary first claims a challenge receiver attribute and sender attribute pair $(ratt^*, satt^*)$. Then the challenger runs the setup algorithm $\mathsf{Setup}(1^\lambda, N)$ to generate the public parameter pp and the master secret key msk, and sends pp to the adversary.

Query 1. In this phase, the adversary can make a series of sender encryption key queries, receiver private key queries and policy-key queries.

- When the adversary $\mathcal{A}$ submits a sender attribute $satt$, the challenger $\mathcal{C}$ invokes the sender encryption key generation algorithm $\mathsf{SKGen}(pp, msk, satt)$ to generate a sender encryption key ek_{satt} and delivers it to $\mathcal{A}$.
- When the adversary $\mathcal{A}$ issues a receiver private key query for a policy f_r such that $f_r(ratt^*) \neq 0$, the challenger $\mathcal{C}$ runs the receiver private key generation algorithm $\mathsf{RKGen}(pp, msk, f_r)$ to generate a receiver private key rk_{f_r} and sends it to the adversary.
- When the adversary $\mathcal{A}$ commits a policy key query for a specified sender policy f_s such that $f_s(satt^*) \neq 0$, the challenger $\mathcal{C}$ executes the policy-key generation algorithm $\mathsf{PoKGen}(pp, msk, f_s)$ to generate a policy-key pok_{f_s}, and sends it to the adversary.

Challenge. In this phase, when the adversary submits a challenge message pair (μ_0^*, μ_1^*), the challenger first flips a random coin $b \leftarrow_\$ \{0,1\}$ and invokes the sender encryption key generation algorithm $\mathsf{SKGen}(pp, msk, satt^*)$ to generate a sender encryption key ek_{satt^*}. Then it computes the challenge ciphertext $ct^* \leftarrow$ Enc $(pp,\ ek_{satt^*},\ \mu_b^*,\ ratt^*)$, and sends it to the adversary.

Query 2. In this phase, the adversary is allowed to submit the same key queries as that in Query 1, the challenger responds them as in Query 1.

Guess. In this phase, the experiment outputs 1 if the adversary's guess $b' = b$, and returns 0 otherwise.

Definition 8. *Let* $\mathsf{Adv}_{\mathcal{A},\Pi}^{sind\text{-}cpa}(\lambda)$ *denote the advantage function that a ppt adversary* $\mathcal{A}$ *wins in the above sIND-CPA experiment,*

$$\mathsf{Adv}_{\mathcal{A},\Pi}^{sind\text{-}cpa}(\lambda) := |\mathsf{Pr}[\mathsf{sIND\text{-}CPA}_{\mathcal{A}}^{\Pi}(\lambda) = 1] - 1/2|.$$

If the advantage function $\mathsf{Adv}_{\mathcal{A},\Pi}^{sind\text{-}cpa}(\lambda)$ *is negligible in* λ*, we say that the key-policy bilateral access control ABE scheme* Π *satisfies sIND-CPA security.*

Next we describe the authentication security experiment $\mathsf{AUTH}_{\mathcal{A}}^{\Pi}(\lambda)$ played between a ppt adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.

The authentication experiment $\mathsf{AUTH}_{\mathcal{A}}^{\Pi}(\lambda)$**.**

The authentication security experiment consists of the following several phases.

Setup. In this phase, the challenger first runs the setup algorithm $\mathsf{Setup}(1^{\lambda}, \ell, N)$ to generate the public parameter pp and the master secret key msk, and sends pp to the adversary.

Query. In this phase, the adversary can issue the sender encryption key queries, the receiver private key queries and the policy key queries.

- When the adversary $\mathcal{A}$ submits a sender attribute $satt$, the challenger $\mathcal{C}$ invokes the sender encryption key generation algorithm $\mathsf{SKGen}(pp, msk, satt)$ to generate a sender encryption key ek_{satt} and delivers it to $\mathcal{A}$.
- When $\mathcal{A}$ issues a receiver private key query for a policy f_r, the challenger $\mathcal{C}$ runs the receiver private key generation algorithm $\mathsf{RKGen}(pp, msk, f_r)$ to generate a receiver private key rk_{f_r} and sends it to the adversary.
- When $\mathcal{A}$ commits a policy key query for a policy f_s, the challenger $\mathcal{C}$ executes the policy key generation algorithm $\mathsf{PokGen}(pp, msk, f_s)$ to generate a policy key pok_{f_s} and sends it to the adversary.

Forge. In this phase, the adversary outputs a forgery $((ct^*, ratt^*, satt^*), f_r^*, f_s^*)$. The challenger runs the receiver private key generation algorithm RKGen (pp, msk, f_r^*) to generate the receiver private key $rk_{f_r^*}$ and the policy key generation algorithm $\mathsf{PoKGen}(pp, msk, f_s^*)$ to generate the policy key $pok_{f_s^*}$. Then it computes $m \leftarrow \mathsf{Dec}(pp, rk_{f_r^*}, pok_{f_s^*}, (ct^*, ratt^*, satt^*))$. If $satt^* \notin \mathcal{Q}$ and $m \neq \perp$, the experiment outputs 1, otherwise, it outputs 0, where $\mathcal{Q}$ denotes the encryption key query set in the Query phase.

Definition 9. *Let* $\mathsf{Adv}_{\mathcal{A},\Pi}^{auth}(\lambda)$ *denote the advantage function that a ppt adversary* $\mathcal{A}$ *wins in the above authentication experiment,*

$$\mathsf{Adv}_{\mathcal{A},\Pi}^{auth}(\lambda) := |\mathsf{Pr}[\mathsf{AUTH}_{\mathcal{A}}^{\Pi}(\lambda) = 1]|.$$

If the advantage function $\mathsf{Adv}_{\mathcal{A},\Pi}^{auth}(\lambda)$ *is negligible in* λ*, we say that the bilateral access control ABE scheme* Π *satisfies authentication security.*

4 Definition of Key-Policy Attribute-Based Matchmaking Encryption

In this section, we give the definition of key-policy attribute-based matchmaking encryption (short for "AB-ME"). An AB-ME scheme given here consists of six ppt algorithms, i.e., $\mathsf{ABME} = (\mathsf{Setup}, \mathsf{SKGen}, \mathsf{RKGen}, \mathsf{PoKGen}, \mathsf{Enc}, \mathsf{Dec})$, which are defined as the same as those of bilateral access control ABE except that the inputs for decryption algorithm $\mathsf{ABME.Dec}$ remove the sender and receiver's attributes $(satt, ratt)$. The authentication security definition of AB-ME is the same as the authentication security definition of the bilateral access control ABE, and the privacy security definition is the same as the sIND-CPA definition of the bilateral access control ABE except that the former additionally includes the confidentiality of the sender and receiver attributes *when mismatch*. More specifically, in the privacy security experiment of AB-ME, the submitted challenge tuple by the adversary before the setup is $((ratt_0, satt_0), (ratt_1, satt_1))$ instead of $(ratt^*, satt^*)$. We stress that the proposed AB-ME scheme in the following section achieves a *weak* security, i.e., the privacy of the sender and the receiver is only guaranteed when there is a mismatch. To facilitate a clearer understanding, we redescribed the detailed syntax and security definition of AB-ME in Appendix B.

5 Construction of Key-Policy Bilateral Access Control ABE

In this section, we present a construction of key-policy bilateral access control ABE with sIND-CPA and authentication security from a preimage sampling function and an extended ABE scheme in [8] with the help of the matrix embedding technique, where the preimage sampling function is used to generate an intermediate signature on the encrypted message μ during encryption, and the matrix embedding technique employs its homomorphic properties to realize the bilateral access control functionalities for the proposed scheme. Then we analyze the parameter setting and correctness of the scheme in subsection 5.2 and present its security proof in subsection 5.3 under the LWE and SIS assumptions.

5.1 Construction

The concrete construction of bilateral access control ABE is described as follows.

$\mathsf{Setup}(1^\lambda, \ell, N)$: This algorithm first invokes the trapdoor generation algorithm $\mathsf{GenTrap}(1^n, m, q) \to (\mathbf{A}, \mathbf{T_A})$ to generate a random matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and a trapdoor $\mathbf{T_A} \in \mathbb{Z}_q^{m\times m}$, then invokes again the trapdoor generation algorithm $\mathsf{GenTrap}(1^n, m, q) \to (\mathbf{A}', \mathbf{T_{A'}})$ to generate another random matrix $\mathbf{A}' \in \mathbb{Z}_q^{n\times m}$ and trapdoor $\mathbf{T_{A'}} \in \mathbb{Z}_q^{m\times m}$. Next, it picks two cryptographic hash functions $H_1 : \{0,1\}^* \to \mathbb{Z}_q^{n\times m}$, $H_2 : \{0,1\}^* \to \mathbb{Z}_q^n$, N random vectors $\mathbf{u}_1, \cdots, \mathbf{u}_N \in \mathbb{Z}_q^n$ and 2ℓ random matrices $\mathbf{B}_1, \cdots, \mathbf{B}_\ell, \mathbf{D}_1, \cdots, \mathbf{D}_\ell \leftarrow_\$ \mathbb{Z}_q^{n\times m}$

where $N = \ell_0 + \ell_1$ for $N, \ell_0, \ell_1 \in \mathbb{N}$. It sets the public parameter as $pp = (\mathbf{A}, \mathbf{A}', \mathbf{u}_1, \cdots, \mathbf{u}_N, \mathbf{B}_1, \cdots, \mathbf{B}_\ell, \mathbf{D}_1, \cdots, \mathbf{D}_\ell, H_1, H_2)$ and the master secret key as $msk = (\mathbf{T_A}, \mathbf{T_{A'}})$. Finally, it returns the public parameter/master secret key pair (pp, msk).

$\mathsf{SKGen}(pp, msk, satt \in \{0,1\}^\ell)$: This algorithm first computes $\mathbf{D}_{satt} = (\mathbf{D}_1 + satt_1\mathbf{G}| \cdots |\mathbf{D}_\ell + satt_\ell\mathbf{G})$, $\mathbf{H}_{satt} = H_1(pp, \mathbf{D}_{satt})$, $\mathbf{F}_{satt} = [\mathbf{A}'|\mathbf{H}_{satt}]$, $\mathbf{T} = \mathsf{DelTrap}(\mathbf{F}_{satt}, \mathbf{T_{A'}}, \delta)$. Then it sets the encryption key as $ek_{satt} = (\mathbf{T}, \mathbf{D}_{satt}, satt)$ and returns ek_{satt}.

$\mathsf{RKGen}(pp, msk, f_r)$: If $\mathbf{u}_{1,1}, \cdots, \mathbf{u}_{1,N}$ are not defined, the algorithm first picks N random vectors $\mathbf{u}_{1,1}, \cdots, \mathbf{u}_{1,N} \leftarrow_\$ \mathbb{Z}_q^n$, and computes $\mathbf{B}_{f_r} = \mathsf{Eval}_{pp}(f_r, (\mathbf{B}_1, \cdots, \mathbf{B}_\ell))$, $\mathbf{r}^i_{f_r} = \mathsf{SampleLeft}(\mathbf{A}, \mathbf{B}_{f_r}, \mathbf{T_A}, \mathbf{u}_{1,i}, \delta)$ for each $i \in [N]$ such that $(\mathbf{A}|\mathbf{B}_{f_r}) \cdot \mathbf{r}^i_{f_r} = \mathbf{u}_{1,i}$. It then sets the receiver private key as $rk_{f_r} = \{\mathbf{r}^i_{f_r}\}_{i\in[N]}$ and returns rk_{f_r}.

$\mathsf{PoKGen}(pp, msk, f_s)$: If $\mathbf{u}_{1,1}, \cdots, \mathbf{u}_{1,N}$ are not defined, the algorithm first picks N random vectors $\mathbf{u}_{1,1}, \cdots, \mathbf{u}_{1,N} \leftarrow_\$ \mathbb{Z}_q^n$ and sets $\mathbf{u}_{2,i} = \mathbf{u}_i - \mathbf{u}_{1,i}$ for each $i \in [N]$, then computes $\mathbf{D}_{f_s} = \mathsf{Eval}_{pp}(f_s, (\mathbf{D}_1, \cdots, \mathbf{D}_\ell))$, $\mathbf{r}^i_{f_s} = \mathsf{SampleLeft}(\mathbf{A}, \mathbf{D}_{f_s}, \mathbf{T_A}, \mathbf{u}_{2,i}, \delta)$ for each $i \in [N]$ such that $(\mathbf{A}|\mathbf{D}_{f_s}) \cdot \mathbf{r}^i_{f_s} = \mathbf{u}_{2,i}$. Finally it sets the policy private key as $pok_{f_s} = \{\mathbf{r}^i_{f_s}\}_{i\in[N]}$, and returns pok_{f_s}.

$\mathsf{Enc}(pp, ek_{satt}, \mu, ratt)$: This algorithm first parses ek_{satt} into $(\mathbf{T}, \mathbf{D}_{satt}, satt)$, and picks $\mathbf{s}_0 \leftarrow_\$ \mathbb{Z}_q^n$, $\mathbf{e}_0 \leftarrow_\$ \chi^m$, $x_1, \cdots, x_N \leftarrow_\$ \chi$, and for each $i \in [\ell]$, chooses $\mathbf{S}_{1,i}, \mathbf{S}_{2,i} \leftarrow_\$ \{-1, +1\}^{m\times m}$. It then sets

$$\mathbf{e}_1 = (\mathbf{S}_{1,1}| \cdots |\mathbf{S}_{1,\ell})^T \cdot \mathbf{e}_0 = (\mathbf{S}_1)^T \cdot \mathbf{e}_0 \in \mathbb{Z}_q^{\ell m},$$

$$\mathbf{e}_2 = (\mathbf{S}_{2,1}| \cdots |\mathbf{S}_{2,\ell})^T \cdot \mathbf{e}_0 = (\mathbf{S}_2)^T \cdot \mathbf{e}_0 \in \mathbb{Z}_q^{\ell m},$$

and computes $\mathbf{h}_\mu = H_2(pp, satt, \mu|ratt)$, $\mathbf{H}_{satt} = H_1(pp, \mathbf{D}_{satt})$, $\mathbf{F}_{satt} = [\mathbf{A}'|\mathbf{H}_{satt}]$, $\boldsymbol{\sigma} \leftarrow \mathsf{PreSamp}(\mathbf{F}_{satt}, \mathbf{T}, \mathbf{h}_\mu, \delta)$ such that $\mathbf{F}_{satt} \cdot \boldsymbol{\sigma} = \mathbf{h}_\mu$. Next, it defines $\mathbf{D}_{ratt} = (\mathbf{B}_1 + ratt_1\mathbf{G}| \cdots |\mathbf{B}_\ell + ratt_\ell\mathbf{G})$, and computes $\mathbf{c}_0 = \mathbf{A}^T\mathbf{s}_0 + \mathbf{e}_0$, $\mathbf{c}_1 = \mathbf{D}^T_{ratt}\mathbf{s}_0 + \mathbf{e}_1$, $\mathbf{c}_2 = \mathbf{D}^T_{satt}\mathbf{s}_0 + \mathbf{e}_2$. Then encode the signature $\boldsymbol{\sigma} \in \mathbb{Z}_q^{2m}$ into a binary string $\sigma' = [\boldsymbol{\sigma}]_2$, where $[\boldsymbol{\sigma}]_2$ denotes the encoding from the vector $\boldsymbol{\sigma} \in \mathbb{Z}_q^{2m}$ into a length-ℓ_1 binary string $\sigma' \in \{0,1\}^{\ell_1}$. Next, it computes

$$c_{3,1} = \mathbf{u}_1^T\mathbf{s}_0 + x_1 + \mu_1\lceil\frac{q}{2}\rceil, \cdots, c_{3,\ell_0} = \mathbf{u}_{\ell_0}^T\mathbf{s}_0 + x_{\ell_0} + \mu_{\ell_0}\lceil\frac{q}{2}\rceil,$$

$$c_{3,\ell_0+1} = \mathbf{u}_{\ell_0+1}^T\mathbf{s}_0 + x_{\ell_0+1} + \sigma'_1\lceil\frac{q}{2}\rceil, \cdots, c_{3,N} = \mathbf{u}_N^T\mathbf{s}_0 + x_N + \sigma'_{\ell_1}\lceil\frac{q}{2}\rceil,$$

where $|\mu| = \ell_0$, $|\sigma'| = \ell_1$, and $\ell_0 + \ell_1 = N$. It sets the ciphertext as $ct = (\mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_2, \{c_{3,i}\}_{i\in[N]})$ and returns ct.

$\mathsf{Dec}(pp, rk_{f_r}, pok_{f_s}, (ct, ratt, satt))$: This algorithm first parses the ciphertext ct into $(\mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_2, \{c_{3,i}\}_{i\in[N]})$, and $\mathbf{c}_1$, $\mathbf{c}_2$ into $\{\mathbf{c}_{1,i}\}_{i\in[\ell]}$, $\{\mathbf{c}_{2,i}\}_{i\in[\ell]}$ respectively. It then computes

$$\mathbf{c}_{f_r} = \mathsf{Eval}_{ct}(f_r, \{(\mathbf{B}_i, ratt_i\mathbf{G}, \mathbf{c}_{1,i})\}_{i\in[\ell]}),$$

$$\mathbf{c}_{f_s} = \mathsf{Eval}_{ct}(f_s, \{(\mathbf{D}_i, satt_i\mathbf{G}, \mathbf{c}_{2,i})\}_{i\in[\ell]}),$$

and

$$w_i = c_{3,i} - (\mathbf{r}^i_{f_r})^T \begin{bmatrix} \mathbf{c}_0 \\ \mathbf{c}_{f_r} \end{bmatrix} - (\mathbf{r}^i_{f_s})^T \begin{bmatrix} \mathbf{c}_0 \\ \mathbf{c}_{f_s} \end{bmatrix}.$$

If $|w_i - \lceil \frac{q}{2} \rceil| < \frac{q}{4}$, it sets $m'_i = 1$, otherwise it sets $m'_i = 0$. Then it sets $\mu' = m'_0 \cdots m'_{\ell_0}$, $\sigma' = m'_{\ell_0+1} \cdots m'_N$, $\boldsymbol{\sigma} = [\sigma']_v$. Note that $[\sigma']_v$ denotes the encoding from the binary string $\sigma' \in \{0,1\}^{\ell_1}$ into a vector $\boldsymbol{\sigma} \in \mathbb{Z}_q^{2m}$. Next, it computes $\mathbf{h}_{\mu'} = H_2(pp, satt, \mu'|ratt)$, $\mathbf{D}_{satt} = (\mathbf{D}_1 + satt_1\mathbf{G}|\cdots|\mathbf{D}_\ell + satt_\ell\mathbf{G})$, $\mathbf{H}_{satt} = H_1(pp, \mathbf{D}_{satt})$, $\mathbf{F}_{satt} = [\mathbf{A}'|\mathbf{D}_{satt}]$, and verifies whether

$$\mathbf{F}_{satt}\cdot\boldsymbol{\sigma} \overset{?}{=} \mathbf{h}_{\mu'}.$$

If yes, it returns μ', and $\perp$ otherwise.

Remark 1. Note that it is necessary to employ distinct masking vectors to individually mask each bit of the payload in the proposed bilateral access control ABE scheme. Since from the definition of the LWE (Learning With Errors) assumption, the public vector $\mathbf{u}_i$ and noisy x_i in each LWE instance $(\mathbf{u}_i, \mathbf{v}_i = \mathbf{u}_i\mathbf{s}_0 + x_i)$ are random. If we use the same vector to mask each bit in the plaintext, this would compromise the randomness of the vector $\mathbf{u}_i$ in the LWE assumption, thereby introducing potential security vulnerabilities to the proposed scheme.

In addition, note that providing the signature in plaintext and enforcing that payload recovery still requires satisfying both policies, which seems sufficient to achieve the intended security, but this is not the case. Since the signature scheme itself does not provide confidentiality for the signed message, which may leak the information of the signed message. Therefore, encrypting the signature on the underlying payload not only does not introduce redundancy but is in fact essential.

5.2 Parameter Setting and Correctness

Parameter Setting. To ensure the correct running and the security requirements of the scheme, we set the parameters $\lambda, N, \ell, m, \delta$ as

$$\lambda = n,\ \ N = poly(n),\ \ \ell = poly(n),\ \ m = \Theta(n\log q),\ \ \delta = O(\sqrt{m}(m{+}1)^d\sqrt{\log m}).$$

Correctness. The correctness of the proposed bilateral access control ABE scheme Π above is discussed as follows. First, by the definition of the algorithm Eval_{ct}, we have

$$\mathbf{c}_{f_r} = (\mathbf{B}_{f_r} + f_r(ratt)\mathbf{G})^T\mathbf{s}_0 + \mathbf{e}_{f_r},$$

$$\mathbf{c}_{f_s} = (\mathbf{D}_{f_s} + f_s(satt)\mathbf{G})^T\mathbf{s}_0 + \mathbf{e}_{f_s}.$$

In addition, by the correctness definition of the bilateral access control ABE scheme, if the scheme can be decrypted correctly, the conditions $f_r(ratt) = 0$ and $f_s(satt) = 0$ hold, hence

$$(\mathbf{r}^i_{f_r})^T \begin{bmatrix} \mathbf{c}_0 \\ \mathbf{c}_{f_r} \end{bmatrix} = (\mathbf{r}^i_{f_r})^T \begin{bmatrix} \mathbf{A}^T\mathbf{s}_0 + \mathbf{e}_0 \\ (\mathbf{B}_{f_r})^T\mathbf{s}_0 + \mathbf{e}_{f_r} \end{bmatrix} = ((\mathbf{A}|\mathbf{B}_{f_r})\mathbf{r}^i_{f_r})^T\mathbf{s}_0 + (\mathbf{r}^i_{f_r})^T \begin{bmatrix} \mathbf{e}_0 \\ \mathbf{e}_{f_r} \end{bmatrix}$$

$$= (\mathbf{u}_{1,i})^T\mathbf{s}_0 + (\mathbf{r}^i_{f_r})^T \begin{bmatrix} \mathbf{e}_0 \\ \mathbf{e}_{f_r} \end{bmatrix},$$

$$(\mathbf{r}^i_{f_s})^T \begin{bmatrix} \mathbf{c}_0 \\ \mathbf{c}_{f_s} \end{bmatrix} = (\mathbf{r}^i_{f_s})^T \begin{bmatrix} \mathbf{A}^T\mathbf{s}_0 + \mathbf{e}_0 \\ (\mathbf{D}_{f_s})^T\mathbf{s}_0 + \mathbf{e}_{f_s} \end{bmatrix} = ((\mathbf{A}|\mathbf{D}_{f_s})\mathbf{r}^i_{f_s})^T\mathbf{s}_0 + (\mathbf{r}^i_{f_s})^T \begin{bmatrix} \mathbf{e}_0 \\ \mathbf{e}_{f_s} \end{bmatrix}$$

$$= (\mathbf{u}_{2,i})^T\mathbf{s}_0 + (\mathbf{r}^i_{f_s})^T \begin{bmatrix} \mathbf{e}_0 \\ \mathbf{e}_{f_s} \end{bmatrix}.$$

Then, it computes

$$w_i = c_{3,i} - (\mathbf{r}^i_{f_r})^T \begin{bmatrix} \mathbf{c}_0 \\ \mathbf{c}_{f_r} \end{bmatrix} - (\mathbf{r}^i_{f_s})^T \begin{bmatrix} \mathbf{c}_0 \\ \mathbf{c}_{f_s} \end{bmatrix}$$

$$= \mathbf{u}_i^T\mathbf{s}_0 + x_i + m_i\lceil\frac{q}{2}\rceil - ((\mathbf{u}_{1,i})^T\mathbf{s}_0 + (\mathbf{r}^i_{f_r})^T \begin{bmatrix} \mathbf{e}_0 \\ \mathbf{e}_{f_r} \end{bmatrix}) - ((\mathbf{u}_{2,i})^T\mathbf{s}_0 + (\mathbf{r}^i_{f_s})^T \begin{bmatrix} \mathbf{e}_0 \\ \mathbf{e}_{f_s} \end{bmatrix})$$

$$= x_i + m'_i\lceil\frac{q}{2}\rceil - (\mathbf{r}^i_{f_r})^T \begin{bmatrix} \mathbf{e}_0 \\ \mathbf{e}_{f_r} \end{bmatrix} - (\mathbf{r}^i_{f_s})^T \begin{bmatrix} \mathbf{e}_0 \\ \mathbf{e}_{f_s} \end{bmatrix}.$$

Since

$$||\mathbf{e}_{f_r}|| \le 20\sqrt{m}Q\sqrt{m}(m+1)^d = 20Qm(m+1)^d,$$
$$||\mathbf{e}_{f_s}|| \le 20\sqrt{m}Q\sqrt{m}(m+1)^d = 20Qm(m+1)^d,$$

we have

$$\left|\left|\begin{bmatrix} \mathbf{e}_0 \\ \mathbf{e}_{f_r} \end{bmatrix}\right|\right| \le 20Qm(m+1)^d + Q\sqrt{m}, \quad \left|\left|\begin{bmatrix} \mathbf{e}_0 \\ \mathbf{e}_{f_s} \end{bmatrix}\right|\right| \le 20Qm(m+1)^d + Q\sqrt{m}.$$

Again since $|x_i| \le Q$, $||\mathbf{r}^i_{f_r}|| \le \delta\sqrt{2m}$, $||\mathbf{r}^i_{f_s}|| \le \delta\sqrt{2m}$, hence when $m'_i = 1$, we have

$$|w_i - \lceil\frac{q}{2}\rceil| \le |x_i| + ||\ (\mathbf{r}_{f_r})^T|| \cdot \left|\left|\begin{bmatrix} \mathbf{e}_0 \\ \mathbf{e}_{f_r} \end{bmatrix}\right|\right| + ||(\mathbf{r}_{f_s})^T|| \cdot \left|\left|\begin{bmatrix} \mathbf{e}_0 \\ \mathbf{e}_{f_s} \end{bmatrix}\right|\right|$$

$$\le Q + \delta\sqrt{2m}(20Qm(m+1)^d + Q\sqrt{m}) + \delta\sqrt{2m}(20Qm(m+1)^d + Q\sqrt{m}).$$

$$\le O(Qm^2(m+1)^{2d}\sqrt{\log m}) < \frac{q}{4}.$$

When $m'_i = 0$,

$$|w_i - \lceil \frac{q}{2} \rceil| \geq \lceil \frac{q}{2} \rceil - (|x_i| + \| (\mathbf{r}_{f_r})^T \|_2 \cdot \left\| \begin{bmatrix} \mathbf{e}_0 \\ \mathbf{e}_{f_r} \end{bmatrix} \right\| + \|(\mathbf{r}_{f_s})^T\|_2 \cdot \left\| \begin{bmatrix} \mathbf{e}_0 \\ \mathbf{e}_{f_s} \end{bmatrix} \right\|)$$

$$> \lceil \frac{q}{2} \rceil - \frac{q}{4} > \frac{q}{4}.$$

Then set $\mu' = m'_0 \cdots m'_{\ell_0}$, $\sigma' = m'_{\ell_0+1} \cdots m'_N$, $\boldsymbol{\sigma} = [\sigma']_v$. Next compute $\mathbf{h}_{\mu'} = H_2(pp, satt, \mu'|ratt)$, $\mathbf{D}_{satt} = (\mathbf{D}_1 + satt_1\mathbf{G}| \cdots |\mathbf{D}_\ell + satt_\ell\mathbf{G})$, $\mathbf{H}_{satt} = H_1(pp, \mathbf{D}_{satt})$, $\mathbf{F}_{satt} = [\mathbf{A}'|\mathbf{H}_{satt}]$ and verify whether $\mathbf{F}_{satt} \cdot \boldsymbol{\sigma} \overset{?}{=} \mathbf{h}_{\mu'}$ holds. If yes, μ' is returned.

5.3 Security Proof

In this section, we give the sIND-CPA and authentication security proof of the proposed bilateral access control ABE where the former is proved via Theorem 3 under the LWE assumption and the latter is proved via Theorem 4 under the SIS assumption.

Theorem 3. *The key-policy bilateral access control ABE scheme Π in Section 5.1 achieves sIND-CPA security under the LWE assumption.*

To prove theorem 3, we assume that if there exists a ppt adversary $\mathcal{A}$ that can break the sIND-CPA security of the scheme Π, then we can construct another ppt algorithm $\mathcal{A}_{lwe}$ that can employ $\mathcal{A}$ to break the LWE assumption. The proof is given via first defining a series of games from game G_0 which denotes the initial experiment, to game G_6 in which the ciphertext is independent of the encrypted message μ, then giving the hybrid arguments about the indistinguishability between any two adjacent games. We use $\mathsf{Pr}[\mathsf{G}_i]$ to denote the probability that the adversary $\mathcal{A}$ succeeds in game G_i for $i \in \{0, 1, \cdots, 6\}$. The definitions for these games are described as follows. In addition, note that at the beginning of each game, we assume that the adversary has submitted a challenge receiver attribute and sender attribute pair $(ratt^*, satt^*)$ beforehand.

G_0 : This game is the real sIND-CPA security experiment.

G_1 : This game is the same as G_0 except that, in game G_1 we replace the generating mode of matrices $\mathbf{B}_i$ and $\mathbf{D}_i$ for $i \in [\ell]$ in the setup phase and the ciphertext components $\mathbf{c}_1^*$ and $\mathbf{c}_2^*$ in the challenge phase. More specifically, we first choose 2ℓ random short matrices $\mathbf{S}^*_{1,1}, \cdots, \mathbf{S}^*_{1,\ell} \leftarrow_\$ \{-1,1\}^{m\times m}$, $\mathbf{S}^*_{2,1}, \cdots, \mathbf{S}^*_{2,\ell} \leftarrow_\$ \{-1,1\}^{m\times m}$, and set $\mathbf{B}_i = \mathbf{A}\mathbf{S}^*_{1,i} - ratt^*_i\mathbf{G}$, $\mathbf{D}_i = \mathbf{A}\mathbf{S}^*_{2,i} - satt^*_i\mathbf{G}$ for each $i \in [\ell]$, then compute $\mathbf{c}_1^* = (\mathbf{S}_1^*)^T\mathbf{c}_0^*$ and $\mathbf{c}_2^* = (\mathbf{S}_2^*)^T\mathbf{c}_0^*$, where $\mathbf{S}_1^* = (\mathbf{S}^*_{1,1}| \cdots |\mathbf{S}^*_{1,\ell})$ and $\mathbf{S}_2^* = (\mathbf{S}^*_{2,1}| \cdots |\mathbf{S}^*_{2,\ell})$.

G_2 : This game is the same as G_1 except that, in game G_2 we change the generating mode of the matrix $\mathbf{A}$. Concretely, we pick a random $\mathbf{A} \leftarrow_\$ \mathbb{Z}_q^{n\times m}$ rather than generating it via the trapdoor generation algorithm $(\mathbf{A}, \mathbf{T_A}) \leftarrow \mathsf{TrapGen}(1^n, m, q)$. In addition, the receiver private key and policy key are generated by the right sampling algorithm $\mathsf{SampleRight}$. Namely, each vector

$\mathbf{r}^i_{f_r}$ in the receiver private key $rk_{f_r} = \{\mathbf{r}^i_{f_r}\}_{i\in[N]}$ for policy f_r ($f_r(ratt^*) \neq 0$) and each vector $\mathbf{r}^i_{f_s}$ in the policy key $pok_{f_s} = \{\mathbf{r}^i_{f_s}\}_{i\in[N]}$ for policy f_s ($f_s(satt^*) \neq 0$) are computed as

$$\mathbf{r}^i_{f_r} \leftarrow \mathsf{SampleRight}(\mathbf{A}, -f_r(ratt^*)\mathbf{G}, \mathbf{S}^*_{1,f_r}, \mathbf{T}_{\mathbf{G}}, \mathbf{u}_{1,i}, \delta),$$

$$\mathbf{r}^i_{f_s} \leftarrow \mathsf{SampleRight}(\mathbf{A}, -f_s(satt^*)\mathbf{G}, \mathbf{S}^*_{2,f_s}, \mathbf{T}_{\mathbf{G}}, \mathbf{u}_{2,i}, \delta),$$

respectively such that $(\mathbf{A}|\mathbf{B}_{f_r})\mathbf{r}^i_{f_r} = \mathbf{u}_{1,i}$, $(\mathbf{A}|\mathbf{D}_{f_s})\mathbf{r}^i_{f_s} = \mathbf{u}_{2,i}$, where $\mathbf{u}_{1,i} \leftarrow_\$ \mathbb{Z}_q^n$, $\mathbf{u}_{2,i} = \mathbf{u}_1 - \mathbf{u}_{1,i}$, and

$$\mathbf{B}_{f_r} = \mathsf{Eval}_{pp}(f_r, (\mathbf{A}\mathbf{S}^*_{1,1} - ratt^*_1\mathbf{G}, \cdots, \mathbf{A}\mathbf{S}^*_{1,\ell} - ratt^*_\ell\mathbf{G})),$$

$$\mathbf{D}_{f_s} = \mathsf{Eval}_{pp}(f_s, (\mathbf{A}\mathbf{S}^*_{2,1} - satt^*_1\mathbf{G}, \cdots, \mathbf{A}\mathbf{S}^*_{2,\ell} - satt^*_\ell\mathbf{G})),$$

$$\mathbf{S}^*_{1,f_r} = \mathsf{Eval}_{sim}(f_r, \{(\mathbf{S}^*_{1,i}, ratt^*_i)\}, \mathbf{A}) \quad s.t. \quad \mathbf{A}\mathbf{S}^*_{1,f_r} - f_r(ratt^*)\mathbf{G} = \mathbf{B}_{f_r},$$

$$\mathbf{S}^*_{2,f_s} = \mathsf{Eval}_{sim}(f_s, \{(\mathbf{S}^*_{2,i}, satt^*_i)\}, \mathbf{A}) \quad s.t. \quad \mathbf{A}\mathbf{S}^*_{2,f_s} - f_s(satt^*)\mathbf{G} = \mathbf{D}_{f_s}.$$

G_3 : This game is the same as G_2 except that in game G_3, the ciphertext components $\mathbf{c}^*_0 \leftarrow_\$ \mathbb{Z}_q^m$, $c^*_{3,i} \leftarrow_\$ \mathbb{Z}_q$ for each $i \in [N]$ are chosen uniformly from the corresponding ciphertext space.

G_4 : This game is identical to G_3 except that in G_4, the matrix $\mathbf{A}$ is generated by invoking the trapdoor generation algorithm $(\mathbf{A}, \mathbf{T}_{\mathbf{A}}) \leftarrow \mathsf{GenTrap}(1^n, m, q)$, and the receiver private key and the policy key are generated using the left sampling algorithm $\mathsf{SampleLeft}$ under the trapdoor $\mathbf{T}_{\mathbf{A}}$ rather than the right sampling algorithm $\mathsf{SampleRight}$ under the trapdoor $\mathbf{T}_{\mathbf{A}}$.

G_5 : This game is identical to G_4 except that in game G_5, the ciphertext $ct^* = (\mathbf{c}^*_0, \mathbf{c}^*_1, \mathbf{c}^*_2, \{c^*_{3,i}\}_{i\in[N]})$ are all chosen uniformly from the corresponding ciphertext space.

G_6 : This game is the same as G_5 except that in game G_6 the matrices $\mathbf{B}_i$ and $\mathbf{D}_i$ for each $i \in [\ell]$ are all chosen randomly, which are identical to that in the initial experiment.

Lemma 7. *Games G_0 and G_1 are statistically indistinguishable, namely,*

$$\Pr[\mathsf{G}_0] \overset{s}{\approx} \Pr[\mathsf{G}_1].$$

Proof. Note that in game G_0, the matrices $\mathbf{B}_1, \cdots, \mathbf{B}_\ell$ and $\mathbf{D}_1, \cdots, \mathbf{D}_\ell$ are chosen randomly from the corresponding space, while in game G_1, they are generated as $\mathbf{B}_i = \mathbf{A}\mathbf{S}^*_{1,i} - ratt^*_i\mathbf{G}$ and $\mathbf{D}_i = \mathbf{A}\mathbf{S}^*_{2,i} - satt^*_i\mathbf{G}$ for $i \in [\ell]$, where $\mathbf{S}^*_{1,i}, \mathbf{S}^*_{2,i} \leftarrow_\$ \{-1,1\}^{m\times m}$ In addition, in game G_0, the ciphertext components $\mathbf{c}^*_1$ and $\mathbf{c}^*_2$ are generated as $\mathbf{c}^*_1 = \mathbf{D}^T_{ratt^*}\mathbf{s}_0 + \mathbf{e}_1$ and $\mathbf{c}^*_2 = \mathbf{D}^T_{satt^*}\mathbf{s}_0 + \mathbf{e}_2$, while in game G_1, they are generated via $\mathbf{c}^*_1 = (\mathbf{S}^*_1)^T\mathbf{c}^*_0$ and $\mathbf{c}^*_2 = (\mathbf{S}^*_2)^T\mathbf{c}^*_0$, where $\mathbf{e}_1 = (\mathbf{S}^*_1)^T \cdot \mathbf{e}_0 \in \mathbb{Z}_q^{\ell m}$, $\mathbf{e}_2 = (\mathbf{S}^*_2)^T \cdot \mathbf{e}_0 \in \mathbb{Z}_q^{\ell m}$, $\mathbf{S}^*_1 = (\mathbf{S}^*_{1,1}|\cdots|\mathbf{S}^*_{1,\ell})$ and $\mathbf{S}^*_2 = (\mathbf{S}^*_{2,1}|\cdots|\mathbf{S}^*_{2,\ell})$.

We now argue that the following joint distribution of the public matrices $\mathbf{A}$, $\mathbf{B}_i$, $\mathbf{D}_i$ and ciphertext components $\mathbf{c}^*_1$, $\mathbf{c}^*_2$ is statistically indistinguishable between two games.

Now, observe that by Lemma 4, we have

$$(\mathbf{A}, \mathbf{A}\mathbf{S}_{1,i}^* - ratt_i^*\mathbf{G}, (\mathbf{S}_{1,i}^*)^T\mathbf{e}_0) \overset{s}{\approx} (\mathbf{A}, \mathbf{B}_i, (\mathbf{S}_{1,i}^*)^T\mathbf{e}_0).$$

Moreover, it also holds for $\mathbf{D}_i$, namely

$$(\mathbf{A}, \mathbf{A}\mathbf{S}_{2,i}^* - satt_i^*\mathbf{G}, (\mathbf{S}_{2,i}^*)^T\mathbf{e}_0) \overset{s}{\approx} (\mathbf{A}, \mathbf{D}_i, (\mathbf{S}_{2,i}^*)^T\mathbf{e}_0).$$

Hence, for all randomly and independently chosen $\mathbf{S}_{1,i}^*$ and $\mathbf{S}_{2,i}^*$ with $i \in [\ell]$, the following holds

$$(\mathbf{A}, \{\mathbf{A}\mathbf{S}_{1,i}^* - ratt_i^*\mathbf{G}\}, \{\mathbf{A}\mathbf{S}_{2,i}^* - satt_i^*\mathbf{G}\}, \{(\mathbf{S}_{1,i}^*)^T\mathbf{e}_0\}, \{(\mathbf{S}_{2,i}^*)^T\mathbf{e}_0\})$$

$$\overset{s}{\approx} (\mathbf{A}, \{\mathbf{B}_i\}, \{\mathbf{D}_i\}, \{(\mathbf{S}_{1,i}^*)^T\mathbf{e}_0\}, \{(\mathbf{S}_{2,i}^*)^T\mathbf{e}_0\}).$$

Note that the ciphertext components $\mathbf{c}_{1,i}^*$ (in $\mathbf{c}_1^* = \{\mathbf{c}_{1,i}^*\}_{i\in[\ell]}$) and $\mathbf{c}_{2,i}^*$ (in $\mathbf{c}_2^* = \{\mathbf{c}_{2,i}^*\}_{i\in[\ell]}$) for $i \in [\ell]$ are derived via adding $(\mathbf{B}_i + ratt_i^*\mathbf{G})^T\mathbf{s}_0$ and $(\mathbf{D}_i + satt_i^*\mathbf{G})^T\mathbf{s}_0$ to $(\mathbf{S}_{1,i}^*)^T\mathbf{e}_0$ and $(\mathbf{S}_{2,i}^*)^T\mathbf{e}_0$ respectively. In addition, since the sender encryption key, the receiver private key and the policy private key are all generated identically between two games, the public parameters, the ciphertext and all keys are statistically close in both hybrids.

Lemma 8. *Games* G_1 *and* G_2 *are statistically indistinguishable, namely,*

$$\mathsf{Pr}[\mathsf{G}_1] \overset{s}{\approx} \mathsf{Pr}[\mathsf{G}_2].$$

Proof. Observe that the difference between games G_1 and G_2 is the way of generating the receiver private key and the policy key. More precisely, in game G_1, these private keys are generated by invoking the left sampling algorithm SampleLeft, while in game G_2, they are computed by invoking the right sampling algorithm SampleRight. By Theorems 1 and 2, the distributions of the two keys in both games are statistically indistinguishable.

Lemma 9. *Games* G_2 *and* G_3 *are computationally indistinguishable under the LWE assumption, namely,*

$$\mathsf{Pr}[\mathsf{G}_2] - \mathsf{Pr}[\mathsf{G}_3] \leq \mathsf{Adv}_{\mathcal{A}_{lwe}}^{lwe}(\lambda).$$

Proof. The difference between games G_2 and G_3 can be reduced to the LWE assumption. The formal reduction is described as follows. Assume that there is a ppt adversary $\mathcal{A}$ that can distinguish games G_2 and G_3, then we can construct a ppt algorithm $\mathcal{A}_{lwe}$ that can employ $\mathcal{A}$ to break the LWE assumption.

At the beginning of the experiment, assume that the algorithm $\mathcal{A}_{lwe}$ has received a challenge $(\mathbf{A}, \{\mathbf{u}_i\}_{i\in[N]}, \mathbf{v}, \{z_i\}_{i\in[N]})$ from its challenger, where $\mathbf{v} = \mathbf{A}^T\mathbf{s}_0 + \mathbf{e}_0$, $\{z_i = \mathbf{u}_i^T\mathbf{s}_0 + x_i\}_{i\in[N]}$ or $\mathbf{v}$, $\{z_i\}_{i\in[N]}$ are all chosen at random. The algorithm $\mathcal{A}_{lwe}$ first invokes the trapdoor generation algorithm GenTrap to generate a matrix $\mathbf{A}'$ and a trapdoor $\mathbf{T}_{\mathbf{A}'}$, and then generates other parameter components $\mathbf{B}_1, \cdots, \mathbf{B}_\ell$, $\mathbf{D}_1, \cdots, \mathbf{D}_\ell$, H_1 and H_2 as in the setup algorithm of the game G_2. Next, it sets the parameter components $\mathbf{A}$, $\mathbf{u}_1, \cdots, \mathbf{u}_N$ as the received challenge components $\mathbf{A}$, $\mathbf{u}_1, \cdots, \mathbf{u}_N$. Finally, it sends the generated public parameter $pp = (\mathbf{A}, \mathbf{A}', \mathbf{u}_1, \cdots, \mathbf{u}_N, \mathbf{B}_1, \cdots, \mathbf{B}_\ell, \mathbf{D}_1, \cdots, \mathbf{D}_\ell, H_1, H_2)$ to the adversary $\mathcal{A}$.

- For the sender encryption key query, the receiver private key query and the policy key query, the algorithm $\mathcal{A}_{lwe}$ responds in the same way as in game G_2.
- When receiving the challenge query (μ_0, μ_1), the algorithm $\mathcal{A}_{lwe}$ first picks a random $b \in \{0,1\}$ and computes the intermediate ciphertext component $\boldsymbol{\sigma}^*$ ($\mathcal{A}_{lwe}$ can do this because it knows $\mathbf{T}_{\mathbf{A}'}$) as in game G_2, and encodes it into binary string σ'^*, and then generates other ciphertext components as follows. That is, it first sets $\mathbf{c}_0^* = \mathbf{v}$, then computes

$$c_{3,1}^* = z_1 + \mu_{b,1}\lceil \frac{q}{2} \rceil, c_{3,\ell_0} = z_{\ell_0} + \mu_{b,\ell_0}\lceil \frac{q}{2} \rceil,$$

$$c_{3,\ell_0+1}^* = z_{\ell_0+1} + \sigma_1'^*\lceil \frac{q}{2} \rceil, \cdots, c_{3,N} = z_N + \sigma_{\ell_1}'^*\lceil \frac{q}{2} \rceil,$$

  and

$$\mathbf{c}_1^* = \{\mathbf{c}_{1,i}^* = (\mathbf{S}_{1,i}^*)^T \mathbf{c}_0^*\}_{i\in[\ell]} = (\mathbf{S}_1^*)^T \mathbf{c}_0^*,$$
$$\mathbf{c}_2^* = \{\mathbf{c}_{2,i}^* = (\mathbf{S}_{2,i}^*)^T \mathbf{c}_0^*\}_{i\in[\ell]} = (\mathbf{S}_2^*)^T \mathbf{c}_0^*,$$

  where $\mathbf{S}_1^* = (\mathbf{S}_{1,1}^*|\cdots|\mathbf{S}_{1,\ell}^*)$ and $\mathbf{S}_2^* = (\mathbf{S}_{2,1}^*|\cdots|\mathbf{S}_{2,\ell}^*)$.

Analysis. We claim that the algorithm $\mathcal{A}_{lwe}$ simulates the correct running environment for the adversary $\mathcal{A}$ in games G_2 and G_3. Since the sender encryption key queries, the receiver private key queries and the policy key queries are computed according to the identical distribution, they are indistinguishable between two games. Now we analyze the distribution of the ciphertext components c_1^* and c_2^*. When plugging $\mathbf{B}_i = \mathbf{A}\mathbf{S}_{1,i}^* - ratt_i^*\mathbf{G}$ and $\mathbf{D}_i = \mathbf{A}\mathbf{S}_{2,i}^* - satt_i^*\mathbf{G}$ for $i \in [\ell]$ into the matrices $\mathbf{D}_{ratt^*} = (\mathbf{B}_1 + ratt_1^*\mathbf{G})|\cdots|\mathbf{B}_\ell + ratt_\ell^*\mathbf{G})$ and $\mathbf{D}_{satt^*} = (\mathbf{D}_1 + satt_1^*\mathbf{G})|\cdots|\mathbf{D}_\ell + satt_\ell^*\mathbf{G})$,

$$\mathbf{D}_{ratt^*} = (\mathbf{A}\mathbf{S}_{1,1}^* - ratt_1^*\mathbf{G} + ratt_1^*\mathbf{G})|\cdots|\mathbf{A}\mathbf{S}_{1,\ell}^* - ratt_\ell^*\mathbf{G} + ratt_\ell^*\mathbf{G})$$
$$= (\mathbf{A}\mathbf{S}_{1,1}^*|\cdots|\mathbf{A}\mathbf{S}_{1,\ell}^*) = \mathbf{A}\mathbf{S}_1^*,$$
$$\mathbf{D}_{satt^*} = (\mathbf{A}\mathbf{S}_{2,1}^* - satt_1^*\mathbf{G} + satt_1^*\mathbf{G})|\cdots|\mathbf{A}\mathbf{S}_{2,\ell}^* - satt_\ell^*\mathbf{G} + satt_\ell^*\mathbf{G})$$
$$= (\mathbf{A}\mathbf{S}_{2,1}^*|\cdots|\mathbf{A}\mathbf{S}_{2,\ell}^*) = \mathbf{A}\mathbf{S}_2^*,$$

where $\mathbf{S}_1^* = (\mathbf{S}_{1,1}^*|\cdots|\mathbf{S}_{1,\ell}^*)$ and $\mathbf{S}_2^* = (\mathbf{S}_{2,1}^*|\cdots|\mathbf{S}_{2,\ell}^*)$.

Then plugging $\mathbf{D}_{ratt^*} = \mathbf{A}\mathbf{S}_1^*$, $\mathbf{D}_{satt^*} = \mathbf{A}\mathbf{S}_2^*$, $\mathbf{e}_1 = (\mathbf{S}_1^*)^T\mathbf{e}_0$, $\mathbf{e}_2 = (\mathbf{S}_2^*)^T\mathbf{e}_0$ into the following formulas,

$$\mathbf{c}_1^* = (\mathbf{D}_{ratt^*})^T\mathbf{s}_0 + \mathbf{e}_1 = (\mathbf{A}\mathbf{S}_1^*)^T\mathbf{s}_0 + (\mathbf{S}_1^*)^T\mathbf{e}_0 = (\mathbf{S}_1^*)^T\mathbf{A}^T\mathbf{s}_0 + (\mathbf{S}_1^*)^T\mathbf{e}_0$$
$$= (\mathbf{S}_1^*)^T(\mathbf{A}^T\mathbf{s}_0 + \mathbf{e}_0) = (\mathbf{S}_1^*)^T\mathbf{c}_0^*,$$

$$\mathbf{c}_2^* = (\mathbf{D}_{satt^*})^T\mathbf{s}_0 + \mathbf{e}_1 = (\mathbf{A}\mathbf{S}_2^*)^T\mathbf{s}_0 + (\mathbf{S}_2^*)^T\mathbf{e}_0 = (\mathbf{S}_2^*)^T\mathbf{A}^T\mathbf{s}_0 + (\mathbf{S}_2^*)^T\mathbf{e}_0$$
$$= (\mathbf{S}_2^*)^T(\mathbf{A}^T\mathbf{s}_0 + \mathbf{e}_0) = (\mathbf{S}_2^*)^T\mathbf{c}_0^*.$$

Therefore, it is easy to see that when $\mathbf{v} = \mathbf{A}^T\mathbf{s}_0 + \mathbf{e}_0$ and $\{z_i = \mathbf{u}_i^T\mathbf{s}_0 + x_i\}_{i\in[N]}$ are the LWE tuples, the algorithm $\mathcal{A}_{lwe}$ simulates the correct running environments for $\mathcal{A}$ in game G_2, When $\mathbf{v}$ and $\{z_i\}_{i\in[N]}$ are all randomly chosen, it simulates the correct running environments for $\mathcal{A}$ in game G_3. Hence, if the adversary $\mathcal{A}$ distinguishes games G_2 and G_3, the algorithm $\mathcal{A}_{lwe}$ breaks the LWE assumption. Thus, the lemma is proved.

Lemma 10. *Games G_3 and G_4 are statistically indistinguishable, namely,*

$$\mathsf{Pr}[\mathsf{G}_3] \overset{s}{\approx} \mathsf{Pr}[\mathsf{G}_4].$$

Proof. In game G_3, the matrix $\mathbf{A}$ is chosen at random, while in game G_4, it is generated by invoking the trapdoor generation algorithm $\mathsf{GenTrap}$. Since the proof of this lemma is the same as Lemma 8, the reader can refer to the proof of Lemma 8 for the similar proof.

Lemma 11. *Games G_4 and G_5 are statistically indistinguishable, namely,*

$$\mathsf{Pr}[\mathsf{G}_4] \overset{s}{\approx} \mathsf{Pr}[\mathsf{G}_5].$$

Proof. The difference between the two games lies in that in game G_4, the ciphertext components $\mathbf{c}_1^*$ and $\mathbf{c}_2^*$ are computed via $\mathbf{c}_1^* = (\mathbf{S}_1^*)^T\mathbf{c}_0^*$, $\mathbf{c}_2^* = (\mathbf{S}_2^*)^T\mathbf{c}_0^*$, while in game G_5, they are chosen at random from the ciphertext space. The indistinguishability of the two games follows the Leftover hash lemma 4. Since the receiver private key and the policy key are generated using the left sampling algorithm under the trapdoor $\mathbf{T}_\mathbf{A}$ which do not use $\mathbf{S}_{1,i}^*$ and $\mathbf{S}_{2,i}^*$ for $i \in [\ell]$, they do not leak any information about $\mathbf{S}_{1,i}^*$ and $\mathbf{S}_{2,i}^*$. Moreover, since the only information leakage about $\mathbf{S}_{1,i}^*$ and $\mathbf{S}_{2,i}^*$ comes from the matrices $\mathbf{B}_i = \mathbf{A}\mathbf{S}_{1,i}^* - ratt_i^*\mathbf{G}$ and $\mathbf{D}_i = \mathbf{A}\mathbf{S}_{2,i}^* - satt_i^*\mathbf{G}$ in the public parameter pp, we have

$$(\mathbf{A}, \mathbf{c}_0^*, \{\mathbf{A}\mathbf{S}_{1,i}^*, (\mathbf{S}_{1,i}^*)^T\mathbf{c}_0^*\}_{i\in[\ell]}, \{\mathbf{A}\mathbf{S}_{2,i}^*, (\mathbf{S}_{2,i}^*)^T\mathbf{c}_0^*\}_{i\in[\ell]})$$
$$\overset{s}{\approx} (\mathbf{A}, \mathbf{c}_0^*, \{\mathbf{A}\mathbf{S}_{1,i}^*, \mathbf{c}_{1,i}^*\}_{i\in[\ell]}, \{\mathbf{A}\mathbf{S}_{2,i}^*, \mathbf{c}_{2,i}^*\}_{i\in[\ell]}),$$

for all randomly chosen $\mathbf{S}_{1,i}^*$, $\mathbf{S}_{2,i}^*$, $\mathbf{c}_{1,i}^*$, $\mathbf{c}_{2,i}^*$ with $i \in [\ell]$. Therefore games G_4 and G_5 are statistically indistinguishable.

Lemma 12. *Games G_5 and G_6 are statistically indistinguishable, namely,*

$$\mathsf{Pr}[\mathsf{G}_5] \overset{s}{\approx} \mathsf{Pr}[\mathsf{G}_6].$$

Proof. The proof of the Lemma is similar to that of Lemma 7.

Lemma 13. *The probability that the adversary succeeds in game G_6 is $1/2$, namely,*

$$\mathsf{Pr}[\mathsf{G}_6] = 1/2.$$

Proof. Since in game G_6, the challenge bit b is independent of the challenge ciphertext $ct^* = (\mathbf{c}_0^*, \mathbf{c}_1^*, \mathbf{c}_2^*, \{c_{3,i}^*\}_{i\in[N]})$ which are all chosen at random from the ciphertext space, the probability that the adversary succeeds in game G_6 is $1/2$.

From Lemma 7 to Lemma 13, we conclude

$$\begin{aligned}\mathsf{Adv}_{\mathcal{A},\Pi}^{sind\text{-}cpa}(\lambda) &= |\Pr[\mathsf{G}_0] - 1/2| = |\Pr[\mathsf{G}_0] - \Pr[\mathsf{G}_6]| \\ &= |\Pr[\mathsf{G}_0] - \Pr[\mathsf{G}_1] + \Pr[\mathsf{G}_1] - \Pr[\mathsf{G}_2] + \cdots + \Pr[\mathsf{G}_5] - \Pr[\mathsf{G}_6]|. \\ &\leq \sum_{i=0}^{5} |\Pr[\mathsf{G}_i] - \Pr[\mathsf{G}_{i+1}]| \leq \mathsf{Adv}_{\mathcal{A}_{lew}}^{lwe}(\lambda) + negl(\lambda).\end{aligned}$$

Hence Theorem 3 is proved.

Theorem 4. *The key-policy bilateral access control ABE scheme Π given in Section 5.1 achieves the authentication security under the selective UF-CMA security of the IBS scheme implicitly applied to our scheme.*

Proof. Assume that there exists a ppt adversary $\mathcal{A}$ that can break the authentication of the scheme, then we can construct a ppt algorithm $\mathcal{A}_{sig}$ that can break the selective UF-CMA security of the IBS signature scheme.

Setup. Assume that at the beginning of this phase, the algorithm $\mathcal{A}_{sig}$ has received a challenge matrix $\mathbf{A}'$ from its challenger. The algorithm first sets the public parameter component $\mathbf{A}'$ as the challenge matrix, then generates the other public parameter components $\mathbf{A}, \mathbf{u}_1, \cdots, \mathbf{u}_N, \mathbf{B}_1, \cdots, \mathbf{B}_\ell, \mathbf{D}_1, \cdots, \mathbf{D}_\ell, H_1, H_2$ as in the initial scheme. Note that the algorithm $\mathcal{A}_{sig}$ knows the master secret key component $\mathbf{T_A}$.

Query. When the adversary $\mathcal{A}$ makes the sender encryption key query with a sender attribute $sattr$, the reduction algorithm $\mathcal{A}_{sig}$ queries its challenger and gets a sender encryption key ek_{satt} (corresponding to the signature key query of the IBS scheme);

When the adversary $\mathcal{A}$ makes the receiver private key query with a receiver policy f_r, the reduction algorithm $\mathcal{A}_{sig}$ first picks N random vectors $\mathbf{u}_{1,i} \leftarrow_\$ \mathbb{Z}_q^n$ for $i \in [N]$ and computes $\mathbf{B}_{f_r} = \mathsf{Eval}_{pp}(f_r, (\mathbf{B}_1, \cdots, \mathbf{B}_\ell))$. Then it runs the algorithm $\mathsf{SampleLeft}$ $(\mathbf{A}, \mathbf{B}_{f_r}, \mathbf{T_A}, \mathbf{u}_{1,i}, \delta)$ to generate $\mathbf{r}_{f_r}^i$ s.t. $(\mathbf{A}|\mathbf{B}_{f_r}) \cdot \mathbf{r}_{f_r}^i = \mathbf{u}_{1,i}$, and sends $rk_{f_r} = \{\mathbf{r}_{f_r}^i\}_{i\in[N]}$ to $\mathcal{A}$.

When $\mathcal{A}$ makes the policy key query with a policy f_s, the reduction algorithm $\mathcal{A}_{sig}$ first computes $\mathbf{D}_{f_s} = \mathsf{Eval}_{pp}(f_s, (\mathbf{D}_1, \cdots, \mathbf{D}_\ell))$ and $\mathbf{u}_{2,i} = \mathbf{u}_i - \mathbf{u}_{1,i}$. Then it runs the left sampling algorithm $\mathsf{SampleLeft}$ $(\mathbf{A}, \mathbf{D}_{f_s}, \mathbf{T_A}, \mathbf{u}_{2,i}, \delta)$ to sample $\mathbf{r}_{f_s}^i$ s.t. $(\mathbf{A}|\mathbf{D}_{f_s}) \cdot \mathbf{r}_{f_s}^i = \mathbf{u}_{2,i}$, and sends $pok_{f_s} = \{\mathbf{r}_{f_s}^i\}_{i\in[N]}$ to $\mathcal{A}$.

Forge. At this phase, $\mathcal{A}$ outputs a forgery $(ct^* = (\mathbf{c}_0^*,\ \mathbf{c}_1^*,\ \mathbf{c}_2^*,\ \{c_{3,i}^*\}_{i\in[N]},$ $ratt^*, satt^*),\ f_r^*,\ f_s^*)$.

The reduction algorithm $\mathcal{A}_{sig}$ first computes the receiver private key $rk_{f_r^*} = \{\mathbf{r}_{f_r^*}^i\}_{i\in[N]}$ for the policy f_r^* as follows. It picks N random vectors $\mathbf{u}_{1,i}^* \leftarrow_\$ \mathbb{Z}_q^n$ for

$i \in [N]$, computes $\mathbf{B}_{f_r^*} = \mathsf{Eval}_{pp}(f_r^*, \mathbf{B}_1, \cdots, \mathbf{B}_\ell)$, and then runs the algorithm $\mathsf{SampleLeft}$ $(\mathbf{A}, \mathbf{B}_{f_r^*}, \mathbf{T_A}, \mathbf{u}_{1,i}^*, \delta)$ to generate $\mathbf{r}_{f_r^*}^i$ s.t. $(\mathbf{A}|\mathbf{B}_{f_r^*}) \cdot \mathbf{r}_{f_r^*}^i = \mathbf{u}_{1,i}^*$.

Then it computes the policy key $pok_{f_s^*} = \{\mathbf{r}_{f_s^*}^i\}_{i\in[N]}$ for the policy f_s^* as follows. It first computes $\mathbf{D}_{f_s^*} = \mathsf{Eval}_{pp}(f_s^*, \mathbf{D}_1, \cdots, \mathbf{D}_\ell)$, $\mathbf{u}_{2,i}^* = \mathbf{u}_i - \mathbf{u}_{1,i}^*$, then runs the left sampling algorithm $\mathsf{SampleLeft}$ $(\mathbf{A}, \mathbf{D}_{f_s^*}, \mathbf{T_A}, \mathbf{u}_{2,i}^*, \delta)$ to sample $\mathbf{r}_{f_s^*}^i$ s.t. $(\mathbf{A}|\mathbf{D}_{f_s^*}) \cdot \mathbf{r}_{f_s^*}^i = \mathbf{u}_{2,i}^*$.

Next the reduction algorithm $\mathcal{A}_{sig}$ parses the ciphertext component $\mathbf{c}_1^*$, $\mathbf{c}_2^*$ into $\mathbf{c}_1^* = \{\mathbf{c}_{1,i}^*\}_{i\in[\ell]}$, $\mathbf{c}_2^* = \{\mathbf{c}_{2,i}^*\}_{i\in[\ell]}$ respectively, and computes

$$\mathbf{c}_{f_r^*} = \mathsf{Eval}_{ct}(f_r^*, \{(\mathbf{B}_i, ratt_i^*\mathbf{G}, \mathbf{c}_{1,i}^*)\}_{i\in[\ell]}),\ \ \mathbf{c}_{f_s^*} = \mathsf{Eval}_{ct}(f_s^*, \{(\mathbf{D}_i, satt_i^*\mathbf{G}, \mathbf{c}_{2,i}^*)\}_{i\in[\ell]}),$$

and

$$w_i^* = c_{3,i}^* - (\mathbf{r}_{f_r^*}^i)^T \begin{bmatrix} \mathbf{c}_0 \\ \mathbf{c}_{f_r^*} \end{bmatrix} - (\mathbf{r}_{f_s^*}^i)^T \begin{bmatrix} \mathbf{c}_0 \\ \mathbf{c}_{f_s^*} \end{bmatrix}.$$

If $|w_i^* - \lceil \frac{q}{2} \rceil| < \frac{q}{4}$, it sets $m_i' = 1$, and otherwise $m_i' = 0$. Then it sets $\mu' = m_0' \cdots m_{\ell_0}'$ and $\sigma' = m_{\ell_0+1}' \cdots m_N'$. Next, it sets $\boldsymbol{\sigma} = [\sigma']_v$, and computes $\mathbf{h}_{\mu'} = H_2(pp, satt^*, \mu'|ratt^*)$, $\mathbf{D}_{satt^*} = (\mathbf{D}_1 + satt_1^*\mathbf{G}| \cdots |\mathbf{D}_\ell + satt_\ell^*\mathbf{G})$, $\mathbf{H}_{satt^*} = H_1(pp, \mathbf{D}_{satt^*})$, $\mathbf{F}_{satt^*} = [\mathbf{A}'|\mathbf{H}_{satt^*}]$, and verifies whether $\mathbf{F}_{satt^*} \cdot \boldsymbol{\sigma} \overset{?}{=} \mathbf{h}_{\mu'}$ holds. If yes, it returns $(satt^*, \mu', \sigma)$ as the signature forgery.

From the construction, we can see that the sender encryption key ek_{satt^*} for the sender attribute $satt^*$ embedded in the forged ciphertext ct^* is not queried by the adversary $\mathcal{A}$. This is because, the signature key query for attribute $satt^*$ is not allowed in the selective UF-CMA security experiment of the IBS, which is identical to the restriction on the sender encryption key query for $satt^*$ in the authentication experiment of the bilateral access control ABE scheme. Therefore, if the adversary $\mathcal{A}$ can break the authentication of the bilateral access control ABE scheme Π, the reduction algorithm $\mathcal{A}_{sig}$ can output a signature forgery, which contradicts with the assumption that the IBS scheme is selective UF-CMA secure, which proves the Theorem.

Remark 2. Note that when the attributes and policies of both parties do not match (i.e., $f_s(satt^*) \neq 0 \wedge f_r(ratt^*) \neq 0$), the provable security of our scheme implies the security in the case where one party's attributes and policies match while the other's do not (i.e., $f_s(satt^*) = 0 \wedge f_r(ratt^*) \neq 0$ or $f_s(satt^*) \neq 0 \wedge f_r(ratt^*) = 0$). This is because, aside from the decryption correctness requirement that both parties' attributes must satisfy their respective policies to correctly recover the encrypted message, the actual security proof reduction does not need the conditions that the attributes of both parties do not meet their respective policies (i.e., $f_s(satt^*) \neq 0 \wedge f_r(ratt^*) \neq 0$).

6 Implementing Attribute-Hiding to Obtain AB-ME

In this section, using the similar transformation [32] from any attribute-based encryption (ABE) without attribute-hiding into another ABE with attribute-hiding where the attributes of the sender and receiver are both hidden from any user except the receiver that is authorized to decrypt, we obtain an attribute-hiding bilateral access control ABE from the basic scheme in subsection 5.1. We call such scheme as weak attribute-based matchmaking encryption (weak AB-ME) where the term "weak" means that the attributes of the sender and receiver are hidden only when there is mismatch (for convenience, we abbreviate weak AB-ME as AB-ME).

6.1 Construction

Using the transformation from ABE to ABE with attribute-hiding in [32], we get an attribute-hiding bilateral access control ABE scheme (i.e., AB-ME) from the bilateral access control ABE scheme (in section 5.1). More concretely, we assume that the AB-ME scheme is $\mathsf{ABME} = (\mathsf{Setup}, \mathsf{SKGen}, \mathsf{RKGen}, \mathsf{PoKGen}, \mathsf{Enc}, \mathsf{Dec})$, the underlying bilateral access control ABE scheme is $\mathsf{BACABE} = (\mathsf{Setup}, \mathsf{SKGen}, \mathsf{RKGen}, \mathsf{PoKGen}, \mathsf{Enc}, \mathsf{Dec})$, the lockable obfuscation scheme is $\mathsf{LOBF} = (\mathsf{Obf}, \mathsf{Eval})$, then the transformation from BACABE to ABME is as follows.

- The setup algorithm $\mathsf{ABME.Setup}(1^\lambda, \ell, N)$ is the same as the setup algorithm of the bilateral access control ABE scheme, which outputs a public parameter pp and a master secret key msk.
- The sender key generation algorithm $\mathsf{ABME.SKGen}(pp, msk, satt \in \{0,1\}^\ell)$ is the same as the sender key generation algorithm of the bilateral access control ABE scheme, which outputs a sender encryption key ek_{satt} for the sender attribute $satt$.
- The receiver key generation algorithm $\mathsf{ABME.RKGen}(pp, msk, f_r \in \mathcal{F}_r)$ is the same as the receiver key generation algorithm of the bilateral access control ABE scheme, which outputs a receiver key rk_{f_r} for the receiver policy f_r.
- The policy key generation algorithm $\mathsf{ABME.PoKGen}(pp, msk, f_s \in \mathcal{F}_s)$ is the same as the policy key generation algorithm of the bilateral access control ABE scheme, which outputs a policy key pok_{f_s} for the policy f_s.
- The encryption algorithm $\mathsf{ABME.Enc}(pp, ek_{satt}, \mu \in \mathcal{M}, ratt)$ first picks a random lock string $\alpha \leftarrow_\$ \{0,1\}^s$. Then according to the encryption algorithm $\mathsf{BACABE.Enc}$, it computes a signature σ on the message $\mu|ratt$ under the attribute $satt$, and a ciphertext ct_α on the message α. Next it constructs a decryption circuit

  $$P := \mathsf{BACABE.Dec}(pp, \cdot, \cdot, (ct_\alpha, ratt, satt)),$$

  and uses the circuit to compute an obfuscated program

  $$\widetilde{P} \leftarrow \mathsf{Obf}(1^\lambda, P, \mu||\sigma||ratt||satt||ct_\alpha, \alpha).$$

  Finally the ciphertext is set as $ct = \widetilde{P}$.
- In the decryption algorithm $\mathsf{ABME.Dec}(rk_{f_r}, pok_{f_s}, ct)$: let $ct = \widetilde{P}$, the algorithm evaluates the obfuscated program $\mathsf{Eval}(\widetilde{P}, rk_{f_r}, pok_{f_s})$ on inputs rk_{f_r}, pok_{f_s} to output $\mu||\sigma||ratt||satt||ct_\alpha$. When getting $\mu||\sigma||ratt||satt||ct_\alpha$, it executes the signature verification for the message/signature pair $(\mu||ratt, \sigma)$

under the sender attribute *satt* according to the verification steps in the decryption algorithm of the scheme BACABE. If the signature verification holds, it outputs μ.

Correctness. For all $\lambda \in \mathbb{N}$, message $\mu \in \mathcal{M}$, attributes $ratt, satt \in \mathcal{S}$ and policies $f_r \in \mathcal{F}_r$, $f_s \in \mathcal{F}_s$, and public parameter and master secret key $(pp, msk) \leftarrow \mathsf{ABME.Setup}(1^\lambda, \ell, N)$,

(1) if the ciphertext corresponding to a message μ and attributes $ratt, satt$ is an obfuscated program

$$\widetilde{P} \leftarrow \mathsf{Obf}(1^\lambda, P, \mu||\sigma||ratt||satt||ct_\alpha, \alpha),$$

where

$$P := \mathsf{BACABE.Dec}(pp, \cdot, \cdot, (ct_\alpha, ratt, satt)),$$

$$ct_\alpha = \mathsf{BACABE.Enc}(pp, ek_{satt}, \alpha, satt), \quad \alpha \leftarrow_\$ \{0,1\}^s$$

and σ is generated according to the steps of the encryption algorithm

$$\mathsf{BACABE.Enc}(pp, ek_{satt}, \mu, satt);$$

(2) if the corresponding receiver private key rk_{f_r} and policy key pok_{f_s} are computed as

$$rk_{f_r} \leftarrow \mathsf{BACABE.RKGen}(pp, msk, f_r),$$

$$pok_{f_s} \leftarrow \mathsf{BACABE.PoKGen}(pp, msk, f_s).$$

Then we have:

– If $f_r(ratt) = 0$ and $f_s(satt) = 0$, then with all but negligible probability $\mathsf{BACABE.Dec}(pp, rk_{f_r}, pok_{f_s}, (ct_\alpha, ratt, satt)) = \alpha$, this follows the correctness of the bilateral access control ABE scheme. Then we can conclude $\mathsf{Eval}(\widetilde{P}, rk_{f_r}, pok_{f_s}) = \mu||\sigma||ratt||satt||ct_\alpha$ when

$$\mathsf{BACABE.Dec}(pp, rk_{f_r}, pok_{f_s}, (ct_\alpha, ratt, satt)) = \alpha.$$

Therefore, when $f_r(ratt) = 0$ and $f_s(satt) = 0$, then

$$\Pr[\mathsf{Eval}(ct = \widetilde{P}, rk_{f_r}, pok_{f_s}) = \mu||\sigma||ratt||satt||ct_\alpha] \geq 1 - negl(\lambda).$$

Furthermore, if the message/signature pair $(\mu||ratt, \sigma)$ is verified, then the message μ is recovered with overwhelming probability, namely

$$\Pr[\mathsf{ABME.Dec}(rk_{f_r}, pok_{f_s}, ct = \widetilde{P}) = \mu] \geq 1 - negl(\lambda).$$

– If $f_r(ratt) = 1$ or $f_s(satt) = 1$, then with all but negligible probability

$$\mathsf{BACABE.Dec}(pp, rk_{f_r}, pok_{f_s}, (ct_\alpha, ratt, satt)) = \perp.$$

By the correctness of the lockable obfuscation scheme, we have

$$\Pr[\mathsf{Eval}(\widetilde{P}, rk_{f_r}, pok_{f_s}) = \perp] \geq 1 - negl'(\lambda),$$

if
$$\mathsf{BACABE.Dec}(pp, rk_{f_r}, pok_{f_s}, (ct_\alpha, ratt, satt)) \neq \alpha.$$

Furthermore, by $\perp \neq \mu||\sigma||ratt||satt||ct_\alpha$, if $f_r(ratt) = 1$ or $f_s(satt) = 1$, then
$$\Pr[\mathsf{ABME.Dec}(rk_{f_r}, pok_{f_s}, ct = \widetilde{P}) = \perp] \geq 1 - negl'(\lambda),$$

which prove the correctness of the proposed scheme ABME.

Remark 3. Note that in our AB-ME scheme, the encryption algorithm ensures that there is no inconsistency between the attribute $satt$ embedded in the ciphertext ct_α during encryption and the attribute $satt$ included in the payload $\mu||\sigma||ratt||satt||ct_\alpha$ obfuscated within the program $\widetilde{P}$. This is because: suppose the encryptor maliciously uses an unauthorized attribute $satt'$ to encrypt α, while embedding a different $satt$ in the obfuscated program $\widetilde{P}$. During the decryption, by the correctness of the BACABE scheme, if both parties' attributes satisfy their respective policies, α will be correctly recovered. Again, by the correctness of the lockable obfuscator, evaluating $\Pr[\mathsf{Eval}(\widetilde{P}, rk_{f_r}, pok_{f_s})$ will output $\mu||\sigma||ratt||satt||ct_\alpha$. The decryption algorithm then uses the attribute $satt$ to verify whether the signature/message pair $(\mu||ratt, \sigma)$ included in the payload is satisfied (note that the signature σ is generated under the attribute $satt'$ during encrypting α by the BACABE encryption algorithm). If $satt' \neq satt$, the signature verification will fail due to the unforgeability of the identity-based signature scheme and the decryption correctness of the BACABE scheme.

6.2 Security Proof

In this section, we prove the security of the proposed AB-ME scheme.

Theorem 5. *If the bilateral access control ABE scheme* BACABE *proposed in section 5.1 satisfies the sIND-CPA security and the authentication security defined in section B.2, and the lockable obfuscation scheme* LOBF *is secure (see section 2.4), then the proposed AB-ME scheme* ABME *in section 6.1 achieves both privacy and authentication (see section 4).*

The proof of Theorem 5 can be obtained via Theorem 6 and Theorem 7.

Theorem 6. *If the bilateral access control ABE scheme* BACABE *proposed in section 5.1 satisfies the sIND-CPA security and the lockable obfuscation scheme* LOBF *is secure, then the AB-ME scheme* ABME *proposed in section 6.1 achieves privacy.*

See Appendix C for the proof of this Theorem.

Theorem 7. *If the bilateral access control ABE scheme proposed in section 5.1 satisfies authentication, then the AB-ME scheme proposed in section 6.1 achieves authentication.*

See Appendix D for the proof of this Theorem.

Another Method to Capture AB-ME via Minor Modification to the Above Scheme. Observe that in the AB-ME scheme proposed in subsection 6.1, the underlying bilateral access control ABE scheme with instantiation under lattice assumptions has pseudorandom cipertext property. Using this property, we can modify the ciphertext $ct = \widetilde{P}$ in the AB-ME scheme into $ct = (\widetilde{P}, ct_\alpha)$, which results in another construction of AB-ME based on the same assumptions. More specifically, we pull the intermediate generated ciphertext ct_α away from the encrypted message $\mu||\sigma||ratt||satt||ct_\alpha$ in the lockable obfuscation algorithm Obf and put it with $\widetilde{P}$ together to consist of the resulted ciphertext $ct = (\widetilde{P}, ct_\alpha)$. Namely, the critical encryption steps are modified as below (the rest remain unchanged):

$$\widetilde{P} \leftarrow \mathsf{Obf}(1^\lambda, P, \mu||\sigma||ratt||satt, \alpha),$$

where

$$ct_\alpha \leftarrow \mathsf{BACABE.Enc}(pp, ek_{satt}, \alpha, ratt),$$
$$P := \mathsf{BACABE.Dec}(pp, \cdot, \cdot, (ct_\alpha, ratt, satt)).$$

The decryption algorithm should be also made the corresponding changes. Specifically, let $ct = (\widetilde{P}, ct_\alpha)$, the decryption algorithm evaluates the obfuscated program $\mathsf{Eval}(\widetilde{P}, rk_{f_r}, pok_{f_s})$ to recover $\mu||\sigma||ratt||satt$, and invokes the signature verification to check whether the message/signature pair $(\mu||ratt, \sigma)$ under the sender attribute $satt$ is satisfied. If the check holds, it outputs μ. For the security, the privacy and authentication proofs are similar to Theorem 6 and Theorem 7.

Note that although the ciphertext of the above construction is larger than the ciphertext in subsection 6.1, here we only stress that there also exists another construction for AB-ME under the same lattice assumptions. Furthermore, as indicated in [32], using universal circuits (or formulas) [12] one can build a ciphertext-policy AB-ME schemes from key-policy AB-ME scheme. Note that this only works for circuits of aprori bounded size.

References

1. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, Alexandria, VA, USA, Ioctober 30-November 3, 2006. ACM.
2. Sahai, A., Waters, B.: Fuzzy Identity-based encryption. In: Cramer, R. (eds) Advances in Cryptology-EUROCRYPT 2005. Lecture Notes in Computer Science, vol 3494. Springer, Berlin, Heidelberg.
3. Jiang, C. S., Xu, C. X., Han, Y. X., Zhang, Z., Chen, K. F.: Two-factor authenticated key exchange from biometrics with low entropy rates. IEEE Transactions on Information Forensics and Security, Volume 19, pp. 3844-3856.
4. Li, T., Li, X. Q., Liu, X. F.: An efficient privacy-preserving bidirectional friends matching scheme in mobile social networks. In Proc. 2019 International Conference on Networking and Network Applications (NaNA), DOI:10.1109/NaNA.2019.00018, 2019.

5. Xu, S.M., Ning, J. T., Huang, X.Y., Zhou, J. Y., Deng, R.H.: Server-aided bilateral access control for secure data sharing with dynamic user groups. IEEE Transactions on Information Forencics and Security, 16: 4746-4761, 2021.
6. Xu, S.M., Ning, J. T., Li, Y.J., Zhang, Y.H., Xu, G.W., Huang, X.Y., Deng, R.H.: Match in my way: Fine-grained bilateral access control for secure cloud-fog computing. IEEE Transactions on Dependable and Secure Computing, 19(2): pp. 1064-1077, 2022.
7. Xu, S.M., Ning, J. T., Ma, J., Huang, X. Y., Pang, H., Deng, R.H.: Expressive bilateral access control for internet-of-things in cloud-fog computing. In Proc. SACMAT 2021, pp. 143-154.
8. Boneh, D., Gentry, C., Gorbunov, S., Halevi, S., et al.: Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In: Nguyen, P.Q., Oswald, E. (eds) Advances in Cryptology-EUROCRYPT 2014. Lecture Notes in Computer Science, vol 8441. Springer, Berlin, Heidelberg.
9. Shamir, A.: Identity-Based Cryptosystems and Signature Schemes. In: Blakley, G.R., Chaum, D. (eds) Advances in Cryptology-CRYPTO 1984. Lecture Notes in Computer Science, vol 196. Springer, Berlin, Heidelberg.
10. Luo, F. C., Al-Kuwari, S., Wang, H. Y., Wang, F. Q., Chen, K. F.: Revocable attribute-based encryption from standard lattices. Computer Standards and Interfaces, Vol. 84, No. C. 2023. https://doi.org/10.1016/j.csi.2022.103698.
11. Brakerski, Z., Vaikuntanathan, V.: Circuit-ABE from LWE: Unbounded attributes and semi-adaptive security. In: Robshaw, M., Katz, J. (eds) Advances in Cryptology-CRYPTO 2016. Lecture Notes in Computer Science, vol 9816. Springer, Berlin, Heidelberg.
12. Goyal, Vipul., Jain, A., Pandey, O., Sahai, A.: Bounded ciphertext policy attribute based encryption. In: Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part II-Track B: Logic, Semantics, and Theory of Programming & Track C: Security and Cryptography Foundations, 2008.
13. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Attribute-based encryption for circuits. Journal of the ACM (JACM) 62 (2013), pp. 1-33.
14. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Predicate encryption for circuits from LWE. In Advances in Cryptology-CRYPTO 2015, LNCS, vol 9216. Springer, Berlin, Heidelberg.
15. Ateniese, G., Francati, D., Nunez, D., Venturi, D.: Match me if you can: Matchmaking encryption and its applications. In Advances in Cryptology-CRYPTO 2019, pp. 701-731, 2019.
16. Ateniese, G., Francati, D., Nunez, D., Venturi, D.: Match me if you can: Matchmaking encryption and its applications. J Cryptol 34, 16 (2021).
17. Francati, D., Guidi, A., Russo, L., Venturi, D.: Identity-based matchmaking encryption without random oracles. In: Adhikari, A., KÍźsters, R., Preneel, B. (eds) Progress in Cryptology-INDOCRYPT 2021. Lecture Notes in Computer Science, vol 13143. Springer, Cham.
18. Chen, J., Li, Y., Wen, J.M., Weng, J.: Identity-based matchmaking encryption from standard assumptions, In Advances in Cryptology-ASIACRYPT 2022: 28th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, December 5-9, 2022, Proceedings, Part III (pp. 394-422). Cham: Springer Nature Switzerland.
19. Wu, A. X., Luo, W. Q., Weng, J., Yang, A. J., Wen, J. H.: Fuzzy identity-based matchmaking encryption and its application. IEEE Transactions on Information Forencics and Security, pp. 5592-5607, 2024.

20. Wang, Y.J., Wang, B.C., Wang, Q.Q., Zhan, Y.: Identity-based matchmaking encryption with stronger security and instantiation on lattices. https://eprint.iacr.org/2022/1718.pdf, 2022.
21. Wang H. G., Chen K. F., Xie Q., Meng Q.. Post-quantum secure identity-based matchmaking encryption. IEEE Transactions on Dependable and Secure Computing, vol. 22, no. 1, pp. 833-844, Jan.-Feb. 2025.
22. Jiang, Z., Wang, X. W., Zhang, K., Gong, J. Q., Chen, J., Qian, H. J.: Revocable identity-based matchmaking encryption in the standard model. IFIP International Conference on ICT Systems Security and Privacy Protection, DOI: 10.1049/ise2.12116. 2023.
23. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008.
24. Agrawal, S., Boneh, D., Boyen, X.: Efficient lattice (H)IBE in the standard model. In: Gilbert, H. (eds) Advances in Cryptology-EUROCRYPT 2010. Lecture Notes in Computer Science, vol 6110. Springer, Berlin, Heidelberg.
25. Dodis, Y., Ostrovsky, R., Reyzin, L., Smith, A.: Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. SIAM J. Comput., 38(1), pp. 97-139, 2008.
26. Micciancio, D., Peikert, C.: Trapdoors for lattices: Simpler, tighter, faster, smaller. In: Pointcheval, D., Johansson, T. (eds) Advances in Cryptology-EUROCRYPT 2012. Lecture Notes in Computer Science, vol 7237. Springer, Berlin, Heidelberg.
27. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. Journal of the ACM, 56(6), Article 1568324.
28. Ajtai M.: Generating hard instances of lattice problems (extended IBStract). In Proceedings of the 28th Annual ACM Symposium on Theory of Computing 1996, pp. 99-108, 1996.
29. Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In: Canetti, R., Garay, J.A. (eds) Advances in Cryptology-CRYPTO 2013. Lecture Notes in Computer Science, vol 8042. Springer, Berlin, Heidelberg.
30. Francati, D., Friolo, D., Malavolta, G., Venturi, D.: Multi-key and Multi-input Predicate Encryption from Learning with Errors. In: Hazay, C., Stam, M. (eds) Advances in Cryptology-EUROCRYPT 2023. Lecture Notes in Computer Science, vol 14006. Springer, Cham.
31. Francati, D., Friolo, D., Malavolta, G. et al. Multi-key and Multi-input Predicate Encryption (for Conjunctions) from Learning with Errors. J Cryptol 37, 24 (2024).
32. Goyal, R., Koppula, V., Waters, B.: Lockable obfuscation. In: Umans, C. (ed.) 58th FOCS, pp. 612-621. IEEE Computer Society Press 2017.
33. Wichs, D., Zirdelis, G.: Obfuscating compute-and-compare programs under LWE. In Chris Umans, editor, 58th FOCS, pp. 600-611. IEEE Computer Society Press, October 2017.

A Construction for Identity-Based Signature in [20]

In this section, we review the SIS-based instantiation of the identity-based signature (IBS) scheme from [20].

$\mathsf{Setup}(1^\lambda) \to (mpk, msk)$: The setup algorithm $\mathsf{Setup}(1^\lambda)$ takes as input a security 1^λ. It first chooses two random hash functions $H_1 : \{0,1\}^* \to \mathbb{Z}^{n\times m}$,

$H_2 : \{0,1\}^* \to \mathbb{Z}^n$ and computes $(\mathbf{A}, \mathbf{T_A}) \leftarrow \mathsf{GenTrap}(1^n, m, q)$. Then it sets the master public key and the master secret key as $mpk := (\mathbf{A}, H_1, H_2)$, $msk := \mathbf{T_A}$, and outputs (mpk, msk).

$\mathsf{KeyExt}(msk, id) \to sk_{id}$: The key generation algorithm $\mathsf{KeyExt}(msk, id)$ takes as input the master secret key msk and a user's identity id. It computes $\mathbf{H}_{id} \leftarrow H_1(mpk, id)$, $\mathbf{F}_{id} \leftarrow [\mathbf{A}|\mathbf{H}_{id}]$, $\mathbf{T}_{id} \leftarrow \mathsf{DelTrap}(\mathbf{F}_{id}, \mathbf{T_A}, \delta)$. Then it sets the private key as $sk_{id} := \mathbf{T}_{id}$, and outputs sk_{id}.

$\mathsf{Sign}(sk_{id}, m) \to \boldsymbol{\sigma}$: The signature algorithm $\mathsf{Sign}(sk_{id}, m)$ takes as input a private key sk_{id} and a message m. It first parses sk_{id} into $\mathbf{T}_{id}$, and computes $\mathbf{h}_m \leftarrow H_2(mpk, id, m)$, $\mathbf{H}_{id} \leftarrow H_1(mpk, id)$, $\mathbf{F}_{id} \leftarrow [\mathbf{A}|\mathbf{H}_{id}]$ and $\mathbf{z} \leftarrow \mathsf{PreSamp}(\mathbf{F}_{id}, \mathbf{T}_{id}, \mathbf{h}_m, \delta)$ such that $\mathbf{F}_{id} \cdot \mathbf{z} = \mathbf{h}_m$. Then it sets the signature as $\boldsymbol{\sigma} := \mathbf{z}$ and returns $\boldsymbol{\sigma}$.

$\mathsf{Ver}(mpk, id, m, \boldsymbol{\sigma}) \to 0/1$: The verification algorithm $\mathsf{Ver}(mpk, id, m, \boldsymbol{\sigma})$ takes as the master public key mpk, an identity id, a message m and a signature $\boldsymbol{\sigma}$. It first computes $\mathbf{H}_{id} \leftarrow H_1(mpk, id)$, $\mathbf{F}_{id} \leftarrow [\mathbf{A}|\mathbf{H}_{id}]$, $\mathbf{h}_m \leftarrow H_2(mpk, id, m)$, and then checks whether the condition $\boldsymbol{\sigma} = 0 \vee \mathbf{F}_{id} \cdot \boldsymbol{\sigma} \neq \mathbf{h}_m$ holds. If yes, it outputs 0, else it outputs 1.

Theorem 8. *The identity-based signature scheme described above achieves selective UF-CMA security (refer to section 2.5) under the SIS assumption if the hash functions H_1, H_2 are modeled as random oracles.*

B Syntax and Security definition of Key-Policy AB-ME

In this section, we give the syntax and security definition of key-policy AB-ME (short for "AB-ME").

B.1 Syntax

A key-policy AB-ME scheme consists of the following six ppt algorithms.

$\mathsf{Setup}(1^\lambda, \ell, N) \to (pp, msk)$. The setup algorithm $\mathsf{Setup}(1^\lambda, \ell, N)$ is executed by the Key Generation Center (KGC) which takes as input the security parameter 1^λ, the maximum length of the attributes ℓ and the maximum length of encrypted message $N \in \mathbb{N}$. It outputs a public parameter pp and a master secret key msk.

$\mathsf{SKGen}(pp, msk, satt) \to ek_{satt}$: The sender encryption key generation algorithm SKGen $(pp, msk, satt)$ is run by the KGC which takes as input the public parameter pp, the master secret key msk and a sender's attribute $satt$, and outputs a sender encryption key ek_{satt}.

$\mathsf{RKGen}(pp, msk, f_r) \to rk_{f_r}$: The receiver key generation algorithm $\mathsf{RKGen}(pp, msk, f_r)$ is executed by the KGC which takes as input the public parameter pp, the master secret key msk and a receiver policy f_r, and outputs the receiver private key rk_{f_r}.

$\mathsf{PoKGen}(pp, msk, f_s) \rightarrow pok_{f_s}$: The policy key generation algorithm $\mathsf{PoKGen}(pp, msk, f_s)$ is run the KGC which takes as input the public parameter pp, the master secret key msk and a specified sender policy f_s, and outputs a policy key pok_{f_s}.

$\mathsf{Enc}(pp, ek_{satt}, \mu, ratt) \rightarrow ct$: The encryption algorithm $\mathsf{Enc}(pp, ek_{satt}, \mu, ratt)$ is executed by a sender which takes as input the public parameter pp, the sender's encryption key ek_{satt}, a message μ and a specified receiver's attribute $ratt$, and outputs a ciphertext ct.

$\mathsf{Dec}(pp, rk_{f_r}, pok_{f_s}, ct) \rightarrow \mu/\perp$: The decryption algorithm $\mathsf{Dec}(pp, rk_{f_r}, pok_{f_s}, (ct_{ratt,satt}, ratt, satt))$ takes as input the public parameter pp, a receiver private key rk_{f_r}, a policy key pok_{f_s} and a ciphertext $ct_{ratt,satt}$, and outputs a message $(\mu, satt, ratt)$ or $\perp$.

Correctness. For all $(pp, msk) \leftarrow \mathsf{Setup}(1^\lambda, \ell, N)$, $ek_{satt} \leftarrow \mathsf{SKGen}\ (pp, msk, satt)$, $rk_{f_r} \leftarrow \mathsf{RKGen}(pp, msk, f_r)$, $pok_{f_s} \leftarrow \mathsf{PoKGen}(pp, msk, f_s)$, we have

(1) If $f_r(ratt) = 0 \wedge f_s(satt) = 0$, then

$$\Pr[\mathsf{Dec}(pp, rk_{f_r}, pok_{f_s}, \mathsf{Enc}(pp, ek_{satt}, \mu, ratt)) = (\mu, satt, ratt)] = 1 - negl(\lambda).$$

(2) Otherwise,

$$\Pr[\mathsf{Dec}(pp, rk_{f_r}, pok_{f_s}, \mathsf{Enc}(pp, ek_{satt}, \mu, ratt)) = \perp] = 1 - negl(\lambda).$$

B.2 Security Model

In this section, we give the privacy and authentication security definition for AB-ME. The privacy security captures the confidentiality of the message μ, the sender's attribute and the sender's attribute encapsulated in a ciphertext when a mismatch does not happen. The authentication security guarantees that a valid ciphertext can not be forged under a sender encryption key that is not authorized by the KGC.

Let Π be a key-policy AB-ME scheme, $\mathcal{A}$ a ppt adversary, and $\mathcal{C}$ a challenger, the privacy security experiment $\mathsf{PRIV}_{\mathcal{A}}^{\Pi}(\lambda)$ played between $\mathcal{A}$ and $\mathcal{C}$ is described as follows.

The privacy experiment $\mathsf{PRIV}_{\mathcal{A}}^{\Pi}(\lambda)$.
The privacy security experiment consists of the following several phases.

Setup. The adversary first claims a challenge pair $((ratt_0, satt_0), (ratt_1, satt_1)$. Then the challenger runs the setup algorithm $\mathsf{Setup}(1^\lambda, N)$ to generate the public parameter pp and the master secret key msk, and sends pp to the adversary.

Query 1. In this phase, the adversary can make a series of sender encryption key queries, receiver private key queries and policy-key queries.

– When the adversary $\mathcal{A}$ submits a sender attribute $satt$, the challenger $\mathcal{C}$ invokes the sender encryption key generation algorithm $\mathsf{SKGen}(pp, msk, satt)$ to generate a sender encryption key ek_{satt} and delivers it to $\mathcal{A}$.

- When the adversary $\mathcal{A}$ issues a receiver private key query for a policy f_r such that $f_r(ratt^*) \neq 0$, the challenger $\mathcal{C}$ runs the receiver private key generation algorithm $\mathsf{RKGen}(pp, msk, f_r)$ to generate a receiver private key rk_{f_r} and sends it to the adversary.
- When the adversary $\mathcal{A}$ commits a policy key query for a specified sender policy f_s such that $f_s(satt^*) \neq 0$, the challenger $\mathcal{C}$ executes the policy-key generation algorithm $\mathsf{PoKGen}(pp, msk, f_s)$ to generate a policy-key pok_{f_s}, and sends it to the adversary.

Challenge. In this phase, when the adversary submits a challenge message pair (μ_0^*, μ_1^*), the challenger first flips a random coin $b \leftarrow_{\$} \{0,1\}$ and invokes the sender encryption key generation algorithm $\mathsf{SKGen}(pp, msk, satt_b^*)$ to generate a sender encryption key $ek_{satt_b^*}$. Then it computes the challenge ciphertext $ct^* \leftarrow \mathsf{Enc}\ (pp,\ ek_{satt_b^*},\ \mu_b^*,\ ratt_b^*)$, and sends it to the adversary.

Query 2. In this phase, the adversary is allowed to submit the same key queries as that in Query 1, the challenger responds them as in Query 1.

Guess. In this phase, the experiment outputs 1 if the adversary's guess $b' = b$, and returns 0 otherwise.

Definition 10. *Let* $\mathsf{Adv}_{\mathcal{A},\Pi}^{priv}(\lambda)$ *denote the advantage function that a ppt adversary* $\mathcal{A}$ *wins in the above privacy experiment,*

$$\mathsf{Adv}_{\mathcal{A},\Pi}^{priv}(\lambda) := |\Pr[\mathsf{PRIV}_{\mathcal{A}}^{\Pi}(\lambda) = 1] - 1/2|.$$

If the advantage function $\mathsf{Adv}_{\mathcal{A},\Pi}^{priv}(\lambda)$ *is negligible in* λ*, we say that the key-policy AB-ME scheme* Π *satisfies privacy security.*

Next we describe the authentication security experiment $\mathsf{AUTH}_{\mathcal{A}}^{\Pi}(\lambda)$ played between a ppt adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.

The authentication experiment $\mathsf{AUTH}_{\mathcal{A}}^{\Pi}(\lambda)$.

The authentication security experiment consists of the following several phases.

Setup. In this phase, the challenger first runs the setup algorithm $\mathsf{Setup}(1^\lambda, \ell, N)$ to generate the public parameter pp and the master secret key msk, and sends pp to the adversary.

Query. In this phase, the adversary can issue the sender encryption key queries, the receiver private key queries and the policy key queries.

- When the adversary $\mathcal{A}$ submits a sender attribute $satt$, the challenger $\mathcal{C}$ invokes the sender encryption key generation algorithm $\mathsf{SKGen}(pp, msk, satt)$ to generate a sender encryption key ek_{satt} and delivers it to $\mathcal{A}$.
- When $\mathcal{A}$ issues a receiver private key query for a policy f_r, the challenger $\mathcal{C}$ runs the receiver private key generation algorithm $\mathsf{RKGen}(pp, msk, f_r)$ to generate a receiver private key rk_{f_r} and sends it to the adversary.

- When $\mathcal{A}$ commits a policy key query for a policy f_s, the challenger $\mathcal{C}$ executes the policy key generation algorithm $\mathsf{PokGen}(pp, msk, f_s)$ to generate a policy key pok_{f_s} and sends it to the adversary.

Forge. In this phase, the adversary outputs a forgery $((ct^*, ratt^*, satt^*), f_r^*, f_s^*)$. The challenger runs the receiver private key generation algorithm RKGen $(pp,$ $msk,$ $f_r^*)$ to generate the receiver private key $rk_{f_r^*}$ and the policy key generation algorithm $\mathsf{PoKGen}(pp, msk, f_s^*)$ to generate the policy key $pok_{f_s^*}$. Then it computes $m \leftarrow \mathsf{Dec}(pp, rk_{f_r^*}, pok_{f_s^*}, (ct^*, ratt^*, satt^*))$. If $satt^* \notin \mathcal{Q}$ and $m \neq \perp$, the experiment outputs 1, otherwise, it outputs 0, where $\mathcal{Q}$ denotes the encryption key query set in the Query phase.

Definition 11. *Let* $\mathsf{Adv}^{auth}_{\mathcal{A},\Pi}(\lambda)$ *denote the advantage function that a ppt adversary* $\mathcal{A}$ *wins in the above authentication experiment,*

$$\mathsf{Adv}^{auth}_{\mathcal{A},\Pi}(\lambda) := |\Pr[\mathsf{AUTH}^{\Pi}_{\mathcal{A}}(\lambda) = 1]|.$$

If the advantage function $\mathsf{Adv}^{auth}_{\mathcal{A},\Pi}(\lambda)$ *is negligible in* λ*, we say that the key-policy AB-ME scheme* Π *satisfies authentication security.*

C The proof of Theorem 6

Proof. In the privacy security experiment of the AB-ME scheme, we denote a ppt adversary by $\mathcal{A}$ and a challenger by $\mathcal{C}$. We give the privacy security proof via defining a series of games from G_0 (which is the original privacy security experiment) to G_2 (which is independent of the challenge bit), and proving that any two adjacent games are indistinguishable. The definitions for these games are as follows.

- G_0: This game is the original privacy security experiment.

  Note that at the beginning of the game, the adversary first submits a challenge receiver and sender attribute pair $((ratt_0, satt_0), (ratt_1, satt_1))$.

  1. **Setup**: In this phase, the challenger gets a public parameter pp and a master secret key msk by computing $(pp, msk) \leftarrow \mathsf{Setup}(1^\lambda, \ell, N)$ and sends pp to the adversary $\mathcal{A}$.

  2. **Query 1**: In this phase, the challenger answers the adversary's sender key queries for attributes $satt$, receiver key queries for policy f_r and policy key queries for policy f_s by invoking the sender key generation algorithm $ek_{satt} \leftarrow \mathsf{BACABE.SKGen}(pp, msk, satt)$, the receiver key generation algorithm $rk_{f_r} \leftarrow \mathsf{BACABE.RKGen}(pp, msk, f_r)$, and the policy key generation algorithm $pok_{f_s} \leftarrow \mathsf{BACABE.PoKGen}(pp, msk, f_s)$. Note that these queries follow the restrictions $f_r(ratt_b) \neq 0$ and $f_s(satt_b) \neq 0$ for $b \in \{0, 1\}$.

  3. **Challenge**: In this phase, the adversary $\mathcal{A}$ sends a challenge message pair (μ_0, μ_1) to the challenger, the challenger first picks a random challenge bit

$b \leftarrow_{\$} \{0,1\}$ and a random lock string $\alpha^* \leftarrow_{\$} \{0,1\}^s$, then invokes the algorithm $\mathsf{BACABE.Enc}(pp, ek_{satt_b}, \alpha^*, ratt_b)$ to encrypt α^* and gets a ciphertext $ct^*_{\alpha^*}$. Next, it constructs a decryption circuit

$$P := \mathsf{BACABE.Dec}(pp, \cdot, \cdot, (ct^*_{\alpha^*}, ratt_b, satt_b)),$$

and computes an intermediate signature σ^* as in the encryption algorithm $\mathsf{BACABE.Enc}(pp, ek_{satt_b}, \mu_b, ratt_b)$, then uses the decryption circuit to compute an obfuscated program

$$\widetilde{P} \leftarrow \mathsf{Obf}(1^\lambda, P, ratt_b, satt_b)), \mu_b||\sigma^*||ratt_b||satt_b||ct^*_{\alpha^*}, \alpha^*).$$

Finally the challenge ciphertext $ct^* = \widetilde{P}$ is submitted to $\mathcal{A}$.

4. **Query 2**: In this phase, $\mathcal{A}$ submits the same queries as in Query 1, and the challenger handles these queries as Query 1. Similarly, the queries need to follow the restrictions $f_r(ratt_b) \neq 0$ and $f_s(satt_b) \neq 0$ for each $b \in \{0,1\}$.

5. **Guess**: $\mathcal{A}$ outputs a bit b' as its guess and wins when $b' = b$.

– G_1: This game is the same as game G_0 except that the challenger computes the challenge ciphertext as the encryption of all zeros instead of $\alpha^* \leftarrow_{\$} \{0,1\}^s$. More concretely, the challenge ciphertext is computed as follows.

3. **Challenge**: The adversary $\mathcal{A}$ submits a challenge message pair (μ_0, μ_1) to the challenger, the challenger first picks a random challenge bit $b \leftarrow_{\$} \{0,1\}$ and a random lock string $\alpha^* \leftarrow_{\$} \{0,1\}^s$, then invokes the algorithm $\mathsf{BACABE.Enc}(pp, ek_{satt_b}, 0^s, ratt_b)$ to encrypt 0^s and gets a ciphertext $ct^*_{0^s}$. Next it first computes an intermediate signature σ^* as in the algorithm $\mathsf{BACABE.Enc}(pp, ek_{satt_b}, \mu_b, ratt_b)$ and constructs a decryption circuit $P := \mathsf{BACABE.Dec}(pp, \cdot, \cdot, (ct^*_{0^s}, ratt_b, satt_b))$, then uses the circuit to compute an obfuscated program

$$\widetilde{P} \leftarrow \mathsf{Obf}(1^\lambda, P, \mu_b||\sigma^*||ratt_b||satt_b||ct^*_{0^s}, \alpha^*).$$

Finally the challenge ciphertext is set as $ct^* = \widetilde{P}$ and sent to the adversary.

– G_2: This game is the same as game G_1 except that the challenger does not choose the lock α anymore, but simulates the obfuscated program $\widetilde{P}$ instead of generating it honestly as an obfuscation of the bilateral access control ABE decryption circuit. More concretely, the challenge ciphertext is simulated as follows.

3. **Challenge**: The adversary $\mathcal{A}$ submits a challenge message pair (μ_0, μ_1) to the challenger, the challenger first picks a random challenge bit $b \leftarrow_{\$} \{0,1\}$, and computes a simulated obfuscation program

$$\widetilde{P} \leftarrow \mathsf{Sim}(1^\lambda, 1^k, 1^{N'}),$$

where

$$k = |\mathsf{BACABE.Dec}(pp, \cdot, \cdot, (ct^*_{0^s}, ratt_b, satt_b))|,$$
$$N' = |\mu_b||\sigma^*||ratt_b||satt_b||ct^*_{0^s}|.$$

Finally the ciphertext is set as $ct^* = \widetilde{P}$ and sent to $\mathcal{A}$.

Let $\mathsf{Pr}[\mathsf{G}_i]$ denote the probability that the adversary wins in game G_i for $i \in \{0,1,2\}$. To complete the proof, we establish a sequence of lemmas that no PPT adversary can distinguish any two adjacent games with non-negligible probability. Next we prove that $|\mathsf{Pr}[\mathsf{G}_i] - \mathsf{Pr}[\mathsf{G}_{i+1}]|$ is negligible for each $i \in \{0,1\}$ by these lemmas.

Lemma 14. *Assume that the bilateral access control ABE scheme* BACABE *is sIND-CPA secure, then we have*

$$|\mathsf{Pr}[\mathsf{G}_0] - \mathsf{Pr}[\mathsf{G}_1]| \leq \mathsf{Adv}^{sind\text{-}cpa}_{\mathsf{BACABE},\mathcal{B}_1}(\lambda).$$

Proof. Assume that there exists a ppt adversary $\mathcal{A}$ that can distinguish games G_0 and G_1, then we can construct a ppt reduction algorithm $\mathcal{B}_1$ that can break the sIND-CPA security of the scheme BACABE.

After $\mathcal{B}_1$ receives the public parameter pp from its challenger, it sends pp to the adversary $\mathcal{A}$. Note that at this time, we assume that the adversary has committed a challenge receiver and sender attribute pair $((ratt_0, satt_0), (ratt_1, satt_1))$.

When $\mathcal{A}$ makes the sender key queries, the receiver key queries and the policy key queries, the algorithm $\mathcal{B}_1$ makes the same queries to its challenger and gets the corresponding keys and sends them to $\mathcal{A}$. Note that these queries need to follow the restrictions required in the security definition.

When $\mathcal{A}$ submits a challenge message pair (μ_0, μ_1), $\mathcal{B}_1$ computes the challenge ciphertext as follows. It first picks a random challenge bit $b \leftarrow_{\$} \{0,1\}$ and a random lock string $\alpha^* \leftarrow_{\$} \{0,1\}^s$, then submits $(\alpha^*, 0^s)$ to its challenger, from which it gets a partial challenge ciphertext $ct^*_{\alpha^*/0^s}$. Next it constructs a decryption circuit

$$P := \mathsf{BACABE.Dec}(pp, \cdot, \cdot, (ct^*_{\alpha^*/0^s}, ratt_b, satt_b)),$$

and computes an intermediate signature σ^* as in the encryption algorithm $\mathsf{BACABE.Enc}(pp, ek_{satt_b}, \mu_b, ratt_b)$, then uses the decryption circuit to compute an obfuscated program

$$\widetilde{P} \leftarrow \mathsf{Obf}(1^\lambda, P, \mu_b || \sigma^* || ratt_b || satt_b || ct^*_{\alpha^*/0^s}, \alpha^*).$$

Finally the ciphertext is set as $ct^* = \widetilde{P}$ and submitted to $\mathcal{A}$.

When the adversary makes the sender key queries, the receiver key queries and the policy key queries, the algorithm $\mathcal{B}_1$ responds them as above.

Finally, $\mathcal{A}$ outputs a guess b' and submits it to $\mathcal{B}_1$.

When the adversary's guess $b' = b$, $\mathcal{B}_1$ outputs α^*, otherwise, it outputs 0^s. Obviously, the challenge ciphetext ct^* in the first case corresponds to the challenge ciphetext in game G_0, and ct^* in the second case corresponds to the challenge ciphetext in game G_1. Thus we have

$$|\mathsf{Pr}[\mathsf{G}_0] - \mathsf{Pr}[\mathsf{G}_1]| \leq \mathsf{Adv}^{sind\text{-}cpa}_{\mathsf{BACABE},\mathcal{B}_1}(\lambda).$$

Lemma 15. *Assume that the lockable obfuscation scheme* LOBF *is secure, then we have*

$$|\mathsf{Pr}[\mathsf{G}_1] - \mathsf{Pr}[\mathsf{G}_2]| \leq \mathsf{Adv}^{lsim}_{\mathsf{LOBF},\mathcal{B}_2}(\lambda).$$

Proof. Assume that there exists a ppt adversary $\mathcal{A}$ which can distinguish games G_1 and G_2, then we can design a ppt reduction algorithm $\mathcal{B}_2$ that can break the security of the lockable obfuscation scheme LOBF. The reduction is described as follows.

The reduction algorithm $\mathcal{B}_2$ first invokes the setup algorithm $\mathsf{BACABE.Setup}$ to generate the public parameter pp and the master secret key msk and sends pp to the adversary $\mathcal{A}$.

When $\mathcal{A}$ makes the sender key queries, the receiver key queries and the policy key queries, $\mathcal{B}_2$ invokes the sender key generation algorithm, the receiver key generation algorithm and the policy key generation algorithm to generate the corresponding keys and sends them to $\mathcal{A}$. Note that these queries follow the restrictions as required in the privacy security definition.

At the challenge phase, when the adversary $\mathcal{A}$ submits a challenge message pair (μ_0, μ_1), the reduction algorithm $\mathcal{B}_2$ first chooses a random bit $b \in \{0,1\}$, and invokes the encryption algorithm $\mathsf{BACABE.Enc}(pp, ek_{satt_b}, 0^s, satt_b)$ to generate a ciphertext $ct^*_{0^s}$. Then it generates a signature σ^* as in game G_1 and constructs a decryption circuit

$$P := \mathsf{BACABE.Dec}(pp, \cdot, \cdot, (ct^*_{0^s}, ratt_b, satt_b)).$$

Meanwhile it sends the circuit along with the message $\mu_b||\sigma^*||ratt_b||satt_b||ct^*_{0^s}$ to its challenger. After receiving the obfuscated program $\widetilde{P}$ from its challenger, $\mathcal{B}_2$ sets $ct^* \leftarrow \widetilde{P}$ and sends it to $\mathcal{A}$. Next, $\mathcal{A}$ makes more key queries and $\mathcal{B}_2$ responds them as before.

Finally, $\mathcal{A}$ outputs a guess b', if $b' = b$, $\mathcal{B}_2$ outputs 1 (i.e., the bilateral access control ABE decryption circuit was obfuscated), otherwise, it outputs 0 (i.e., the obfuscated program is simulated).

Analysis. If $\widetilde{P}$ is the obfuscation of the decryption circuit

$$\mathsf{BACABE.Dec}(pp, \cdot, \cdot, (ct^*_{0^s}, ratt_b, satt_b)),$$

for some lock α, then $\mathcal{B}_2$ simulates game G_1 for $\mathcal{A}$, otherwise, it simulates game G_2 for $\mathcal{A}$. By $\mathsf{Pr}[\mathsf{G}_2] = \frac{1}{2}$ and Lemmas 14 and 15, we have

$$\mathsf{Adv}^{priv}_{\mathsf{ABME},\mathcal{A}}(\lambda) = |\mathsf{Pr}[\mathsf{G}_0] - \frac{1}{2}| = |\mathsf{Pr}[\mathsf{G}_0] - \mathsf{Pr}[\mathsf{G}_2]| = |\mathsf{Pr}[\mathsf{G}_0] - \mathsf{Pr}[\mathsf{G}_1] + \mathsf{Pr}[\mathsf{G}_1] - \mathsf{Pr}[\mathsf{G}_2]|$$

$$\leq |\mathsf{Pr}\mathsf{G}_0] - \mathsf{Pr}[\mathsf{G}_1]| + |\mathsf{Pr}[\mathsf{G}_1] - \mathsf{Pr}[\mathsf{G}_2]| \leq \mathsf{Adv}^{sind\text{-}cpa}_{\mathsf{BACABE},\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{lsim}_{\mathsf{LOBF},\mathcal{B}_2}(\lambda),$$

which proves Theorem 6.

D The proof of Theorem 7

Proof. Assume that there exists a ppt adversary $\mathcal{A}$ that can break the authentication of the AB-ME scheme ABME, then we can construct a ppt reduction algorithm $\mathcal{B}_3$ that can break the authentication of the bilateral access control ABE scheme BACABE. The reduction is described as follows.

Assume that at the beginning of the simulation, the reduction algorithm $\mathcal{B}_3$ has received the public parameter pp and has sent it to the adversary $\mathcal{A}$.

When $\mathcal{A}$ makes the sender key queries, the receiver key queries and the policy key queries, the reduction algorithm $\mathcal{B}_3$ makes the same queries to its challenger and gets the corresponding keys and sends them to $\mathcal{A}$. Note that these queries follow the restrictions required in the authentication security definition.

Forge. In this phase, the adversary $\mathcal{A}$ outputs a forgery (ct^*, f_r^*, f_s^*), where $ct^* = \widetilde{P}$. The reduction algorithm $\mathcal{B}_3$ first makes the receiver key query for policy f_r^* and the policy key query for policy f_s^* to its challenger and gets the receiver key $rk_{f_r^*}$ and policy key $pok_{f_s^*}$. Then $\mathcal{B}_3$ invokes the evaluation algorithm $\mathsf{Eval}(\widetilde{P}, rk_{f_r^*}, pok_{f_s^*})$ with $rk_{f_r^*}$ and $pok_{f_s^*}$ as inputs and gets $\mu^*||\sigma^*||ratt^*||satt^*||ct_{\alpha^*}$. If the attribute $satt^*$ has been made the sender key queries, $\mathcal{B}_3$ outputs $\perp$, otherwise, $\mathcal{B}_3$ checks whether the message/signature pair $(\mu^*||ratt^*, \sigma^*)$ under the sender attribute $satt^*$ satisfies the signature verification of the IBS scheme. If so, $\mathcal{B}_3$ sends $((ct_{\alpha^*}, ratt^*, satt^*), f_r^*, f_s^*)$ as a forgery to its challenger.