PriSrv+: Privacy and Usability-Enhanced Wireless Service Discovery with Fast and Expressive Matchmaking Encryption

Anonymous Authors

***Abstract*—Service discovery is a fundamental process in wireless networks, enabling devices to find and communicate with services dynamically, and is critical for the seamless operation of modern systems like 5G and IoT. This paper introduces PriSrv+, an advanced privacy and usability-enhanced service discovery protocol for modern wireless networks and resource-constrained environments. PriSrv+ builds upon PriSrv (NDSS'24), by addressing critical limitations in expressiveness, privacy, scalability, and efficiency, while maintaining compatibility with widely-used wireless protocols such as mDNS, BLE, and Wi-Fi.**

A key innovation in PriSrv+ is the development of Fast and Expressive Matchmaking Encryption (FEME), the first matchmaking encryption scheme capable of supporting expressive access control policies with an unbounded attribute universe, allowing any arbitrary string to be used as an attribute. FEME significantly enhances the flexibility of service discovery while ensuring robust message and attribute privacy. Compared to PriSrv, PriSrv+ optimizes cryptographic operations, achieving 7.62$\times$ faster for encryption and 6.23$\times$ faster for decryption, and dramatically reduces ciphertext sizes by 87.33%. In addition, PriSrv+ reduces communication costs by 87.33% for service broadcast and 86.64% for anonymous mutual authentication compared with PriSrv. Formal security proofs confirm the security of FEME and PriSrv+. Extensive evaluations on multiple platforms demonstrate that PriSrv+ achieves superior performance, scalability, and efficiency compared to existing state-of-the-art protocols.

***Index Terms*—Privacy-preserving service discovery, fast and expressive matchmaking encryption**

1. Introduction

Service discovery (SD) protocols, such as Wi-Fi [1], AirDrop [2], and Bluetooth Low Energy (BLE) [3], enable devices to dynamically detect and advertise services in network environments. These protocols streamline device interaction by facilitating automatic service discovery. However, they also introduce significant privacy concerns, particularly regarding the exposure of sensitive information during their execution [11], [15]. Despite their widespread use, many SD protocols, such as DNS-SD [5], mDNS [19], SSDP [14], and UPnP [6], lack adequate privacy protections, making them susceptible to attacks like man-in-the-middle (MitM), spoofing, and denial-of-service (DoS) [8], [28]. This vulnerability is exacerbated by the use of cleartext advertisements in Wi-Fi and BLE, which reveal device identifiers and allow adversaries to track and profile users [27].

Although some protocols, such as AirDrop [2], PrivateDrop [17], and WTSB [30], incorporate encryption and authentication mechanisms to enhance privacy during service discovery, they still suffer from tracking, MitM, and DoS attacks due to the absence of comprehensive privacy features like attribute hidden and policy control [8], [25]. Another protocol, CBN [11], features anonymous client authentication but fails to secure service providers, leaving it vulnerable to impersonation and other attacks.

Recently, Yang et al. [32] developed PriSrv in NDSS'24, offering a comprehensive solution to many of the privacy and usability challenges present in the existing SD protocols. By leveraging its dual-layer architecture, PriSrv ensures that services are only discoverable by authorized clients, mitigating the risks of exposing sensitive information during device interaction. In contrast to protocols like AirDrop and BLE, which lack robust privacy mechanisms, PriSrv enhances mutual authentication and shields private attributes of involved parties from unauthorized access. This mechanism allows PriSrv to counteract common vulnerabilities in SD protocols, such as MitM attacks, user tracking, and device profiling, by applying anonymous credential-based matchmaking encryption (ACME) that supports bilateral policy control for secure communication.

However, despite its strengths, PriSrv has limitations due to its reliance on ACME. One major drawback is the leakage of public attributes in the outer layer, which can still enable tracking attacks. The use of binary attribute vectors (each attribute can only represent 1 or 0) significantly constrains the expressiveness of attributes, and increases the computational costs during encryption and decryption due to large vector size in case of numerous attributes. Additionally, ACME's small-universe construction limits the total number of attributes in the system. Adding new attributes requires a full system rebuild, reducing flexibility for large-scale deployments. Furthermore, the large size of the service discovery broadcast ciphertext in PriSrv results in high transmission overhead and delays, particularly constraining its usage in low-bandwidth environments such as BLE and mDNS. The requirement for pre-issued anonymous credentials in PriSrv also adds complexity to credential management, potentially affecting system efficiency.

Aiming to address these issues, we propose a fast and expressive matchmaking encryption (FEME) scheme, which is also of *independent interest* in advancing matchmaking encryption (ME) techniques. FEME ensures high efficiency and robust privacy.

Unlike earlier ME schemes, such as Ateniese et al.'s Identity-Based Matchmaking Encryption (IBME) instantiation [7] and Chen et al.'s IBME scheme [13], which are constrained to basic equality policies, FEME supports expressive policies with arbitrary attribute values. It *solves the open problem* highlighted by Ateniese et al. in CRYPTO'19 and Chen et al. in ASIACRYPT'22, calling for the development of matchmaking encryption that balances policy expressiveness with privacy. Additionally, FEME is significantly faster in encryption and decryption than existing ME schemes supporting expressive policy control. Based on a partially hidden access structure and a randomness splitting technique, FEME preserves the privacy of expressive policies and attribute values while balancing privacy and efficiency in matchmaking encryption.

Building on the strengths of FEME, PriSrv+ overcomes the shortcomings of its predecessor, PriSrv, while introducing new features. PriSrv+ enhances the usability of service discovery by eliminating its dependency on anonymous credentials and the management overhead associated with credential issuance and revocation. PriSrv+ achieves high expressiveness in bilateral policy control and attribute representation, breaking the constraints of binary attribute vectors and a small attribute universe, as seen in PriSrv. PriSrv+ significantly reduces communication overhead by decreasing the size of broadcast messages by up to 87.33%, thereby improving its scalability and efficiency, which is crucial in low-bandwidth environments. Furthermore, it enhances privacy by ensuring that outer layer attributes in PriSrv remain protected during the service discovery process, offering a comprehensive solution for private service discovery.

The key contributions of PriSrv+ are outlined as follows.

• *Fast and Expressive Matchmaking Encryption (FEME)*. At the core of PriSrv+, FEME is the first matchmaking encryption scheme capable of supporting expressive access control policies with an unbounded attribute universe, allowing any arbitrary string to be used as an attribute. This solves an open problem identified by Ateniese et al. in CRYPTO'19 and by Chen et al. in ASIACRYPT'21. FEME offers up to 7.62× faster encryption and 6.23× faster decryption compared with ACME, making PriSrv+ suitable for resource-constrained environments.

• *Enhanced Protocol Scalability and Flexibility*. PriSrv+ significantly improves the scalability over PriSrv by supporting unrestricted attribute space. Attributes in PriSrv+ can be arbitrary strings, such as postal addresses, eliminating the restriction of rigid binary vectors used in PriSrv. This enhancement provides greater flexibility in service discovery and access control management, enabling the applicability of PriSrv+ across diverse real-world settings while maintaining low computation and communication overheads.

• *Optimized Performance and Scalability*. By reducing ciphertext size and optimizing cryptographic operations, PriSrv+ significantly lowers packet transmission overhead, leading to up to 7.17× faster service broadcast and 3.32× faster anonymous mutual authentication compared to PriSrv. This positions PriSrv+ as a more efficient and scalable protocol, particularly suitable for bandwidth-limited and latency-sensitive networks.

• *Interoperability with Existing Protocols*. PriSrv+ maintains compatibility with widely-used wireless protocols such as mDNS, BLE, EAP, AirDrop, and Wi-Fi, while addressing scalability issues in PriSrv. In comparison to PriSrv, for instance, PriSrv+ reduces the packet size in mDNS by 88.89%, in BLE by 87.73%, and in Wi-Fi by 86.64%, which makes it more suitable for low-bandwidth environments.

• *Versatile Implementation across Platforms*. PriSrv+ has been tested on a range of platforms, including desktops, laptops, mobile devices, and IoT systems like Raspberry Pi. Experimental results indicate that PriSrv+ reduces delays in both privacy-preserving service broadcast and mutual authentication, delivering immediate responses even in resource-constrained environments. On average, PriSrv+ performs 7.17× faster for broadcast and 3.32× faster for anonymous mutual authentication than its predecessor PriSrv, demonstrating its practical efficiency in real-world scenarios.

• *Formal Security and Privacy Guarantees*. Rigorous formal security proofs demonstrate that FEME satisfies confidentiality, anonymity, and authenticity. PriSrv+ is a secure service discovery protocol with bilateral anonymity, offering superior protection compared to other state-of-the-art protocols.

These contributions establish PriSrv+ as an efficient, secure, and scalable solution for wireless networks, offering robust security and privacy guarantees and adaptability to the evolving demands of modern communication systems.

2. Related Work

2.1. Service Discovery Protocols

Service discovery (SD) protocols such as Wi-Fi [1], AirDrop [2], and Bluetooth Low Energy (BLE) [3] are essential for facilitating the detection and advertisement of services and devices within dynamic network environments. These protocols enable devices to discover available services automatically, streamlining device interactions. However, SD protocols present considerable privacy risks, especially for users seeking to protect identifying or sensitive information during their interactions. Research indicates that approximately 90% of users consider the exposure of device names to be a privacy concern [18], as it allows adversaries to infer personal details, including location, mobility patterns, and user profiles [24], [25], [30], [33]. For instance, in public Wi-Fi networks, device names may provide Internet Service Providers (ISPs) with tracking capabilities [11], and in IoT networks, attackers can extract information from smart devices to deduce users' routines [15].

Most existing SD protocols, such as DNS-SD [5], mDNS [19], SSDP [14], and UPnP [6], lack strong privacy safeguards, making them susceptible to attacks like man-in-the-middle (MitM), spoofing, and denial-of-service (DoS) [8],

[28]. These vulnerabilities are worsened by the widespread use of cleartext broadcasts in Wi-Fi and BLE, which exposes device identifiers, allowing adversaries to track and profile users [27]. While some protocols, like the CBN scheme [11], offer a level of anonymous client authentication, they do not provide adequate protections for service providers, leaving them exposed to impersonation and MitM attacks.

Protocols like AirDrop [2], PrivateDrop [17], and WTSB [30] have introduced encryption and authentication techniques to enhance privacy during service discovery. These protocols strengthen mutual authentication and anonymity, yet they still exhibit weaknesses to tracking, MitM, and DoS attacks due to incomplete privacy features, such as selective attribute disclosure and multi-show unlinkability [8], [25]. For instance, AirDrop and PrivateDrop's reliance on client and server certificates can be exploited by attackers to link sessions and track users across interactions [17].

Yang et al. introduced PriSrv [32], a private service discovery protocol that enables service providers and clients to specify fine-grained access control policies, ensuring mutual authentication while concealing private information. PriSrv incorporates Anonymous Credential-based Matchmaking Encryption (ACME) to achieve bilateral policy control, selective attribute disclosure, and multishow unlinkability. PriSrv's primary limitation is its large message size, which leads to high transmission overhead and reception delays, particularly constraining its usage in slower networks like BLE and congested Wi-Fi channels. This scalability challenge affects the protocol's efficiency and performance. Another limitation is its leakage of public attributes, which may result in tracing and profiling attacks.

Therefore, there is a critical need for private service discovery protocols that provide stronger privacy protections and enhanced usability, a gap that PriSrv+ is designed to fill.

2.2. Matchmaking Encryption (ME)

Matchmaking encryption (ME) was first introduced by Ateniese et al. [7] in CRYPTO'19 as a new encryption paradigm where both the sender and receiver can define policies that must be satisfied for the message to be decrypted. In ME [7], a sender, associated with an identity or attribute σ, specifies a target identity or policy $\mathbb{R}$ during encryption, while a receiver, associated with identity or attribute ρ, sets a target identity or policy $\mathbb{S}$. Successful decryption occurs only if both the sender's σ satisfies the receiver's $\mathbb{S}$ and the receiver's ρ satisfies the sender's $\mathbb{R}$. Ateniese et al. also instantiated Identity-Based Matchmaking Encryption (IBME) in the random oracle model, where sender and receiver identities are specified as equality-based policies, and a sender's encryption key is embedded in the ciphertext to authenticate the sender.

Later, Francati et al. [16] in INDOCRYPT'21 developed an IBME under non-standard assumptions in the standard model using non-interactive zero-knowledge (NIZK) proofs. Chen et al. [13] in ASIACRYPT'22 further proposed an IBME scheme under standard assumptions in the standard model. Although these IBME schemes ensure data privacy and authenticity, they are limited to equality-based policies and one-to-one data sharing.

To enable one-to-many secure data sharing in ME, Sun et al. [26] in TIFS'23 proposed a privacy-aware ME scheme (PSME), and Yang et al. [31] in TIFS'23 introduced a certificateless ME scheme (CLME), both of which extend IBME to multi-user scenarios using identity-based broadcast encryption. Additionally, Wu et al. [29] in TIFS'23 presented a fuzzy IBME (FBME) that supports fuzzy bilateral access control, allowing decryption if the overlap between the sender's and receiver's attribute sets exceeds a threshold. However, FBME supports threshold-based policy matching with limited expressiveness and incurs high computation costs during decryption.

More recently, Yang et al. [32] in NDSS'24 developed an anonymous credential-based matchmaking encryption (ACME) scheme that supports flexible bilateral policy control. While ACME offers valuable functionality, it is limited by large ciphertext size and a small-universe construction. Since the attribute sets used in ACME are denoted by binary attribute vectors, ACME requires large vectors to represent attributes in a wide range, leading to increased computation costs in both encryption and decryption. In comparison, FEME not only matches ACME's ability to support monotonic Boolean formulas as policies but also extends expressivity to an unrestricted attribute space, enabling any arbitrary string to be used as an attribute. Moreover, FEME significantly improves efficiency, reducing ciphertext size by 87.33%, and achieving encryption and decryption speeds of 7.62$\times$ and 6.23$\times$ faster, respectively.

3. Preliminary

We present notations, bilinear pairing, access structure, linear secret sharing scheme, and partially hidden access structure, for constructing FEME and PriSrv+.

3.1. Notation and Bilinear Pairing

Let integers m and n satisfy $m < n$, with $[m, n]$ representing the set $\{m, m+1, ..., n\}$, and $[n]$ denoting the set $\{1, ..., n\}$. For a prime p, define $\mathbb{Z}_p$ as the set $\{0, 1, ..., p-1\}$, where addition and multiplication are performed modulo p. The set $\mathbb{Z}_p^*$ excludes 0 from $\mathbb{Z}_p$. The security parameter is denoted by λ. We use bold lowercase letters for vectors and bold uppercase letters for matrices. A vector $\mathbf{v}$ denotes a column vector by default, and $\mathbf{v}_k$ represents its k-th element. For a matrix $\mathbf{M}$, $\mathbf{M}_i$ is the i-th row, and $\mathbf{M}_{i,j}$ denotes the element at position (i, j).

For a set S, the notation $s \xleftarrow{\$} S$ indicates that s is uniformly sampled from S. The notation $y \leftarrow \mathsf{Algo}(x)$ refers to the output y after running algorithm Algo on input x. An algorithm is probabilistic polynomial time (PPT) if it runs in polynomial time with respect to the input length. We assume a master public key is an implicit input to all algorithms in a scheme.

A bilinear group with Type-III pairings is defined as $\mathcal{BG} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, p)$, where there is no efficiently computable isomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$. For any $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$, the pairing $e(g_1, g_2)$ maps to $\mathbb{G}_T$. For $a, b \xleftarrow{\$} \mathbb{Z}_p^*$, one has $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$.

3.2. Access Structure

Definition 1 (Access Structure [9]). *Let $\{P_1, \cdots, P_n\}$ be a set of parties. A collection $\mathbb{A} \subseteq 2^{\{P_1,\cdots,P_n\}}$ is monotone if $\forall B, C$: if $B \in \mathbb{A}$ and $B \subseteq C$, then $C \in \mathbb{A}$. An access structure (respectively, monotone access structure) is a collection (respectively, monotone collection) $\mathbb{A}$ of non-empty subsets of $\{P_1, \cdots, P_n\}$, i.e., $\mathbb{A} \subseteq 2^{\{P_1,\cdots,P_n\}} \backslash \{\varnothing\}$. The sets in $\mathbb{A}$ are called authorized sets, and the sets not in $\mathbb{A}$ are called unauthorized sets.*

An access structure is said to be monotone if, for any two sets S and T of attributes, $S \subseteq T$ and S being authorized imply that T is also authorized. It ensures that any user possessing a set of attributes that satisfies the access policy continues to have access if additional attributes are granted.

3.3. Linear Secret Sharing Scheme (LSSS)

Definition 2 (Linear Secret Sharing Scheme (LSSS) [9]). *A secret sharing scheme Π over a set of parties $\mathcal{P}$ is called linear (over $\mathbb{Z}_p$) if (1) the shares of each party form a vector over $\mathbb{Z}_p$. (2) there exists a matrix $\mathbf{A}$ with m rows and n columns called the share-generating matrix for Π. For all $i = 1, \cdots, m$, the i-th row of $\mathbf{A}$ is labeled by a party $\rho(i)$ (ρ is a function from $\{1, \cdots, m\}$ to $\mathcal{P}$). When we consider the column vector $v = (s, r_2, \cdots, r_n)$, where $s \in \mathbb{Z}_p$ is the secret to be shared, and $r_2, \cdots, r_n \in \mathbb{Z}_p$ are randomly chosen, then $\mathbf{A}v$ is the vector of m shares of the secret s according to Π. The share $(\mathbf{A}v)_i$ belongs to party $\rho(i)$.*

As shown in [9], each LSSS possesses the linear reconstruction property, which is defined as follows. Let Π represent an LSSS for the access structure $\mathbb{A}$, and let $S \in \mathbb{A}$ be an authorized set with $I \subset \{1, \cdots, m\}$, where $I = \{i | \rho(i) \in S\}$. There exists a set of constants $\{\omega_i \in \mathbb{Z}_p\}_{i \in I}$ such that, given any valid shares $\{\lambda_i\}$ of a secret s in Π, the relationship $\sum_{i \in I} \omega_i \lambda_i = s$ holds. Let A_i represent the i-th row of $\mathbf{A}$, we similarly have $\sum_{i \in I} \omega_i \lambda_i = (1, 0, \cdots, 0)$. These constants $\{\omega_i\}$ are computable in time polynomial [9] in the size of $\mathbf{A}$. Notably, such constants $\{\omega_i\}$ cannot be constructed for unauthorized sets.

Boolean Formulas. Boolean formulae are a common way to model access control. LSSS is a more general class of functions and include Boolean formulas. Using established methods [9], any monotone Boolean formula can be transformed into an LSSS format. Such a formula can be structured as an access tree, where an access tree with m nodes yields an LSSS matrix of m rows.

3.4. Partially Hidden Access Structure

In a partially hidden access structure, attributes are divided into attribute names and attribute values, where only attribute names are exposed, while attribute values remain hidden. For example, consider an access policy that requires "Role: Admin **AND** Department: Research **OR** Level: Confidential" to access certain data, where "Role", "Department", and "Level" are attribute names, and "Admin", "Research", and "Confidential" are attribute values. In a partially hidden access structure, the policy is transformed to "Role **AND** Department **OR** Level", revealing attribute names only. This contrasts with a traditional access structure, where attributes are fully exposed in the policy.

Before defining a partially hidden access policy, we define the structures of an attribute set and an access policy. Let the attribute set be $\mathcal{S} = \{u_i\}_{i \in [\ell]}$ containing ℓ attributes, where each attribute belongs to a unique category. Each attribute is denoted as $u_i = \langle n_i, v_i \rangle$, with n_i representing the attribute name and v_i the attribute value. An access policy is defined as $\mathbb{A} = (\mathbf{M}, \pi, \mathcal{T})$, where $\mathbf{M}$ is an $m \times n$ access control matrix, $\mathbf{M}_i$ is the i-th row of $\mathbf{M}$, and π is a mapping function that associates each row $\mathbf{M}_i$ with an attribute $\pi(i)$. The policy $\mathcal{T}$ is expressed as $(\Psi_{\pi(1)}, \cdots, \Psi_{\pi(m)})$, where each $\Psi_{\pi(i)} = \langle n_{\pi(i)}, v_{\pi(i)} \rangle$ consists of an attribute name $n_{\pi(i)}$ and value $v_{\pi(i)}$.

In a partially hidden attribute set, attribute values v_i are concealed, leaving only attribute names n_i visible. The resulting attribute set is modified to $\mathcal{S}_{\text{partial}} = \{n_i\}_{i \in [\ell]}$. Similarly, in a partially hidden access policy, the attribute values $v_{\pi(i)}$ are removed from $\mathcal{T}$, exposing only the attribute names. The modified policy is represented as $\mathbb{A}_{\text{partial}} = (\mathbf{M}, \pi, \mathcal{T}_{\text{name}})$, where $\mathcal{T}_{\text{name}} = (n_{\pi(1)}, \cdots, n_{\pi(m)})$. This method conceals attribute values to enhance privacy while using attribute names for efficient policy matching. We define partial satisfaction $\mathcal{S}_{\text{partial}} \models \mathbb{A}_{\text{partial}}$ if the attribute names in $\mathcal{S}_{\text{partial}}$ match those in $\mathbb{A}_{\text{partial}}$. Full satisfaction ($\mathcal{S} \models \mathbb{A}$) requires matching both names and values, whereas partial satisfaction only matches attribute names.

4. Fast and Expressive Matchmaking Encryption (FEME)

We construct FEME, a fast and expressive matchmaking encryption scheme, as the core component of PriSrv+. It is also of independent interest for advancing matchmaking encryption (ME) techniques.

4.1. Technical Roadmap

In an ME system, both sender and receiver, each possessing a set of attributes, define access policies that the other must meet to decrypt any message. FEME features privacy-preserving policy matching and user anonymity. We leverage Attribute-Based Encryption (ABE) with expressive access policies to enable bilateral matching of the policies of both sender and receiver. ABE is available in two forms: ciphertext-policy ABE (CP-ABE) and key-policy ABE (KP-ABE), both essential for building FEME.

The design of FEME, as shown in Fig. 1, follows a structured, multi-stage roadmap that enhances existing ABE schemes to address privacy and efficiency challenges. We

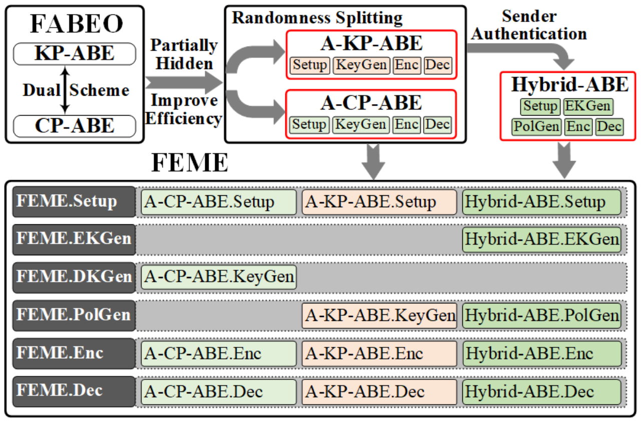


Figure 1: Technical Roadmap of FEME

build on FABEO [21], a dual-form KP-ABE and CP-ABE scheme[1] that supports expressive policies without restrictions on policy type or attribute range. However, while FABEO excels in policy expressiveness, it lacks privacy-preserving policy matching or anonymity. FABEO's CP-ABE exposes access policies with plaintext attribute values, and its KP-ABE reveals plaintext attribute values in attribute sets. Moreover, FABEO's decryption incurs high computation overhead due to pairing and exponentiation operations that scale with policy complexity.

FEME addresses these limitations in three distinct stages. Stage 1 enhances FABEO's KP-ABE and CP-ABE schemes, creating anonymous versions (A-KP-ABE in Fig. 2 and A-CP-ABE in Fig. 3), that hide attribute values in attribute sets and access policies, greatly improving computational efficiency. Stage 2 introduces Hybrid-ABE (Fig. 4), bridging the gap between ME with CP-ABE/KP-ABE and supporting bilateral policy-matching and *sender authentication*. Stage 3 integrates A-KP-ABE, A-CP-ABE, and Hybrid-ABE schemes to create FEME, an ME that enhances both privacy and efficiency.

4.2. Technical Details

Following the above roadmap, we transform FABEO into a privacy-preserving and efficient ME scheme.

Stage 1. We create A-CP-ABE and A-KP-ABE as anonymous variants of FABEO's CP-ABE and KP-ABE, respectively, using the following techniques.

(1) Partially Hidden Access Structure. To meet privacy and efficiency requirements, we adopt the *partially hidden access structure* (§3.4), which separates each attribute into a visible *attribute name* and concealed *attribute value*, ensuring sensitive attribute value remain protected. Although attribute names are visible in an attribute set (A-KP-ABE) or policy (A-CP-ABE) within a ciphertext, they are generally less sensitive, enabling substantial efficiency gains in FEME construction. Our A-CP-ABE and A-KP-ABE schemes eliminate tremendous costly pairing and exponentiation operations, making policy matching faster and better suited for resource-constrained environments.

1. Both schemes control access by matching attributes to policies, but they reverse the roles of the ciphertext and key in defining access control. They share the same design mechanism and common parameters.

A-KP-ABE: Anonymous KP-ABE

$\underline{\mathsf{Setup}(1^\lambda) \rightarrow (\mathsf{mpk}, \mathsf{msk})}$. Generate $\mathcal{G} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$. Pick $\alpha, b_1, b_2 \xleftarrow{\$} \mathbb{Z}_p^*$ and a hash function $H : \{0,1\}^* \rightarrow \mathbb{G}_1$. Compute $Z = e(g_1, g_2)^\alpha$, $\delta_1 = g_2^{b_1}$, $\delta_2 = g_2^{b_2}$. Output the master public key $\mathsf{mpk} := (\mathcal{G}, H, Z, \delta_1, \delta_2)$ and master secret key $\mathsf{msk} := (\alpha, b_1, b_2)$.
$\underline{\mathsf{KeyGen}(\mathsf{msk}, \mathbb{A} = (\mathbf{A}, \rho, \{\Psi_{\rho(i)}\}_{i\in[m]})) \rightarrow \mathsf{SK}_{\mathbb{A}}}$. Remind that $\{\Psi_{\rho(i)}\}_{i\in[m]} = \{\langle n_{\rho(i)}, v_{\rho(i)}\rangle\}_{i\in[m]}$. Pick $r' \xleftarrow{\$} \mathbb{Z}_p^*$ and $\mathbf{y} \xleftarrow{\$} \mathbb{Z}_p^{n-1}$. Compute $\mathsf{sk}_1 = g_2^{r'}$, $\mathsf{sk}_{2,i} = \left(g_1^{\mathbf{A}_i(\alpha||\mathbf{y})^\top} \cdot H(\Psi_{\rho(i)})^{r'}\right)^{\frac{1}{b_1}}$, $\mathsf{sk}_{3,i} = \left(g_1^{\mathbf{A}_i(\alpha||\mathbf{y})^\top} \cdot H(\Psi_{\rho(i)})^{r'}\right)^{\frac{1}{b_2}}$. Output $\mathsf{SK}_{\mathbb{A}} := ((\mathbf{A}, \rho, \{n_{\rho(i)}\}_{i\in[m]}), \mathsf{sk}_1, \{\mathsf{sk}_{2,i}, \mathsf{sk}_{3,i}\}_{i\in[m]})$.
$\underline{\mathsf{Enc}(\mathcal{S} = \{u_i\}_{i\in[\ell]} = \{\langle n_i, v_i\rangle\}_{i\in[\ell]}, \mathsf{msg}) \rightarrow \mathsf{CT}_{\mathcal{S}}}$. Pick $s', s'' \xleftarrow{\$} \mathbb{Z}_p^*$. Let $s = s' + s''$. Compute $\mathsf{ct}_0 = e(g_1, g_2)^{\alpha s} \cdot \mathsf{msg}$, $\mathsf{ct}_{1,i} = H(u_i)^s$, $\mathsf{ct}_2 = \delta_1^{s'}$, $\mathsf{ct}_3 = \delta_2^{s''}$. Output $\mathsf{CT}_{\mathcal{S}} := (\{n_i\}_{i\in[\ell]}, \mathsf{ct}_0, \{\mathsf{ct}_{1,i}\}_{i\in[\ell]}, \mathsf{ct}_2, \mathsf{ct}_3)$.
$\underline{\mathsf{Dec}(\mathsf{SK}_{\mathbb{A}}, \mathsf{CT}_{\mathcal{S}}) \rightarrow \mathsf{msg}/\perp}$. If there is any subset I that matches the attribute names $\{n_i\}_{i\in[\ell]}$ in CT with $(\mathbf{A}, \rho, \{n_{\rho(i)}\}_{i\in[m]})$ in SK, there exist constants $\{\omega_i\}_{i\in I}$ s.t. $\sum_{i\in I} \omega_i \mathbf{A}_i = (1, 0, \cdots, 0)$. Output

$$\mathsf{msg} = \frac{\mathsf{ct}_0 \cdot e(\prod_{i\in I}(\mathsf{ct}_{1,\rho(i)})^{\omega_i}, \mathsf{sk}_1)}{e(\prod_{i\in I}(\mathsf{sk}_{2,i})^{\omega_i}, \mathsf{ct}_2) \cdot e(\prod_{i\in I}(\mathsf{sk}_{3,i})^{\omega_i}, \mathsf{ct}_3)}.$$

Figure 2: A-KP-ABE Scheme

A-CP-ABE: Anonymous CP-ABE

$\underline{\mathsf{Setup}(1^\lambda) \rightarrow (\mathsf{mpk}, \mathsf{msk})}$. Generate $\mathcal{G} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$. Pick $\alpha \xleftarrow{\$} \mathbb{Z}_p^*$, $h \xleftarrow{\$} \mathbb{G}_1$ and a hash function $H : \{0,1\}^* \rightarrow \mathbb{G}_1$. Compute $Z = e(g_1, g_2)^\alpha$. Output the master public key $\mathsf{mpk} := (\mathcal{G}, H, Z, h)$ and master secret key $\mathsf{msk} := \alpha$.
$\underline{\mathsf{KeyGen}(\mathsf{msk}, \mathcal{S} = \{u_i\}_{i\in[\ell]} = \{\langle n_i, v_i\rangle\}_{i\in[\ell]}) \rightarrow \mathsf{SK}_{\mathcal{S}}}$.
Pick $r \xleftarrow{\$} \mathbb{Z}_p^*$. Computes $\mathsf{sk}_1 = g_1^\alpha h^r$, $\mathsf{sk}_3 = g_2^r$, $\mathsf{sk}_{2,i} = H(u_i)^r$ for $i \in [\ell]$. Output $\mathsf{SK}_{\mathcal{S}} := (\{n_i\}_{i\in[\ell]}, \mathsf{sk}_1, \{\mathsf{sk}_{2,i}\}_{i\in\ell}, \mathsf{sk}_3)$.
$\underline{\mathsf{Enc}(\mathbb{A} = (\mathbf{M}, \pi, \{\Psi_{\pi(i)}\}_{i\in[m]}), \mathsf{msg}) \rightarrow \mathsf{CT}_{\mathbb{A}}}$. Remind that $\{\Psi_{\pi(i)}\}_{i\in[m]} = \{\langle n_{\pi(i)}, v_{\pi(i)}\rangle\}_{i\in[m]}$. Pick $s_1, s' \xleftarrow{\$} \mathbb{Z}_p^*$ and vector $\mathbf{v} \xleftarrow{\$} \mathbb{Z}_p^{n-1}$. Compute ciphertext $\mathsf{ct}_0 = e(g_1, g_2)^{\alpha s_1} \cdot \mathsf{msg}$, $\mathsf{ct}_1 = g_2^{s_1}$, $\mathsf{ct}_2 = g_2^{s'}$, $\mathsf{ct}_{3,i} = h^{\mathbf{M}_i(s_1||\mathbf{v})^\top} \cdot H(\Psi_{\pi(i)})^{s'}$.
Output $\mathsf{CT} := ((\mathbf{M}, \pi, \{n_{\pi(i)}\}_{i\in[m]}), \mathsf{ct}_0, \mathsf{ct}_1, \mathsf{ct}_2, \{\mathsf{ct}_{3,i}\}_{i\in[m]})$.
$\underline{\mathsf{Dec}(\mathsf{SK}_{\mathcal{S}}, \mathsf{CT}_{\mathbb{A}}) \rightarrow \mathsf{msg}/\perp}$. If there is any subset I that matches the attribute names $\{n_i\}_{i\in[\ell]}$ in SK with $(\mathbf{M}, \pi, \{n_{\pi(i)}\}_{i\in[m]})$ in CT, there exist constants $\{\gamma_i\}_{i\in I}$ s.t. $\sum_{i\in I} \gamma_i \mathbf{M}_i = (1, 0, \cdots, 0)$. Output

$$\mathsf{msg} = \frac{\mathsf{ct}_0 \cdot e(\prod_{i\in I}(\mathsf{ct}_{3,i})^{\gamma_i}, \mathsf{sk}_3)}{e(\mathsf{sk}_1, \mathsf{ct}_1) \cdot e(\prod_{i\in I}(\mathsf{sk}_{2,\pi(i)})^{\gamma_i}, \mathsf{ct}_2)}.$$

Figure 3: A-CP-ABE Scheme

(2) Randomness Splitting Technique. To address the vulnerability of attribute guessing attacks in the FABEO KP-ABE scheme, we implement a *randomness splitting technique*. In the original FABEO KP-ABE scheme (see Fig. 1 in [21]), the reuse of a single random value s across ciphertext components $\mathsf{ct}_{1,u} = H(u)^s$ and $\mathsf{ct}_2 = g_2^s$ makes it possible for an attacker to deduce an attribute u by testing the equal-

Hybrid-ABE: Bridging CP-ABE and KP-ABE

$\underline{\mathsf{Setup}(1^\lambda) \to (\mathsf{mpk}, \mathsf{msk})}$. Generate $\mathcal{G} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$. Pick $x, \mu, b_1, b_2 \xleftarrow{\$} \mathbb{Z}_p^*$, $h \xleftarrow{\$} \mathbb{G}_1$ and a hash function $H : \{0,1\}^* \to \mathbb{G}_1$. Compute $Y = e(g_1, g_2)^{x\mu}$, $\delta_0 = g_2^{\mu}$, $\delta_1 = g_2^{b_1}$, $\delta_2 = g_2^{b_2}$. Output the master public key $\mathsf{mpk} := (\mathcal{G}, H, Y, h, \delta_0, \delta_1, \delta_2)$ and master secret key $\mathsf{msk} := (x, \mu, b_1, b_2)$.
$\underline{\mathsf{EKGen}(\mathsf{msk}, \mathcal{S}_{\mathsf{snd}} = \{u_i\}_{i\in[\ell]} = \{\langle n_i, v_i\rangle\}_{i\in[\ell]}) \to \mathsf{EK}_{\mathcal{S}_{\mathsf{snd}}}}$. Pick $\tau \xleftarrow{\$} \mathbb{Z}_p^*$. Compute $\mathsf{ek}_1 = g_1^x h^\tau$, $\mathsf{ek}_3 = \delta_1^\tau$, $\mathsf{ek}_4 = \delta_2^\tau$, $\mathsf{ek}_{2,i} = H(u_i)^\tau$. Output $\mathsf{EK}_{\mathcal{S}_{\mathsf{snd}}} := (\{n_i\}_{i\in[\ell]}, \mathsf{ek}_1, \{\mathsf{ek}_{2,i}\}_{i\in\ell}, \mathsf{ek}_3, \mathsf{ek}_4)$.

$\underline{\mathsf{PolGen}(\mathbb{A}_{\mathsf{rcv}} = (\mathsf{msk}, \mathbf{A}, \rho, \{\Psi_{\rho(i)}\}_{i\in[m]})) \to \mathsf{SK}_{\mathbb{A}_{\mathsf{rcv}}}}$. Pick $r' \xleftarrow{\$} \mathbb{Z}_p^*$ and $\mathbf{y} \xleftarrow{\$} \mathbb{Z}_p^{n-1}$. Compute $\mathsf{sk}_1 = g_2^{r'}$, $\mathsf{sk}_{2,i} = \left(h^{\mathbf{M}_i(s_1||\mathbf{y})^\top} \cdot H(\Psi_{\rho(i)})^{r'}\right)^{\frac{1}{b_1}}$, $\mathsf{sk}_{3,i} = \left(h^{\mathbf{M}_i(s_1||\mathbf{y})^\top} \cdot H(\Psi_{\rho(i)})^{r'}\right)^{\frac{1}{b_2}}$. Output $\mathsf{SK}_{\mathbb{A}_{\mathsf{rcv}}} := ((\mathbf{A}, \rho, \{n_{\rho(i)}\}_{i\in[m]}), \mathsf{sk}_1, \{\mathsf{sk}_{2,i}, \mathsf{sk}_{3,i}\}_{i\in[m]})$.
$\underline{\mathsf{Enc}(\mathsf{EK}_{\mathcal{S}_{\mathsf{snd}}}, \mathsf{msg}) \to \mathsf{CT}_{\mathsf{snd}}}$. Pick $\tau', s', s'' \xleftarrow{\$} \mathbb{Z}_p^*$. Let $s = s' + s''$. Compute $\mathsf{ct}_0 = Y^s \cdot \mathsf{msg}$, $\mathsf{ct}_{1,i} = (\mathsf{ek}_{1,i} \cdot H(u_i)^{\tau'})^s$, $\mathsf{ct}_2 = (\mathsf{ek}_2 \cdot \delta_1^{\tau'})^{s'}$, $\mathsf{ct}_3 = (\mathsf{ek}_2 \cdot \delta_2^{\tau'})^{s''}$, $\mathsf{ct}_4 = (\mathsf{ek}_4 \cdot h^{\tau'})^s$. Output $\mathsf{CT}_{\mathsf{snd}} := (\{n_i\}_{i\in[\ell]}, \mathsf{ct}_0, \{\mathsf{ct}_{1,i}\}_{i\in[\ell]}, \mathsf{ct}_2, \mathsf{ct}_3, \mathsf{ct}_4)$.
$\underline{\mathsf{Dec}(\mathsf{SK}_{\mathbb{A}_{\mathsf{rcv}}}, \mathsf{CT}_{\mathsf{snd}}) \to \mathsf{msg}/\bot}$. If there is any subset I that matches $\{n_i\}_{i\in[\ell]}$ in $\mathsf{CT}_{\mathsf{snd}}$ with $(\mathbf{A}, \rho, \{n_{\rho(i)}\}_{i\in[m_2]})$ in $\mathsf{SK}_{\mathbb{A}_{\mathsf{rcv}}}$, there exist constants $\{\omega_i\}_{i\in I}$ s.t. $\sum_{i\in I} \omega_i \mathbf{A}_i = (1, 0, \cdots, 0)$. Output

$$\mathsf{msg} = \mathsf{ct}_0 \cdot \frac{e(\prod_{i\in I}(\mathsf{sk}_{2,i})^{\omega_i}, \mathsf{ct}_2)e(\prod_{i\in I}(\mathsf{sk}_{3,i})^{\omega_i}, \mathsf{ct}_3)}{e(\mathsf{ct}_4, \delta_0)e(\prod_{i\in I}(\mathsf{ct}_{1,\rho(i)})^{\omega_i}, \mathsf{sk}_1)}.$$

Figure 4: Hybrid-ABE Scheme

ity $e(\mathsf{ct}_{1,u}, g_2) = e(H(u), \mathsf{ct}_2)$. To mitigate this risk, our A-KP-ABE scheme (see Fig. 2) splits the randomness s into two independent values, s' and s'', such that $s = s' + s''$. This adjustment modifies the ciphertext components as follows: $\mathsf{ct}_{1,i} = H(u_i)^s$, $\mathsf{ct}_2 = \delta_1^{s'}$, and $\mathsf{ct}_3 = \delta_2^{s''}$, where $\delta_1 = g_2^{b_1}$ and $\delta_2 = g_2^{b_2}$. To cancel the exponentiation b_1 and b_2, the decryption key includes components $\mathsf{sk}_{2,i}$ and $\mathsf{sk}_{3,i}$, which use exponentiation by $\frac{1}{b_1}$ and $\frac{1}{b_2}$. These modifications ensure that the attribute set remains concealed, and the modified components $\mathsf{ct}_{1,i}$, ct_2, and ct_3 reveal no information for any attacker to infer attributes, thus effectively preventing attribute guessing attacks.

(3) Scalability and Efficiency Enhancement. We take several steps to make FEME highly scalable and efficient.

First, we construct large-universe A-KP-ABE and A-CP-ABE schemes that allow FEME to handle an extensive and potentially unbounded set of attributes dynamically. This approach reduces the need for pre-defining and managing fixed attribute sets, enhancing scalability and minimizing the overhead associated with system updates and attribute expansions. By replacing the term $H(|\mathcal{U}| + 1)$ in FABEO with an element $h \xleftarrow{\$} \mathbb{G}_1$ in the master public key mpk, A-KP-ABE and A-CP-ABE become independent of the size of the attribute universe $|\mathcal{U}|$. This modification eliminates dependence on a fixed set of attributes, thereby improving both scalability and efficiency, and allowing for a more flexible and adaptable system.

Second, we address the main efficiency bottleneck in FABEO CP-ABE, which stems from its ciphertext components. Specifically, in FABEO CP-ABE, the ciphertext components $\mathsf{ct}_{2,j} = g_2^{s'[j]}$ and $\mathsf{ct}_{3,i} = H(|\mathcal{U}| + 1)^{\mathbf{M}_i(s_1||\mathbf{v})^\top} \cdot H(\Psi_{\pi(i)})^{s'[\zeta(i)]}$ involve a random vector $\vec{s'} \xleftarrow{\$} \mathbb{Z}_p^\tau$, where τ represents vector size and $\zeta(i) := |\{z|\pi(z) = \pi(i), z \leqslant i\}|$. This setup leads to decryption involving τ pairing operations and approximately τI exponentiations, which becomes computationally expensive, where I represents the number of attributes required to satisfy an access policy. To mitigate this, we propose modifications to the ciphertext components, simplifying them to $\mathsf{ct}_2 = g_2^{s'}$ and $\mathsf{ct}_{3,i} = h^{\mathbf{M}_i(s_1||\mathbf{v})^\top} \cdot H(\Psi_{\pi(i)})^{s'}$ in our developed A-CP-ABE (Fig. 3), where $s' \xleftarrow{\$} \mathbb{Z}_p$. This significantly reduces the decryption workload to a single pairing operation and I exponentiations, resulting in a more efficient decryption.

Third, recognizing the dual structure between FABEO KP-ABE and FABEO CP-ABE, we apply the above optimization technique to A-KP-ABE (Fig. 2). We replace the secret key components $\mathsf{sk}_{1,j} = g_2^{r'[j]}$, $\mathsf{sk}_{2,i} = g_1^{\mathbf{A}_i(\alpha||\mathbf{y})^\top} \cdot H(\Psi_{\rho(i)})^{r'[\eta(i)]}$ in FABEO KP-ABE, where $\vec{r'} \xleftarrow{\$} \mathbb{Z}_p^\tau$, with $\mathsf{sk}_1 = g_2^{r'}$ and $\mathsf{sk}_{2,i} = g_1^{\mathbf{A}_i(\alpha||\mathbf{y})^\top} \cdot H(\Psi_{\rho(i)})^{r'}$, using $r' \xleftarrow{\$} \mathbb{Z}_p$ in our developed A-KP-ABE. Here, $\eta(i)$ is defined as $|\{z|\rho(z) = \rho(i), z \leqslant i\}|$. Then, we utilize the *randomness splitting technique* to split $\mathsf{sk}_{2,i}$ into two terms $\mathsf{sk}_{2,i} = \left(g_1^{\mathbf{A}_i(\alpha||\mathbf{y})^\top} \cdot H(\Psi_{\rho(i)})^{r'}\right)^{\frac{1}{b_1}}$, $\mathsf{sk}_{3,i} = \left(g_1^{\mathbf{A}_i(\alpha||\mathbf{y})^\top} \cdot H(\Psi_{\rho(i)})^{r'}\right)^{\frac{1}{b_2}}$. These optimizations effectively reduce the computational overhead in both A-CP-ABE and A-KP-ABE, making them more suitable for real-world applications, particularly those in resource-constrained environments.

Stage 2. To address the gap between ME and A-CP-ABE/A-KP-ABE, we develop a Hybrid-ABE scheme as shown in Fig. 4. This gap stems from the *sender authentication* requirement in ME, which ensures that only authorized senders (with a valid encryption key EK tied to their attributes) can generate legitimate ciphertexts. However, FABEO CP-ABE and KP-ABE do not inherently support sender authentication, as senders only use the master public key mpk and an attribute set (in KP-ABE) or access policy (in CP-ABE) to derive ciphertext.

To close this gap, Hybrid-ABE integrates the frameworks of A-CP-ABE and A-KP-ABE. Hybrid-ABE comprises the following algorithms: Setup, EKGen, PolGen, Enc and Dec. The EKGen algorithm generates the sender's attribute encryption key $\mathsf{EK}_{\mathcal{S}_{\mathsf{snd}}}$, drawing from A-CP-ABE's KeyGen algorithm and incorporating the *randomness splitting technique* from A-KP-ABE's Enc algorithm. PolGen produces the receiver's policy decryption key $\mathsf{SK}_{\mathbb{A}_{\mathsf{rcv}}}$, utilizing the exponentiation tricks from A-KP-ABE's KeyGen.

During encryption, the sender's encryption key $\mathsf{EK}_{\mathcal{S}_{\mathsf{snd}}}$ is used to generate ciphertext $\mathsf{CT}_{\mathsf{snd}}$, which wraps message msg in ciphertext component $\mathsf{ct}_0 = Y^s \cdot \mathsf{msg}$. The sender's encryption key is re-randomized into ciphertext components $\mathsf{ct}_{1,i}$, ct_2, ct_3, ct_4, using a nonce $\tau' \xleftarrow{\$} \mathbb{Z}_p^*$ and two split randomness values s' and s'' (where $s = s' + s''$). The decryption algorithm unifies the processes of A-CP-ABE and A-KP-ABE, requiring only 4 pairings and $3I$ exponentiations (where I represents the number of attributes required to satisfy an access policy), ensuring high efficiency.

FEME: Fast and Expressive Matchmaking Encryption

1. $\mathsf{Setup}(1^\lambda) \rightarrow (\mathsf{mpk}, \mathsf{msk})$. // System Setup

This algorithm takes in the security parameter 1^λ and generates a bilinear pairing $\mathcal{G} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$. The algorithm picks random numbers $\alpha, x, \mu, b_1, b_2 \xleftarrow{\$} \mathbb{Z}_p^*$, $h \xleftarrow{\$} \mathbb{G}_1$, hash functions $H : \{0,1\}^* \rightarrow \mathbb{G}_1$, $\hat{H} : \mathbb{G}_T \rightarrow \{0,1\}^{l_0}$, and a polynomial-time computable padding function $\phi : \{0,1\}^n \rightarrow \{0,1\}^{\ell_0}$. It computes $Z = e(g_1, g_2)^\alpha, Y = e(g_1, g_2)^{x\mu}, \delta_0 = g_2^\mu, \delta_1 = g_2^{b_1}, \delta_2 = g_2^{b_2}$. It outputs the master public key as $\mathsf{mpk} := (\mathcal{G}, H, \hat{H}, \phi, Z, Y, h, \delta_0, \delta_1, \delta_2)$, and the master secret key as $\mathsf{msk} := (\alpha, x, \mu, b_1, b_2)$.

2. $\mathsf{EKGen}(\mathsf{msk}, \mathcal{S}_{\mathsf{snd}} = \{u_i\}_{i\in[\ell_1]} = \{\langle n_i, v_i\rangle\}_{i\in[\ell_1]}) \rightarrow \mathsf{EK}_{\mathcal{S}_{\mathsf{snd}}}$. // Attribute Encryption Key Generation

This algorithm generates the sender's attribute encryption key $\mathsf{EK}_{\mathcal{S}_{\mathsf{snd}}}$ for attributes $\mathcal{S}_{\mathsf{snd}} = \{u_i\}_{i\in[\ell_1]} = \{\langle n_i, v_i\rangle\}_{i\in[\ell_1]}$, where n_i denotes the attribute name and v_i the attribute value. It picks a random number $\tau \xleftarrow{\$} \mathbb{Z}_p^*$ and computes as follows:

$$\mathsf{ek}_{1,i} = H(u_i)^\tau \text{ for } i \in [\ell_1], \quad \mathsf{ek}_2 = \delta_1^\tau, \quad \mathsf{ek}_3 = \delta_2^\tau, \quad \mathsf{ek}_4 = g_1^x h^\tau.$$

It outputs the sender attribute encryption key $\mathsf{EK}_{\mathcal{S}_{\mathsf{snd}}} := (\{n_i\}_{i\in[\ell_1]}, \{\mathsf{ek}_{1,i}\}_{i\in[\ell_1]}, \mathsf{ek}_2, \mathsf{ek}_3, \mathsf{ek}_4)$.

3. $\mathsf{DKGen}(\mathsf{msk}, \mathcal{S}_{\mathsf{rcv}} = \{u_i\}_{i\in[\ell_2]} = \{\langle n_i, v_i\rangle\}_{i\in[\ell_2]}) \rightarrow \mathsf{DK}_{\mathcal{S}_{\mathsf{rcv}}}$. // Attribute Decryption Key Generation

To generate the receiver's attribute decryption key $\mathsf{DK}_{\mathcal{S}_{\mathsf{rcv}}}$ for attributes $\mathcal{S}_{\mathsf{rcv}} = \{u_i\}_{i\in[\ell_2]} = \{\langle n_i, v_i\rangle\}_{i\in[\ell_2]}$, this algorithm picks a random number $r \xleftarrow{\$} \mathbb{Z}_p^*$ and computes as follows:

$$\mathsf{dk}_1 = g_1^\alpha h^r, \quad \mathsf{dk}_{2,i} = H(u_i)^r, \quad \mathsf{dk}_3 = g_2^r.$$

It outputs the receiver attribute decryption key $\mathsf{DK}_{\mathcal{S}_{\mathsf{rcv}}} := (\{n_i\}_{i\in[\ell_2]}, \mathsf{dk}_1, \{\mathsf{dk}_{2,i}\}_{i\in[\ell_2]}, \mathsf{dk}_3)$.

4. $\mathsf{PolGen}(\mathsf{msk}, \mathbb{A}_{\mathsf{rcv}} = (\mathbf{A}, \rho, \{\Psi_{\rho(i)}\}_{i\in[m_2]})) \rightarrow \mathsf{SK}_{\mathbb{A}_{\mathsf{rcv}}}$. // Policy Decryption Key Generation

This receiver's policy decryption key generation algorithm generates the secret key $\mathsf{sk}_{\mathbb{A}_{\mathsf{rcv}}}$ with receiver's monotone span policy $\mathbb{A}_{\mathsf{rcv}} = (\mathbf{A}, \rho, \{\Psi_{\rho(i)}\}_{i\in[m_2]})$, where $\mathbf{A}$ is an $m_2 \times n_2$ access control matrix, $\{\Psi_{\rho(i)}\}_{i\in[m_2]} = \{\langle n_{\rho(i)}, v_{\rho(i)}\rangle\}_{i\in[m_2]}$, $n_{\rho(i)}$ denotes attribute name and $v_{\rho(i)}$ attribute value. It picks a random number $r' \xleftarrow{\$} \mathbb{Z}_p^*$, a random vector $\mathbf{y} \xleftarrow{\$} \mathbb{Z}_p^{n_2-1}$ and computes as follows:

$$\mathsf{sk}_1 = g_2^{r'}, \quad \mathsf{sk}_{2,i} = (g_1^{\mathbf{A}_i(\alpha||\mathbf{y})^\top} \cdot H(\Psi_{\rho(i)})^{r'})^{\frac{1}{b_1}}, \quad \mathsf{sk}_{3,i} = (g_1^{\mathbf{A}_i(\alpha||\mathbf{y})^\top} \cdot H(\Psi_{\rho(i)})^{r'})^{\frac{1}{b_2}},$$

$$\mathsf{sk}_{4,i} = (h^{\mathbf{A}_i(\mu||\mathbf{y})^\top} \cdot H(\Psi_{\rho(i)})^{r'})^{\frac{1}{b_1}}, \quad \mathsf{sk}_{5,i} = (h^{\mathbf{A}_i(\mu||\mathbf{y})^\top} \cdot H(\Psi_{\rho(i)})^{r'})^{\frac{1}{b_2}}, \text{ for each row } i \in [m_2].$$

It outputs the receiver policy decryption key $\mathsf{SK}_{\mathbb{A}_{\mathsf{rcv}}} := ((\mathbf{A}, \rho, \{n_{\rho(i)}\}_{i\in[m_2]}), \mathsf{sk}_1, \{\mathsf{sk}_{2,i}, \mathsf{sk}_{3,i}, \mathsf{sk}_{4,i}, \mathsf{sk}_{5,i}\}_{i\in[m_2]})$.

5. $\mathsf{Enc}(\mathsf{EK}_{\mathcal{S}_{\mathsf{snd}}}, \mathbb{A}_{\mathsf{snd}} = (\mathbf{M}, \pi, \{\Psi_{\pi(i)}\}_{i\in[m_1]}), \mathsf{msg}) \rightarrow \mathsf{CT}_{\mathsf{snd}}$. // Encrypt

This algorithm encrypts a message $\mathsf{msg} \in \{0,1\}^n$ with sender's monotone span policy $\mathbb{A}_{\mathsf{snd}} = (\mathbf{M}, \pi, \{\Psi_{\pi(i)}\}_{i\in[m_1]})$ and sender attribute encryption key $\mathsf{EK}_{\mathcal{S}_{\mathsf{snd}}}$, where $\{\Psi_{\pi(i)}\}_{i\in[m_1]} = \{\langle n_{\pi(i)}, v_{\pi(i)}\rangle\}_{i\in[m_1]}$ and matrix $\mathbf{M} \in \mathbb{Z}^{m_1\times n_1}$. It selects $s_1, s_2', s_2'', s_3', s_3'', \tau' \xleftarrow{\$} \mathbb{Z}_p^*$, a vector $\mathbf{v} \xleftarrow{\$} \mathbb{Z}_p^{n_1-1}$. Let $s_2 = s_2' + s_2''$ and $s_3 = s_3' + s_3''$. It computes as follows:

$$V = Z^{s_1+s_2} \cdot Y^{s_3}, \quad \mathsf{ct}_0 = \phi(\mathsf{msg}) \oplus \hat{H}(V), \quad \mathsf{ct}_1 = g_2^{s_1}, \quad \mathsf{ct}_2 = g_2^{s_3},$$

$$\mathsf{ct}_{3,i} = h^{\mathbf{M}_i(s_1||\mathbf{v})^\top} \cdot H(\Psi_{\pi(i)})^{s_3} \text{ for each row } i \in [m_1], \quad \mathsf{ct}_{4,1} = \delta_1^{s_2'}, \quad \mathsf{ct}_{4,2} = \delta_2^{s_2''},$$

$$\mathsf{ct}_{5,i} = H(u_i)^{s_2}, \quad \mathsf{ct}_{6,i} = (\mathsf{ek}_{1,i} \cdot H(u_i)^{\tau'})^{s_3} \text{ for } i \in [\ell_1], \quad \mathsf{ct}_7 = (\mathsf{ek}_2 \cdot \delta_1^{\tau'})^{s_3'}, \quad \mathsf{ct}_8 = (\mathsf{ek}_3 \cdot \delta_2^{\tau'})^{s_3''}, \quad \mathsf{ct}_9 = (\mathsf{ek}_4 \cdot h^{\tau'})^{s_3}.$$

It outputs the ciphertext

$$\mathsf{CT}_{\mathsf{snd}} := ((\mathbf{M}, \pi, \{n_{\pi(i)}\}_{i\in[m_1]}), \{n_i\}_{i\in[\ell_1]}, \mathsf{ct}_0, \mathsf{ct}_1, \mathsf{ct}_2, \{\mathsf{ct}_{3,i}\}_{i\in[m_1]}, \mathsf{ct}_{4,1}, \mathsf{ct}_{4,2}, \{\mathsf{ct}_{5,i}, \mathsf{ct}_{6,i}\}_{i\in[\ell_1]}, \mathsf{ct}_7, \mathsf{ct}_8, \mathsf{ct}_9).$$

6. $\mathsf{Dec}(\mathsf{DK}_{\mathcal{S}_{\mathsf{rcv}}}, \mathsf{SK}_{\mathbb{A}_{\mathsf{rcv}}}, \mathsf{CT}_{\mathsf{snd}}) \rightarrow \mathsf{msg}/\perp$: // Decrypt

This algorithm decrypts a given ciphertext $\mathsf{CT}_{\mathsf{snd}}$ using $\mathsf{DK}_{\mathcal{S}_{\mathsf{rcv}}}$ and $\mathsf{SK}_{\mathbb{A}_{\mathsf{rcv}}}$. If $\mathcal{S}_{\mathsf{rcv}} \models \mathbb{A}_{\mathsf{snd}}$ (denoting that $\mathcal{S}_{\mathsf{rcv}}$ satisfies $\mathbb{A}_{\mathsf{snd}}$), there exist constants $\{\gamma_i\}_{i\in I_1}$ s.t. $\sum_{i\in I_1} \gamma_i \mathbf{M}_i = (1, 0, \cdots, 0)$. If $\mathcal{S}_{\mathsf{snd}} \models \mathbb{A}_{\mathsf{rcv}}$ (denoting that $\mathcal{S}_{\mathsf{snd}}$ satisfies $\mathbb{A}_{\mathsf{rcv}}$), there exist constants $\{\omega_i\}_{i\in I_2}$ s.t. $\sum_{i\in I_2} \omega_i \mathbf{A}_i = (1, 0, \cdots, 0)$. This algorithm recovers V by computing

$$V = \frac{e(\mathsf{dk}_1, \mathsf{ct}_1)e(\prod_{i\in I_1}(\mathsf{dk}_{2,\pi(i)})^{\gamma_i}, \mathsf{ct}_2)}{e(\prod_{i\in I_1}(\mathsf{ct}_{3,\pi(i)})^{\gamma_i}, \mathsf{dk}_3)} \cdot \frac{e(\prod_{i\in I_2}(\mathsf{sk}_{2,\rho(i)})^{\omega_i}, \mathsf{ct}_{4,1})e(\prod_{i\in I_2}(\mathsf{sk}_{3,\rho(i)})^{\omega_i}, \mathsf{ct}_{4,2})}{e(\prod_{i\in I_2}(\mathsf{ct}_{5,\rho(i)})^{\omega_i}, \mathsf{sk}_1)}$$

$$\cdot \frac{e(\mathsf{ct}_9, \delta_0)e(\prod_{i\in I_2}(\mathsf{ct}_{6,\rho(i)})^{\omega_i}, \mathsf{sk}_1)}{e(\prod_{i\in I_2}(\mathsf{sk}_{4,\rho(i)})^{\omega_i}, \mathsf{ct}_7)e(\prod_{i\in I_2}(\mathsf{sk}_{5,\rho(i)})^{\omega_i}, \mathsf{ct}_8)}.$$

It computes $\phi(\mathsf{msg}) = \mathsf{ct}_0 \oplus \hat{H}(V)$. If the padding is valid, this algorithm returns msg. Otherwise, it returns $\perp$.

Figure 5: FEME: Fast and Expressive Matchmaking Encryption Scheme

Stage 3. We construct FEME as shown in Fig. 5 based on A-CP-ABE, A-KP-ABE, and Hybrid-ABE developed in Stages 1 and 2. FEME consists of the following algorithms: Setup, EKGen, DKGen, PolGen, Enc and Dec. The Setup algorithm initializes the master public key and the master secret key. EKGen generates the sender's attribute encryption key following the process outlined in Hybrid-ABE. DKGen produces the receiver's attribute decryption key, as KeyGen does from A-CP-ABE. Meanwhile, PolGen generates the receiver's policy decryption key following the KeyGen method from A-KP-ABE for producing its components $(\mathsf{sk}_1, \{\mathsf{sk}_{2,i}, \mathsf{sk}_{3,i}\}_{i\in[m_2]})$. Additionally, PolGen adopts the Hybrid-ABE framework for producing its components $(\{\mathsf{sk}_{4,i}, \mathsf{sk}_{5,i}\}_{i\in[m_2]})$, where m_2 denotes the number of rows in a receiver's access matrix $\mathbf{A}$.

During encryption, a message msg is encapsulated in ciphertext component $\mathsf{ct}_0 = \phi(\mathsf{msg}) \oplus \hat{H}(V)$ with $V = Z^{s_1+s_2} \cdot Y^{s_3}$, which combines elements from the ct_0 components of all three schemes A-CP-ABE, A-KP-ABE, and Hybrid-ABE, where $\hat{H}$ is a hash function, and ϕ is a polynomial-time computable and efficiently invertible padding function to realize authenticated encryption. The encryption process generates ciphertext components $(\mathsf{ct}_1, \mathsf{ct}_2, \{\mathsf{ct}_{3,i}\}_{i\in[m_2]})$ based on Enc in A-CP-ABE, $(\mathsf{ct}_{4,1}, \mathsf{ct}_{4,2}, \{\mathsf{ct}_{5,i}\}_{i\in[m_1]})$ from A-KP-ABE, and $(\{\mathsf{ct}_{6,i}\}_{i\in[\ell_1]}, \mathsf{ct}_7, \mathsf{ct}_8, \mathsf{ct}_9)$ based on Enc in Hybrid-ABE. Here, m_1 represents the number of rows in sender's access matrix $\mathbf{M}$, and ℓ_1 represents the number of attributes in sender's attribute set.

The decryption of FEME incorporates the decryption processes of A-CP-ABE, A-KP-ABE, and Hybrid-ABE, with its three decryption fractions corresponding to the Dec algorithms in each scheme.

4.3. FEME Construction

Following the above technical details, we describe the construction of FEME in Fig. 5. Its syntax definition is presented in App. A.1, and correctness proof in App. B.

In FEME, a key generation center (KGC) runs Setup to generate the master public and secret keys for the system. To enable secure communication, the KGC executes EKGen to create the sender's attribute encryption key, and DKGen/PolGen to generate the receiver's attribute decryption key and policy decryption key. During Enc, the sender specifies an access policy that the receiver must meet to access the message. FEME ensures that decryption is only possible if the sender's and receiver's attributes match their respective policies, guaranteeing *sender authenticity* by certifying sender attributes through the attribute encryption key to prevent forged ciphertexts.

The Setup phase in FEME includes a padding function ϕ that ensures authenticated message encryption. It should be efficiently computable and invertible, allowing for polynomial-time verification of correct padding [7]. This guarantees that all encrypted messages can be checked for integrity, preventing unauthorized modifications and enhancing the overall security of the encryption process.

FEME's partially hidden access structure plays a crucial role in optimizing the decryption process. By exposing only attribute names while keeping attribute values hidden, the receiver can initially filter out unmatched ciphertexts without performing costly computations. In the Dec algorithm, the receiver first verifies if the sender's attribute name set satisfies its access policy and checks if its own attribute names meet the sender's policy. These preliminary checks are done without computation, streamlining the initial matching phase. If either condition is not met, decryption is aborted, and $\perp$ is outputted to indicate failure. When both checks pass, the receiver proceeds to full decryption, performing the necessary calculations outlined in Dec to verify the attribute values. The message is returned only if both policies are fully satisfied and the padding verification confirms validity. If the padding fails, the decryption outputs $\perp$, ensuring that only correctly formatted messages are accepted and preserving the integrity of the process.

Security Model. The security model for FEME, detailed in Appendix A.2, outlines its confidentiality, anonymity, and authenticity properties. *Confidentiality* ensures that no probabilistic polynomial-time (PPT) adversary can distinguish between two challenge messages encrypted under a target attribute set and policy, even with access to all the key generation oracles, with a restriction that the decryption keys for the target attribute set and policy have not been queried. *Anonymity* guarantees that no PPT adversary, who outputs two target attribute sets and policies, can distinguish which attribute set or policy was used by the challenger to create a ciphertext, even with access to all the key generation oracles, with a restriction that the decryption keys for the two target attribute sets and policies have not been queried. *Authenticity* ensures that an adversary cannot forge a valid ciphertext capable of passing decryption without possessing an attribute encryption key with attributes that satisfy the target access policy, even with access to all the key generation oracles.

Theorem 1. *FEME satisfies confidentiality under the Generic Group Model (GGM) by modeling the hash function H as a random oracle.*

Theorem 2. *FEME satisfies anonymity under GGM by modeling the hash function H as a random oracle.*

Theorem 3. *FEME satisfies authenticity under GGM by modeling the hash function H as a random oracle.*

The proofs of Theorems 1-3 are deferred to Appendices C.1-C.3, respectively.

4.4. Comparative Advantages of FEME

Table 1 compares the existing ME schemes in terms of expressiveness, security and privacy, and usability.

In terms of expressiveness, FEME and ACME [32] support monotonic Boolean formula-based access structures, whereas other schemes focus on identity-based matching (either single, multiple, or fuzzy identities). However, ACME's small-universe design constrains the system to a fixed set of

Scheme	**Expressiveness**			**Security and Privacy**			**Usability**	
	Monotonic Policy	Arbitrary Attribute	Large Universe	Data Privacy	Data Authenticity	Attribute Privacy	No Pre-registration Pairing	No Additional Component
IBME [7] (Crypto'19)	$\times$	$\times$	$\surd$	$\surd$	$\surd$	$\surd$	$\times$	$\surd$
IBME [16] (IndoCrypt'21)	$\times$	$\times$	$\surd$	$\surd$	$\times/\surd$	$\surd$	$\times$	$\surd$
IBME [13] (AsiaCrypt'21)	$\times$	$\times$	$\surd$	$\surd$	$\surd$	$\surd$	$\times$	$\surd$
FBME [29] (TIFS'23)	$\times$	$\times$	$\times$	$\surd$	$\surd$	$\surd$	$\times$	$\surd$
PSME [26] (TIFS'23)	$\times$	$\times$	$\times$	$\surd$	$\surd$	$\surd$	$\times$	$\surd$
CLME [31] (TIFS'23)	$\times$	$\times$	$\surd$	$\surd$	$\surd$	$\surd$	$\times$	$\surd$
ACME [32] (NDSS'24)	$\surd$	$\times$	$\times$	$\surd$	$\surd$	$\times$	$\surd$	$\times$
FEME	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$

TABLE 1: Comparison of Matchmaking Encryption (ME) Schemes

attributes, necessitating a complete rebuild when expanding the attribute set. Furthermore, ACME uses binary attribute vectors (where each element is either 1 or 0) to denote attribute sets, necessitating larger vectors to represent a wide range of attributes, which in turn significantly increases the computational cost during encryption and decryption. In contrast, FEME supports an unrestricted attribute universe, enabling any arbitrary string to function as an attribute.

In terms of security and privacy, all ME schemes ensure data privacy. Regarding data authenticity, Francati et al. [16] proposed one IBME scheme without authenticity and another IBME scheme with authenticity using NIZK. Regarding attribute privacy, ACME exposes the attributes in the outer layer as it employs a dual-layer matching mechanism. Except for ACME, all other ME schemes in comparison preserve attribute/identity privacy.

In terms of usability, IBME [7], [13], [16], FBME [29], PSME [26], and CLME [31] require pre-registration pairing, in that the sender must know the receiver's identity before encryption. ACME [32] relies on anonymous credentials as an additional component for sender authentication, adding overhead due to credential issuance, management, and revocation. In contrast, FEME avoids pre-registration pairing and additional components for sender authentication, achieving higher usability among the ME schemes.

In summary, FEME stands out as the only ME scheme that achieves expressive bilateral access control, robust security and privacy, and advantageous usability.

5. PriSrv+ Protocol

PriSrv+ is a private service discovery protocol that leverages FEME to enable privacy-preserving service broadcasts and mutual authentication between a service provider and a client. Building on its predecessor PriSrv [32], PriSrv+ replaces the core ACME scheme with FEME to eliminate the need for issuing, managing, and revoking credentials.

The overall workflow of PriSrv+ is as follows. (1) During the system setup phase, a KGC generates the attribute encryption key, attribute decryption key, and policy decryption key for the service provider (S) and the client (C) according to the FEME scheme. Both parties require a complete set of encryption and decryption keys as they act as both sender and receiver during interactions. (2) During the broadcast phase, the service provider announces an encrypted broadcast message using FEME. This message includes a service policy (in the partially hidden structure), service details, the provider's Diffie-Hellman (DH) public key, and a MAC key for authentication. Consider a private file-sharing service in a corporate Wi-Fi network as an example, where the provider's policy is "(Device Type: Laptop **AND** Network: Corporate) **OR** (Location: Main Office **AND** Security Level: High)" which is transformed to its partially hidden form "(Device Type **AND** Network) **OR** (Location **AND** Security Level)" in the broadcast.

(3) In the service discovery phase, the client checks if its attribute names match the service policy and if the provider's attribute names (included in the broadcast message) meet its own policy. For the same example, the client may set its connection policy as "(Device Type: Server **AND** Network: Corporate) **OR** Security Level: High". If both checks pass, the client executes the FEME decrypt algorithm to verify whether the hidden attribute values of both parties satisfy those in each other's policies. If it succeeds, the client generates a response, and sends a FEME encrypted reply with its policy in its partially hidden form (which is "(Device Type **AND** Network) **OR** Security Level" in the above example) and an authentication tag, including its DH public key and MAC key, back to the provider. (4) The service provider decrypts and verifies the client's response, then sends a confirmation message with an authentication tag to the client. (5) Both parties independently compute a shared session key using their respective DH secret keys, ensuring mutual authentication and maintaining privacy for both the client and the provider.

5.1. PriSrv+ Construction

Fig. 6 illustrates the PriSrv+ protocol, which operates through two phases: a privacy-preserving service broadcast phase and an anonymous mutual authentication phase. The differences between PriSrv+ and PriSrv, highlighted in blue, are as follows: (1) PriSrv+ employs the sender's encryption key, EK, for sender authentication, whereas PriSrv uses anonymous credentials for this purpose. (2) PriSrv+ utilizes FEME to achieve bilateral policy control, while PriSrv relies on ACME for this functionality.

A unique broadcast identifier (bid) is assigned to each broadcast cycle, and a session identifier (sid) is assigned to each session. The broadcast cycle has

Privacy-preserving Service Broadcast Phase

Service Provider S's Broadcast: $bid, \mathsf{CT_B} \leftarrow \mathcal{FEME}.\mathsf{Enc}(\mathsf{EK}_{\mathcal{S}_s}, \mathbb{A}_s, \mathsf{MSG_B})$

where $\mathsf{MSG_B} = (bid||Z||Service_{Type}||Service_{Par}||K_c)$, $z \xleftarrow{\$} \mathbb{Z}_p^*$, $Z \leftarrow h^z \in G_2$, $K_c \leftarrow \mathcal{MAC}.\mathsf{KeyGen}(1^\lambda)$

Anonymous Mutual Authentication Phase

Client (C)		**Service Provider** (S)
$(\mathsf{EK}_{\mathcal{S}_c}, \mathsf{DK}_{\mathcal{S}_c}, \mathsf{SK}_{\mathbb{A}_c})$		$(\mathsf{EK}_{\mathcal{S}_s}, \mathsf{DK}_{\mathcal{S}_s}, \mathsf{SK}_{\mathbb{A}_s})$
$\mathsf{MSG_B} \leftarrow \mathcal{FEME}.\mathsf{Dec}(\mathsf{DK}_{\mathcal{S}_c}, \mathsf{SK}_{\mathbb{A}_c}, \mathsf{CT_B})$		
$x_1, x_2 \xleftarrow{\$} \mathbb{Z}_p^*$, $X_1 \leftarrow g^{x_1} \in G_1$, $X_2 \leftarrow h^{x_2} \in G_2$		
$\sigma_c \leftarrow \mathcal{MAC}.\mathsf{MAC}(K_c, M_c)$		
where $M_c = (\text{“}C \rightarrow S\text{”}, bid, sid, X_1, X_2, Z)$		
$K_s \leftarrow \mathcal{MAC}.\mathsf{KeyGen}(1^\lambda)$	$\xrightarrow{bid, sid, \sigma_c, \mathsf{CT}_c}$	$\mathsf{MSG}_c \leftarrow \mathcal{FEME}.\mathsf{Dec}(\mathsf{DK}_{\mathcal{S}_s}, \mathsf{SK}_{\mathbb{A}_s}, \mathsf{CT}_c)$
$\mathsf{CT}_c \leftarrow \mathcal{FEME}.\mathsf{Enc}(\mathsf{EK}_{\mathcal{S}_c}, \mathbb{A}_c, \mathsf{MSG}_c)$		$b_c \leftarrow \mathcal{MAC}.\mathsf{Verify}(K_c, M_c, \sigma_c)$
where $\mathsf{MSG}_c = (K_s, M_c)$		If $b_c = 0$, abort; otherwise,
		$y \xleftarrow{\$} \mathbb{Z}_p^*$, $Y \leftarrow g^y \in G_1$
$b_s \leftarrow \mathcal{MAC}.\mathsf{Verify}(K_s, M_s, \sigma_s)$		$\sigma_s \leftarrow \mathcal{MAC}.\mathsf{MAC}(K_s, M_s)$
If $b_s = 0$, abort; otherwise,	$\xleftarrow{M_s, \sigma_s}$	where $M_s = (\text{“}S \rightarrow C\text{”}, bid, sid, X_1, X_2, Y, Z)$
$SSK_{c,s} \leftarrow \bar{H}(Y^{x_1}, Z^{x_2})$		$SSK_{c,s} \leftarrow \bar{H}(X_1^y, X_2^z)$

Figure 6: PriSrv+ Protocol

a lifetime (e.g., 30 seconds), with the timestamp included in bid. Clients verify the timestamp upon decryption to ensure message freshness. Let $\mathcal{FEME} = (\mathsf{Setup}, \mathsf{EKGen}, \mathsf{DKGen}, \mathsf{PolGen}, \mathsf{Enc}, \mathsf{Dec})$ be a FEME scheme, $\mathcal{MAC} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{MAC}, \mathsf{Verify})$ be a message authentication code (MAC) scheme, and $\bar{H} : \{0,1\}^* \rightarrow \mathcal{K}$ be a hash function where $\mathcal{K}$ represents the secret session key space.

Service Broadcast Phase: To initiate a broadcast, S selects an access policy $\mathbb{A}_s$ that C should satisfy. S chooses an ephemeral Diffie-Hellman (DH) exponent $z \xleftarrow{\$} \mathbb{Z}_p^*$ and computes $Z = h^z$. It generates a MAC key $K_c \leftarrow \mathcal{MAC}.\mathsf{KeyGen}(1^\lambda)$. The broadcast message $\mathsf{MSG}_B = (bid||Z||Service_{Type}||Service_{Par}||K_c)$ includes the broadcast identifier, service type, parameters, and the MAC key. S encrypts it into a ciphertext CT_B using $\mathcal{FEME}.\mathsf{Enc}$, and broadcasts bid and CT_B publicly.

Anonymous Mutual Authentication Phase: This phase establishes a session key ($SSK_{c,s}$) between C and S.

(1) Client Response: C checks whether its attribute name set $\{n_i\}_{i\in[\ell_2]}$ satisfies S's policy $(\mathbf{M}, \pi, \{n_{\pi(i)}\}_{i\in[m_1]})$, and whether S's attribute name set $\{n_i\}_{i\in[\ell_1]}$ satisfies C's policy $(\mathbf{A}, \rho, \{n_{\rho(i)}\}_{i\in[m_2]})$. If either test fails, C discards the broadcast ciphertext without decryption. Otherwise, C decrypts CT_B using its decryption keys $(\mathsf{DK}_{\mathcal{S}_c}, \mathsf{DK}_{\mathbb{A}_c})$. If the decryption succeeds, C generates DH values $X_1 = g^{x_1}$ and $X_2 = h^{x_2}$, where $x_1, x_2 \xleftarrow{\$} \mathbb{Z}_p^*$. C then computes an authentication tag σ_c for the message $M_c = (\text{“}C \rightarrow S\text{”}, bid, sid, X_1, X_2, Z)$ using K_c from the broadcast message MSG_B. C defines a policy $\mathbb{A}_c$ for S, encrypts its message $\mathsf{MSG}_c = (K_s, M_c)$ to ciphertext CT_c using $\mathcal{FEME}.\mathsf{Enc}$, and sends $(bid, sid, \sigma_c, \mathsf{CT}c)$ to S.

(2) Service Provider Response: S decrypts C's ciphertext, verifies σ_c, and generates its own DH value $Y \leftarrow g^y$ using a random exponent $y \xleftarrow{\$} \mathbb{Z}_p^*$. It creates a message $M_s = (\text{“}S \rightarrow C\text{”}, bid, sid, X_1, X_2, Y, Z)$ and a tag σ_s using the MAC key K_s from MSG_c. S then computes a session key $SSKc, s \leftarrow \bar{H}(X_1^y, X_2^z)$ and sends (M_s, σ_s) to C.

(3) Client Finalization: Upon receiving (M_s, σ_s), C verifies σ_s. If valid, C computes a session key $SSK_{c,s} \leftarrow \bar{H}(Y^{x_1}, Z^{x_2})$ using its secret DH exponents (x_1, x_2). As $X_1^y = Y^{x_1} = g^{x_1 y}$ and $X_2^z = Z^{x_2} = h^{x_2 z}$, both C and S derive the same session key $SSK_{c,s}$.

Security Model. The security model of PriSrv+ follows that of PriSrv [32], which defines service discovery security and bilateral anonymity (i.e., both anonymity of service provider and anonymity of client). *Service discovery security* ensures privacy-preserving service advertisement and anonymous mutual authentication with bilateral policy control, protecting sessions from adversarial exposure. It maintains confidentiality and authentication, allowing only authorized clients and service providers to establish secure communications. The *bilateral anonymity* property implies that neither the PPT service provider nor the PPT client can learn anything about another participant's attribute values unless they satisfy each other's access policies.

Theorem 4. *Suppose that the DDH assumption holds, $\mathcal{FEME}$ is secure, $\mathcal{MAC}$ is unforgeable, and H is a random oracle, then PriSrv+ is a secure service discovery protocol and satisfies bilateral anonymity.*

Proof Sketch. The security proof of PriSrv+ in Theorem 4 parallels that of PriSrv, as the main distinction between the protocols lies in replacing ACME (in PriSrv) with FEME (in PriSrv+). The service discovery security is proved based on the confidentiality and authenticity of FEME in Theorems 1 and 3. The bilateral anonymity is proved based on FEME's anonymity in Theorem 2.

5.2. Comparative Advantages of PriSrv+

In Table I of [32], PriSrv distinguishes itself as the only privacy-preserving service discovery protocol that enhances privacy while maintaining high usability, based on a thorough comparison with 7 standard service discovery protocols (DNS-SD [5], mDNS [19], SSDP [14], UPnP [6],

Protocol	**Expressiveness**			**Security and Privacy**				**Usability**		
	Monotonic Policy	Arbitrary Attribute	Large Universe	Privacy Broadcast	Mutual Authn.	Bilateral Anon.	Attribute Hidden	No Pre-reg. Pairing	No 3rd-party Dependence	No In-advance ID Issuance
PriSrv [32]	√	×	×	√	√	√	×	√	√	×
PriSrv+	√	√	√	√	√	√	√	√	√	√

TABLE 2: Comparison of Private Service Discovery Protocols with Bilateral Policy Control

Wi-Fi [1], BLE [3], AirDrop [2]) and 3 privacy-preserving SD protocols (PrivateDrop [17], CBN [11], WTSB [30]). Instead of revisiting these earlier comparisons, we focus on comparing PriSrv+ directly with PriSrv in terms of expressiveness, security and privacy, and usability. Table 2 summarizes our comparison results.

In terms of *expressiveness*, PriSrv+ supports expressive policies defined by LSSS and uses a large-universe construction to accommodates an unrestricted attribute set, allowing any arbitrary string, such as a postal address, to serve as an attribute. In contrast, ACME's small-universe design restricts PriSrv to a fixed set of attributes, necessitating a full system rebuild to add new attributes, making it less adaptable for large, flexible deployments. PriSrv also employs binary attribute vectors, where each attribute is represented as 1 or 0, leading to longer vectors and increased computational overheads for broader attribute ranges during encryption and decryption.

In terms of *security and privacy*, both protocols provide similar privacy protections. However, PriSrv discloses a set of public attributes and a public access policy in its outer layer during the matching process due to its dual-layer architecture. PriSrv+ addresses this attribute exposure by employing a partially hidden access structure, which separates each attribute into an attribute name and an attribute value. Only the attribute names are revealed during matching, while the attribute values remain confidential.

In terms of *usability*, neither protocol requires pre-registered pairings or third-party dependencies for service discovery. However, PriSrv requires the issuance of anonymous credentials in advance, which adds overhead for credential management. PriSrv+ eliminates this requirement by using the sender's encryption key directly, thereby avoiding the need for credential management.

This comparison shows that PriSrv+ achieves greater expressiveness, improved security and privacy, and usability.

6. Implementation and Evaluation

For a fair comparison, we used the same benchmarks, crypto library, elliptic curves, and parameters as PriSrv [32] when implementing PriSrv+ and PriSrv in C/C++. We utilized (i) MIRACL library[2] for FEME and PriSrv+ implementations, (ii) three elliptic curves, including MNT159 (80-bit security), MNT201 (90-bit security), and BN256 (100-bit security) to evaluate different security levels[3], (iii) SHA-256 for the the hash function, and (iv) MAC-GGM [12] for the MAC scheme. Our source code is available at [20].

2. MIRACL: multiprecision integer and rational arithmetic c/c++ library. https://github.com/miracl/MIRACL.

3. Pairing-Friendly Curves. https://datatracker.ietf.org/doc/draft-irtf-cfrg-pairing-friendly-curves.

6.1. Evaluation of FEME and ACME

Table 3 presents the computation cost (comp.) and communication (comm.) cost of FEME and ACME for various algorithms on a desktop (Intel Core i9-7920X, 12 cores, 16GB RAM). The experimental setup ensures that FEME and ACME operate under equivalent conditions as used in [32], with the same attribute numbers and policies. For FEME, parameters are $\ell_1 = \ell_2 = 4$, $m_1 = m_2 = 2$, and $I_1 = I_2 = 4$, representing the sender's and receiver's attribute numbers, access matrix rows, and the number of attributes satisfying the access policy. ACME parameters are set similarly for comparability: $n = 10$, $\ell_1 = \ell_2 = 4$, $k = 2$, and $I_1 = I_2 = 4$, where n is the system's total attribute number (which is fixed in the small-universe setting), and k is the number of access matrix rows. This alignment allows a fair comparison of both schemes.

Curves		MNT159 (80-bit Security)		MNT201 (90-bit Security)		BN256 (100-bit Security)	
Schemes		ACME	FEME	ACME	FEME	ACME	FEME
Algorithms		Computation Costs (ms)					
Setup		20.526	8.411	26.882	9.699	33.344	11.402
EKGen		35.451	8.124	41.365	9.393	48.485	10.031
DKGen		21.630	4.150	18.640	3.836	15.750	4.787
PolGen		359.807	2.998	327.796	3.026	237.675	3.697
Enc		146.931	19.660	167.337	20.302	187.822	18.275
Dec		123.772	20.109	188.346	28.832	231.214	27.283
Algo.	Param.	Communication Costs (KB)					
Setup	|mpk|	1.044	0.344	1.332	0.428	4.128	1.071
Setup	|msk|	1.2	0.065	1.36	0.074	1.6	0.083
EKGen	|EK|	0.172	0.320	0.220	0.417	0.544	1.184
DKGen	|DK|	0.86	0.235	1.1	0.306	2.72	0.912
PolGen	|SK|	13.932	0.127	17.82	0.165	44.064	1.169
Enc	|CT|	164.34	23.287	212.964	29.599	537.984	63.104

TABLE 3: Performance of FEME and ACME (on Desktop)

Table 3 shows that FEME's computation (upper part) and communication costs (lower part) are substantially lower than ACME's. While both schemes see performance declines as security levels increase from 80-bit to 100-bit, FEME's advantage generally grows, except for DKGen and PolGen, where the performance gap narrows.

In Setup, ACME's computation cost is $(n+3)k^2 \cdot \exp_1 + k \cdot \exp_T$, while FEME's is only $3\exp_2 + 2\exp_T$, where $\exp_1$, $\exp_2$, and $\exp_T$ represent the computation costs of exponentiation in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$, respectively. ACME averages 26.917 ms for system setup, whereas FEME takes 9.837 ms, making it 1.74× faster. FEME also reduces the

Curves		MNT159 (80-bit Security)				MNT201 (90-bit Security)				BN256 (100-bit Security)			
Costs		Comp. (ms)		Comm. (KB)		Comp. (ms)		Comm. (KB)		Comp. (ms)		Comm. (KB)	
Protocols		PriSrv	PriSrv+	PriSrv	PriSrv+	PriSrv	PriSrv+	PriSrv	PriSrv+	PriSrv	PriSrv+	PriSrv	PriSrv+
No.	Devices	Privacy-preserving Service Broadcast											
1	Desktop	158.931	22.891	164.34	23.29	180.337	23.512	212.96	29.602	202.822	20.674	537.98	63.107
2	Laptop	216.493	31.168	164.34	23.29	261.059	34.035	212.96	29.602	287.287	29.281	537.98	63.107
3	Phone	385.553	55.531	164.34	23.29	443.686	57.853	212.96	29.602	482.725	49.202	537.98	63.107
4	Raspberry Pi	638.259	91.927	164.34	23.29	880.868	114.832	212.96	29.602	1188.392	121.139	537.98	63.107
No.	Devices	Anonymous Mutual Authentication											
1	Desktop	429.282	97.067	164.45	24.69	517.512	119.389	213.09	31.364	673.039	158.003	538.83	66.385
2	Laptop	576.161	130.268	164.45	24.69	686.054	158.271	213.09	31.364	854.177	201.518	538.83	66.385
3	Phone	727.572	164.512	164.45	24.69	892.712	205.952	213.09	31.364	972.163	228.222	538.83	66.385
4	Raspberry Pi	1224.365	276.851	164.45	24.69	1832.187	422.671	213.09	31.364	2711.013	636.443	538.83	66.385

TABLE 4: Performance of PriSrv+ and PriSrv (on Four Platforms)

sizes of the master public key and master private key by 71.68% and 94.66%, respectively.

In EKGen, FEME's time cost is $(l_1+2)\exp_1 + 2\exp_2$, while ACME's time cost is $(n+2)\exp_1 + 7\exp_2 + 2\,pair$, where $pair$ is the computation time for a bilinear pairing operation. FEME averages 9.183 ms versus ACME's 41.767 ms, making it 3.55× faster, though FEME's EK size is larger at 0.640 KB compared to ACME's 0.312 KB.

In DKGen and PolGen, FEME shows significant improvements: DKGen time drops from 18.673 ms to 4.258 ms, and PolGen time from 308.426 ms to 3.240 ms, achieving 3.39× and 94.19× speedups, respectively. FEME's DK and SK sizes are also reduced by 68.97% and 98.07%, respectively. FEME's Enc and Dec are 7.62× and 6.23× faster than ACME's, with times reduced from 167.383 ms to 19.412 ms and from 181.111 ms to 25.041 ms, respectively. Additionally, FEME's ciphertext size |CT| is 87.33% smaller on average.

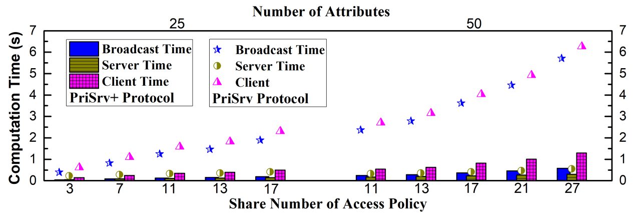
Overall, FEME significantly outperforms ACME, except in the |EK| size of EKGen.

6.2. Evaluation of PriSrv+ and PriSrv

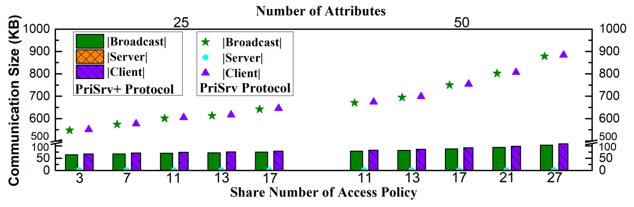
To comprehensively evaluate PriSrv+ and PriSrv [32], we conducted performance tests on four platforms as shown in Table 4, using the same parameter as in Table 3 across three elliptic curves. The platforms include a desktop (Intel Core i9-7920X, 12 cores, 16GB RAM), a laptop (Intel Core i5-10210U, 4 cores, 8GB RAM), a smartphone (ARM Cortex @2.4GHz, 4 cores, 4GB RAM), and a Raspberry Pi (ARM Cortex @1.5GHz, 4 cores, 2GB RAM). The desktop setup matched our earlier evaluations of FEME and ACME, while the other platforms were chosen for mobile device relevance to wireless service discovery.

PriSrv+ outperforms PriSrv in computation overhead (comp.) across all scenarios. During the privacy-preserving broadcast phase (upper section of Table 4), PriSrv+ averages 54.336 ms across four platforms and three elliptic curves, whereas PriSrv averages 443.868 ms, making PriSrv+ 7.17× faster. In the anonymous mutual authentication phase (lower section), PriSrv+ averages 233.181 ms, compared to 1008.02 ms for PriSrv, showing a 3.32× improvement. PriSrv+ achieves an average total computation time of 287.517 ms for both phases, which is well below one second and thus can be perceived as an "immediate response" by users [17], [32]. In comparison, PriSrv's total computation time is 1451.88 ms, making PriSrv+ 4.05× faster than PriSrv.

Table 4 indicates similar communication costs between the two phases within each protocol, dominated by the ciphertext size. FEME's efficiency results in PriSrv+ reducing broadcast phase communication by 87.33% and authentication phase by 86.64% compared to PriSrv on average.



(a) Computation Overheads



(b) Communication Overheads

Figure 7: Comparison of PriSrv+ and PriSrv

We implemented two protocols in a real-world wireless environment using an open-source Wi-Fi Alliance project [1] supporting IEEE 802.1X, with *wpa_supplicant* for the client and *hostapd* for the service provider. Experiments were run on two laptops using BN256.

Fig. 7(a) presents the computation costs of two protocols with more complex policies and larger attribute numbers, where the bars represent PriSrv+'s broadcast time (T_B), server's computation time (T_S), and client's computation time (T_C) during the anonymous authentication phase, while the symbols represent those for PriSrv. We compare them by varying the number of attributes (top x-axis) and the share number of access policy (bottom x-axis), where the latter is the number of shares needed to reconstruct a shared secret based on the access policy [9] (i.e., the edge number in an equivalent access tree). The experiment ensures that they share the same number of attributes and policies for a fair comparison.

In Fig. 7(a), we vary the attribute number between $\{25, 50\}$ and the share number of the access policy among $\{3, 7, 11, 13, 17, 21, 27\}$ to test performance with expressive policies. PriSrv+ shows significantly lower computation times than PriSrv. Average times for PriSrv+ are T_B = 0.252 s, T_S = 0.166 s, and T_C = 0.592 s, while for PriSrv, T_B = 2.478 s, T_S = 0.374 s, and T_C = 2.853 s, making PriSrv+ 8.83×, 1.25×, and 3.82× faster, respectively.

Fig. 7(b) shows communication overheads using the same parameters as Fig. 7(a). Bars indicate PriSrv+ broadcast and the service provider's/client's authentication overheads ($|\mathsf{Broadcast}|$, $|\mathsf{Server}|$, $|\mathsf{Client}|$), with PriSrv shown as symbols. The server's communication cost during the authentication phase is constant and identical for both protocols at $|\mathsf{Server}|$ = 0.82 KB. PriSrv+ has average communication costs of $|\mathsf{Broadcast}|$ = 79.623 KB and $|\mathsf{Client}|$ = 83.64 KB, while PriSrv averages $|\mathsf{Broadcast}|$ = 677.488 KB and $|\mathsf{Client}|$ = 681.505 KB. PriSrv+ reduces these costs by 88.25% and 87.73%, respectively, compared to PriSrv.

The evaluation shows that PriSrv+ is notably more efficient than PriSrv in computation and communication costs.

6.3. Interoperability

PriSrv+ is an enhanced version of PriSrv, improving expressiveness, privacy, and usability. Like PriSrv, it is interoperable with existing wireless service discovery protocols such as mDNS, BLE, EAP, and AirDrop, aligning with the evolving security needs of modern wireless networks.

Integration Approaches. PriSrv+ can be integrated into wireless protocols through two approaches. The first positions PriSrv+ at the application layer, allowing it to work with existing lower-layer protocols. If payload size exceeds protocol limits, lower layers manage segmentation and reassembly without modifying PriSrv+'s logic. The second approach replaces lower-layer protocols with PriSrv+, requiring specific adaptations. Below we use mDNS and BLE as examples to illustrate the first approach and use EAP and AirDrop as examples to illustrate the second.

Privacy-Enhanced mDNS and BLE. PriSrv+ can integrate into the Vanadium framework [4] to develop privacy-enhanced mDNS and BLE. Vanadium provides service discovery APIs for protocols like mDNS [19] and BLE [3]. mDNS, often paired with DNS Service Discovery (DNS-SD) [5], carries additional attributes in TXT records of up to 65,535 bytes. PriSrv+ broadcasts (bid, CT_B) using 64,622 bytes on BN256, fitting within a single TXT record. In contrast, PriSrv's broadcast of the same message takes 531,996 bytes on BN256 and thus requires nine TXT records. PriSrv+ reduces the mDNS packet size by 88.89%, demonstrating its efficiency in minimizing transmission size.

BLE's standard 31-byte broadcast payload challenges transmitting large ciphertexts. To support privacy-enhanced BLE with PriSrv+, the BLE Attribute Protocol (ATT) and PDU Segmentation can increase payload capacity. If the payload exceeds BLE's standard packet size, ATT segments the data into multiple PDUs for sequential transmission, which are reassembled by the receiver. PriSrv+ achieves an average 87.73% reduction for clients compared to PriSrv, enabling efficient segmentation and minimal fragmentation.

Privacy-Enhanced EAP. PriSrv+ enhances privacy in the Extensible Authentication Protocol (EAP) by integrating encrypted broadcasts and responses. An access point (AP) facilitates interactions between the client and service provider, acting as a pass-through for the authentication server. The service provider broadcasts privacy-preserving service information, including a broadcast identifier and ciphertext generated using FEME, aligning with EAP's initial authentication request. If the client successfully decrypts a broadcast message, it responds with an encrypted reply. The service provider decrypts the reply and, upon success, sends an authentication tag to the client, containing DH shares and a MAC. The client verifies the authentication tag, computes a secret session key, and signals success to the server. The service provider computes the same secret session key to establish a secure session. After this, EAP messages are encapsulated in EAPOL frames and sent as RADIUS packets. Compared to PriSrv, PriSrv+ reduces communication costs by 86.64% during mutual authentication, making it more suitable for privacy-preserving Wi-Fi connections in resource-constrained environments.

Privacy-Enhanced Apple AirDrop. AirDrop uses BLE to broadcast the hashed identity of a service provider to detect nearby potential clients. If a match is found, a TLS handshake follows, exchanging their certificates in cleartext and exposing their identities to potential tracking attacks. Using the PrivateDrop mechanism [17], PriSrv+ enhances AirDrop's privacy by preventing the transmission of service provider identifiers during BLE advertising and encrypting both parties' certificates with FEME at the start of the TLS handshake. In this setting, Apple may act as a key generation center to generate necessary secret keys for the involving parties alongside existing iCloud certificates.

7. Conclusion

PriSrv+ marks a significant advance in privacy-preserving service discovery protocols for wireless networks. By introducing Fast and Expressive Matchmaking Encryption (FEME), PriSrv+ addresses the limitations of prior schemes, supporting expressive access control policies while improving efficiency, security and privacy, and usability. Comprehensive evaluations show substantial improvements in encryption and decryption performance, with reduced ciphertext size and communication overhead, making PriSrv+ suitable for resource-constrained devices. The interoperability of PriSrv+ with existing wireless service discovery protocols, coupled with formal security proofs, ensures that PriSrv+ fulfills the security and privacy requirements of modern wireless networks.

References

[1] Wi-fi. https://w1.fi, 2023.

[2] Airdrop. https://support.apple.com/en-us/HT204144, 2024.

[3] Bluetooth. https://www.bluetooth.com, 2024.

[4] Vanadium. https://vanadium.github.io/, 2024.

[5] DNS-SD. RFC 6763. Dns-based service discovery, 2013.

[6] UPnP. RFC 6970. Universal plug and play (upnp) internet gateway device - port control protocol interworking function, 2013.

[7] Giuseppe Ateniese, Danilo Francati, David Nunez, and Daniele Venturi. Match me if you can: Matchmaking encryption and its applications. In *CRYPTO*, 2019.

[8] Xiaolong Bai, Luyi Xing, Nan Zhang, XiaoFeng Wang, Xiaojing Liao, Tongxin Li, and Shi-Min Hu. Staying secure and unprepared: understanding and mitigating the security risks of apple zeroconf. In *IEEE S&P*, 2016.

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

[10] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *IEEE S&P*, 2007.

[11] Aldo Cassola, Erik-Oliver Blass, and Guevara Noubir. Authenticating privately over public wi-fi hotspots. In *CCS*, 2015.

[12] Melissa Chase, Sarah Meiklejohn, and Greg Zaverucha. Algebraic macs and keyed-verification anonymous credentials. In *CCS*, 2014.

[13] Jie Chen, Yu Li, Jinming Wen, and Jian Weng. Identity-based matchmaking encryption from standard assumptions. In *ASIACRYPT*. Springer, 2022.

[14] SSDP. IETF draft-cai-ssdp-v1 03. Simple service discovery protocol/1.0 operating without on arbiter, 1999.

[15] Kassem Fawaz, Kyu-Han Kim, and Kang G Shin. Protecting privacy of ble device users. In *USENIX Security*, 2016.

[16] Danilo Francati, Alessio Guidi, Luigi Russo, and Daniele Venturi. Identity-based matchmaking encryption without random oracles. *INDOCRYPT*, 2021.

[17] Alexander Heinrich, Matthias Hollick, Thomas Schneider, Milan Stute, and Christian Weinert. Privatedrop: Practical privacy-preserving authentication for apple airdrop. In *USENIX Security*, 2021.

[18] Bastian Könings, Christoph Bachmaier, Florian Schaub, and Michael Weber. Device names in the wild: Investigating privacy risks of zero configuration networking. In *MDM*, 2013.

[19] mDNS. RFC 6762. Multicast dns, 2013.

[20] PriSrv+. https://github.com/PriSrv-Plus, 2024.

[21] Doreen Riepel and Hoeteck Wee. Fabeo: Fast attribute-based encryption with optimal security. In *CCS*, 2022.

[22] Jacob T Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *Journal of the ACM*, 1980.

[23] Victor Shoup. Lower bounds for discrete logarithms and related problems. In *EUROCRYPT*, 1997.

[24] Milan Stute, Alexander Heinrich, Jannik Lorenz, and Matthias Hollick. Disrupting continuity of apple's wireless ccosystem security: new tracking, dos, and mitm attacks on ios and macos through bluetooth low energy, awdl, and wi-fi. In *USENIX Security*, 2021.

[25] Milan Stute, Sashank Narain, Alex Mariotto, Alexander Heinrich, David Kreitschmann, Guevara Noubir, and Matthias Hollick. A billion open interfaces for eve and mallory: Mitm, dos, and tracking attacks on ios and macos through apple wireless direct link. In *USENIX Security*, 2019.

[26] Jianfei Sun, Guowen Xu, Tianwei Zhang, Xuehuan Yang, Mamoun Alazab, and Robert H Deng. Privacy-aware and security-enhanced efficient matchmaking encryption. *IEEE TIFS*, 2023.

[27] Raghav H Venkatnarayan, Muhammad Shahzad, Sangki Yun, Christina Vlachou, and Kyu-Han Kim. Leveraging polarization of wifi signals to simultaneously track multiple people. In *IMWUT*, 2020.

[28] Xueqiang Wang, Yuqiong Sun, Susanta Nanda, and XiaoFeng Wang. Looking from the mirror: Evaluating iot device security through mobile companion apps. In *USENIX Security*, 2019.

[29] Axin Wu, Weiqi Luo, Jian Weng, Anjia Yang, and Jinghang Wen. Fuzzy identity-based matchmaking encryption and its application. *IEEE TIFS*, 2023.

[30] David J Wu, Ankur Taly, Asim Shankar, and Dan Boneh. Privacy, discovery, and authentication for the internet of things. In *ESORICS*, 2016.

[31] Ningbin Yang, Chunming Tang, and Debiao He. A lightweight certificateless multi-user matchmaking encryption for mobile devices: Enhancing security and performance. *IEEE TIFS*, 2023.

[32] Yang Yang, Robert H Deng, Guomin Yang, Yingjiu Li, HweeHwa Pang, Minming Huang, Rui Shi, and Jian Weng. Prisrv: Privacy-enhanced and highly usable service discovery in wireless communications. In *NDSS*, 2024. https://dx.doi.org/10.14722/ndss.2024.24174, https://eprint.iacr.org/2024/1783 (Full version).

[33] Wei Zhou, Yan Jia, Yao Yao, Lipeng Zhu, Le Guan, Yuhang Mao, Peng Liu, and Yuqing Zhang. Discovering and understanding the security hazards in the interactions between iot devices, mobile apps, and clouds on smart home platforms. In *USENIX Security*, 2019.

[34] Richard Zippel. Probabilistic algorithms for sparse polynomials. In *ISSAC*, 1979.

Appendix

A. Definition and Security Model of FEME

A.1. FEME: Syntax Definition

FEME is composed of the following polynomial-time algorithms.

$\underline{(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)}$: Upon input the security parameter 1^λ, the system setup algorithm outputs the master public key mpk and the master secret key msk.

$\underline{\mathsf{EK}_{\mathcal{S}_{\mathsf{snd}}} \leftarrow \mathsf{EKGen}(\mathsf{msk}, \mathcal{S}_{\mathsf{snd}})}$: Upon input the master secret key msk and sender's attributes $\mathcal{S}_{\mathsf{snd}}$, it outputs a sender attribute encryption key $\mathsf{EK}_{\mathcal{S}_{\mathsf{snd}}}$.

$\underline{\mathsf{DK}_{\mathcal{S}_{\mathsf{rcv}}} \leftarrow \mathsf{DKGen}(\mathsf{msk}, \mathcal{S}_{\mathsf{rcv}})}$: Upon input master secret key msk and receiver's attributes $\mathcal{S}_{\mathsf{rcv}}$, it outputs a receiver attribute decryption key $\mathsf{DK}_{\mathcal{S}_{\mathsf{rcv}}}$.

$\underline{\mathsf{SK}_{\mathbb{A}_{\mathsf{rcv}}} \leftarrow \mathsf{PolGen}(\mathsf{msk}, \mathbb{A}_{\mathsf{rcv}})}$:Upon input master secret key msk and receiver's access structure $\mathbb{A}_{\mathsf{rcv}}$, it outputs a receiver's policy decryption key $\mathsf{SK}_{\mathbb{A}_{\mathsf{rcv}}}$.

$\underline{\mathsf{CT}_{\mathsf{snd}} \leftarrow \mathsf{Enc}(\mathsf{EK}_{\mathcal{S}_{\mathsf{snd}}}, \mathbb{A}_{\mathsf{snd}}, \mathsf{msg})}$: Upon input sender's attribute encryption key $\mathsf{EK}_{\mathcal{S}_{\mathsf{snd}}}$, access structure $\mathbb{A}_{\mathsf{snd}}$, and a message msg, it outputs a ciphertext $\mathsf{CT}_{\mathsf{snd}}$.

$\underline{\mathsf{msg}/\perp \leftarrow \mathsf{Dec}(\mathsf{DK}_{\mathcal{S}_{\mathsf{rcv}}}, \mathsf{SK}_{\mathbb{A}_{\mathsf{rcv}}}, \mathsf{CT}_{\mathbb{A}_{\mathsf{snd}}, \mathcal{S}_{\mathsf{snd}}})}$: Upon input receiver's attribute decryption key $\mathsf{DK}_{\mathcal{S}_{\mathsf{rcv}}}$, policy decryption key $\mathsf{SK}_{\mathbb{A}_{\mathsf{rcv}}}$, and a ciphertext $\mathsf{CT}_{\mathsf{snd}}$, it outputs a message msg if $\mathcal{S}_{\mathsf{snd}} \models \mathbb{A}_{\mathsf{rcv}}$ (denoting that $\mathcal{S}_{\mathsf{snd}}$ satisfies $\mathbb{A}_{\mathsf{rcv}}$) and $\mathcal{S}_{\mathsf{rcv}} \models \mathbb{A}_{\mathsf{snd}}$ (denoting that $\mathcal{S}_{\mathsf{rcv}}$ satisfies $\mathbb{A}_{\mathsf{snd}}$); and $\perp$ (denoting an error) otherwise.

A.2. Security Model of FEME

Definition 3. *A FEME scheme $\mathcal{FEME}$ is correct if for all security parameter λ,*

$$\mathsf{Pr}\left[\begin{array}{l} (\mathsf{mpk}, \mathsf{msk}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda); \\ \mathsf{EK}_{\mathcal{S}_{\mathsf{snd}}} \xleftarrow{\$} \mathsf{EKGen}(\mathsf{msk}, \mathcal{S}_{\mathsf{snd}}); \\ \mathsf{DK}_{\mathcal{S}_{\mathsf{rcv}}} \xleftarrow{\$} \mathsf{DKGen}(\mathsf{msk}, \mathcal{S}_{\mathsf{rcv}}); \\ \mathsf{SK}_{\mathbb{A}_{\mathsf{rcv}}} \xleftarrow{\$} \mathsf{PolGen}(\mathsf{msk}, \mathbb{A}_{\mathsf{rcv}}); \\ \mathsf{CT}_{\mathsf{snd}} \xleftarrow{\$} \mathsf{Enc}(\mathsf{EK}_{\mathcal{S}_{\mathsf{snd}}}, \mathbb{A}_{\mathsf{snd}}, \mathsf{msg}) : \\ \mathcal{S}_{\mathsf{snd}} \models \mathbb{A}_{\mathsf{rcv}} \bigwedge \mathcal{S}_{\mathsf{rcv}} \models \mathbb{A}_{\mathsf{snd}} \bigwedge \\ \mathsf{Dec}(\mathsf{DK}_{\mathcal{S}_{\mathsf{rcv}}}, \mathsf{SK}_{\mathbb{A}_{\mathsf{rcv}}}, \mathsf{CT}_{\mathsf{snd}}) = \perp \end{array}\right]$$

$\leqslant \nu(\lambda)$, where ν is a negligible function.

Definition 4. *A FEME scheme $\mathcal{FEME}$ satisfies confidentiality if for any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a negligible function ν such that $\mathsf{Adv}^{\mathsf{conf}}_{\mathcal{FEME}}(\lambda) \overset{\mathsf{def}}{=}$*

$$\mathsf{Pr}\left[b' = b \left| \begin{array}{l} (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda), b \xleftarrow{\$} \{0,1\} \\ (\mathsf{msg}^*_0, \mathsf{msg}^*_1, \mathcal{S}^*_{\mathsf{snd}}, \mathbb{A}^*_{\mathsf{snd}}) \\ \qquad\qquad \leftarrow \mathcal{A}_1^{\mathcal{O}_1,\mathcal{O}_2,\mathcal{O}_3}(\mathsf{mpk}) \\ \mathsf{CT}^*_{\mathsf{snd}} \xleftarrow{\$} \mathsf{Enc}(\mathsf{EK}_{\mathcal{S}^*_{\mathsf{snd}}}, \mathbb{A}^*_{\mathsf{snd}}, \mathsf{msg}_b) \\ b' \leftarrow \mathcal{A}_2^{\mathcal{O}_1,\mathcal{O}_2,\mathcal{O}_3}(\mathsf{CT}^*_{\mathsf{snd}}) \end{array} \right.\right]$$

$\leqslant \nu(\lambda)$, where oracles $\mathcal{O}_1$, $\mathcal{O}_2$, $\mathcal{O}_3$ are implemented by $\mathsf{EKGen}\ (\mathsf{msk}, \cdot)$, $\mathsf{DKGen}(\mathsf{msk}, \cdot)$, $\mathsf{PolGen}(\mathsf{msk}, \cdot)$, respectively. It is required that $\mathcal{O}_2$ and $\mathcal{O}_3$ are not queried for attributes and policies that can satisfy $(\mathcal{S}^*_{\mathsf{snd}}, \mathbb{A}^*_{\mathsf{snd}})$.

Definition 5. *A FEME scheme $\mathcal{FEME}$ satisfies anonymity if for any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a negligible function ν such that $\mathsf{Adv}^{\mathsf{anon}}_{\mathcal{FEME}}(\lambda) \overset{\mathsf{def}}{=}$*

$$\mathsf{Pr}\left[b' = b \left| \begin{array}{l} (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda), b \xleftarrow{\$} \{0,1\} \\ (\mathsf{msg}^*, \mathcal{S}^*_{\mathsf{snd}_0}, \mathcal{S}^*_{\mathsf{snd}_1}, \mathbb{A}^*_{\mathsf{snd}_0}, \mathbb{A}^*_{\mathsf{snd}_1}) \\ \qquad\qquad \leftarrow \mathcal{A}_1^{\mathcal{O}_1,\mathcal{O}_2,\mathcal{O}_3}(\mathsf{mpk}) \\ \mathsf{CT}^*_{\mathsf{snd}} \xleftarrow{\$} \mathsf{Enc}(\mathsf{EK}_{\mathcal{S}^*_{\mathsf{snd}}}, \mathbb{A}^*_{\mathsf{snd}}, \mathsf{msg}_b) \\ b' \leftarrow \mathcal{A}_2^{\mathcal{O}_1,\mathcal{O}_2,\mathcal{O}_3}(\mathsf{CT}^*_{\mathsf{snd}}) \end{array} \right.\right]$$

$\leqslant \nu(\lambda)$, where oracles $\mathcal{O}_1$, $\mathcal{O}_2$, $\mathcal{O}_3$ are implemented by $\mathsf{EKGen}\ (\mathsf{msk}, \cdot)$, $\mathsf{DKGen}(\mathsf{msk}, \cdot)$, $\mathsf{PolGen}(\mathsf{msk}, \cdot)$, respectively. It is required that $\mathcal{O}_2$ and $\mathcal{O}_3$ are not queried for attributes and policies that can satisfy $(\mathcal{S}^*_{\mathsf{snd}_0}, \mathbb{A}^*_{\mathsf{snd}_0})$ or $(\mathcal{S}^*_{\mathsf{snd}_1}, \mathbb{A}^*_{\mathsf{snd}_1})$, where $\mathcal{S}^*_{\mathsf{snd}_0} = \{n_i, v_{i,0}\}_{i\in[\ell_1]}$, $\mathcal{S}^*_{\mathsf{snd}_1} = \{n_i, v_{i,1}\}_{i\in[\ell_1]}$, $\mathbb{A}^*_{\mathsf{snd}_0} = (\boldsymbol{M}, \pi, \Psi_{\pi(i),0} = \{n_{\pi(i)}, v_{\pi(i),0}\}_{i\in[m_1]})$, and $\mathbb{A}^*_{\mathsf{snd}_1} = (\boldsymbol{M}, \pi, \Psi_{\pi(i),1} = \{n_{\pi(i)}, v_{\pi(i),1}\}_{i\in[m_1]})$.

Definition 6. *A FEME scheme $\mathcal{FEME}$ satisfies authenticity if for any PPT adversary $\mathcal{A}$, there exists a negligible function ν such that $\mathsf{Adv}^{\mathsf{auth}}_{\mathcal{FEME}}(\lambda) \overset{\mathsf{def}}{=}$*

$$\mathsf{Pr}\left[\begin{array}{l} (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda) \\ (\mathsf{CT}_{\mathsf{snd}}, \mathbb{A}_{\mathsf{rcv}}, \mathcal{S}_{\mathsf{rcv}}) \leftarrow \mathcal{A}^{\mathcal{O}_1,\mathcal{O}_2,\mathcal{O}_3}(\mathsf{mpk}) \\ \mathsf{DK}_{\mathcal{S}_{\mathsf{rcv}}} \leftarrow \mathsf{DKGen}(\mathsf{msk}, \mathcal{S}_{\mathsf{rcv}}) \\ \mathsf{SK}_{\mathbb{A}_{\mathsf{rcv}}} \leftarrow \mathsf{PolGen}(\mathsf{msk}, \mathbb{A}_{\mathsf{rcv}}) \\ \mathsf{msg} = \mathsf{Dec}(\mathsf{DK}_{\mathcal{S}_{\mathsf{rcv}}}, \mathsf{SK}_{\mathbb{A}_{\mathsf{rcv}}}, \mathsf{CT}_{\mathsf{snd}}) \\ \forall \mathcal{S}_{\mathsf{snd}} \in \mathcal{Q}_{\mathcal{O}_1} : (\mathcal{S}_{\mathsf{snd}} \not\models \mathbb{A}_{\mathsf{rcv}}) \bigwedge (\mathsf{msg} \neq \perp) \end{array}\right]$$

$\leqslant \nu(\lambda)$, where oracles $\mathcal{O}_1$, $\mathcal{O}_2$, $\mathcal{O}_3$ are implemented by $\mathsf{EKGen}(\mathsf{msk}, \cdot)$, $\mathsf{DKGen}(\mathsf{msk}, \cdot)$, $\mathsf{PolGen}(\mathsf{msk}, \cdot)$, and $\mathcal{S} \not\models \mathbb{A}$ denotes that attributes $\mathcal{S}$ does not satisfy $\mathbb{A}$.

B. Correctness Proof of FEME

The correctness of FEME is proved below.

$$\begin{aligned}
& \frac{e(\mathsf{dk}_1, \mathsf{ct}_1) \cdot e(\prod_{i\in I_1} (\mathsf{dk}_{2,\pi(i)})^{\gamma_i}, \mathsf{ct}_2)}{e(\prod_{i\in I_1} (\mathsf{ct}_{3,\pi(i)})^{\gamma_i}, \mathsf{dk}_3)} \\
& \cdot \frac{e(\prod_{i\in I_2} (\mathsf{sk}_{2,\rho(i)})^{\omega_i}, \mathsf{ct}_{4,1}) \cdot e(\prod_{i\in I_2} (\mathsf{sk}_{3,\rho(i)})^{\omega_i}, \mathsf{ct}_{4,2})}{e(\prod_{i\in I_2} (\mathsf{ct}_{5,\rho(i)})^{\omega_i}, \mathsf{sk}_1)} \\
& \cdot \frac{e(\mathsf{ct}_9, \delta_0) \cdot e(\prod_{i\in I_2} (\mathsf{ct}_{6,\rho(i)})^{\omega_i}, \mathsf{sk}_1)}{e(\prod_{i\in I_2} (\mathsf{sk}_{4,\rho(i)})^{\omega_i}, \mathsf{ct}_7) \cdot e(\prod_{i\in I_2} (\mathsf{sk}_{5,\rho(i)})^{\omega_i}, \mathsf{ct}_8)} \\
= \; & \frac{e(g_1^{\alpha} h^r, g_2^{s_1}) \cdot e(\prod_{i\in I_1} (H(\Psi_{\pi(i)})^r)^{\gamma_i}, g_2^{s_3})}{e(\prod_{i\in I_1} (h^{\mathbf{M}_i(s_1||\mathbf{v})^\top} \cdot H(\Psi_{\pi(i)})^{s_3})^{\gamma_i}, g_2^r)} \\
& \cdot \frac{e(\prod_{i\in I_2} (g_1^{\mathbf{A}_i(\alpha||\mathbf{y})^\top} \cdot H(\Psi_{\rho(i)})^{r'})^{\omega_i}, g_2^{s_2})}{e(\prod_{i\in I_2} (H(\Psi_{\rho(i)})^{s_2})^{\omega_i}, g_2^{r'})} \\
& \cdot \frac{e(g_1^x h^{\tau_1}, g_2^{\mu})^{s_3} \cdot e(\prod_{i\in I_2} (H(\Psi_{\rho(i)})^{\tau_1})^{\omega_i}, g_2^{r'})^{s_3}}{e(\prod_{i\in I_2} (h^{\mathbf{A}_i(\mu||\mathbf{y})^\top} \cdot H(\Psi_{\rho(i)})^{r'})^{\omega_i}, g_2^{\tau_1})^{s_3}} \\
= \; & \frac{e(g_1^{\alpha}, g_2^{s_1}) \cdot e(h^r, g_2^{s_1})}{e(\prod_{i\in I_1} (h^{\mathbf{M}_i(s_1||\mathbf{v})^\top})^{\gamma_i}, g_2^r)} \cdot e(\prod_{i\in I_2} (g_1^{\mathbf{A}_i(\alpha||\mathbf{y})^\top})^{\omega_i}, g_2^{s_2}) \\
& \cdot \frac{e(g_1^x, g_2^{\mu})^{s_3} \cdot e(h^{\tau_1}, g_2^{\mu})^{s_3}}{e(\prod_{i\in I_2} (h^{\mathbf{A}_i(\mu||\mathbf{y})^\top})^{\omega_i}, g_2^{\tau_1})^{s_3}} \\
= \; & e(g_1, g_2)^{\alpha s_1} \cdot e(g_1, g_2)^{\alpha s_2} \cdot e(g_1, g_2)^{x\mu s_3} = V,
\end{aligned}$$

where $\sum_{i\in I_1} (\mathbf{M}_i(s_1||\mathbf{v})^\top \cdot \gamma_i) = s_1$, $\sum_{i\in I_2} \mathbf{A}_i(\alpha||\mathbf{y})^\top \cdot \omega_i = \alpha$, $\sum_{i\in I_2} (\mathbf{A}_i(\mu||\mathbf{y})^\top \cdot \omega_i) = \mu$ and $\tau_1 = \tau + \tau'$.

Then, compute $\phi(\mathsf{msg}) = \mathsf{ct}_0 \oplus \hat{H}(V)$. If the padding is valid, this algorithm returns msg. Otherwise, it returns $\perp$.

C. Security Proofs of FEME

Generic Group Model (GGM). We use an extended GGM for bilinear groups, as outlined in [23]. This model includes three random encodings σ_1, σ_2, and σ_T for the additive group $\mathbb{Z}_q$. These encodings are injective mappings $\sigma_1, \sigma_2, \sigma_T : \mathbb{Z}_q \to \{0,1\}^m$, where $m > 3\log(q)$. The probability that an adversary $\mathcal{A}$ can guess an element within the image of σ_1, σ_2, or σ_T is negligible. For $i = 1, 2, T$, we define the sets $\mathbb{G}_i = \sigma_i(x) : x \in \mathbb{Z}_p$. The model provides oracles to compute group operations on $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$, as well as an oracle for a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$.

Random Oracle. The challenger $\mathcal{C}$ maintains a list $\mathcal{L}_H$ containing entries of the form (u_i, h_i, t_i), which starts out empty. When the adversary $\mathcal{A}$ queries the oracle with an attribute $u_i = (n_i, v_i)$, the challenger checks whether a tuple with u_i already exists in $\mathcal{L}_H$. If such a tuple is found, $\mathcal{C}$ responds with $H(u_i) = h_i \in \mathbb{G}_1$. If no match is found, $\mathcal{C}$ selects a random value $t_i \xleftarrow{\$} \mathbb{Z}_p^*$, computes $h_i = g_1^{t_i} \in \mathbb{G}_1$, returns $H(u_i) = h_i$, and adds the tuple (u_i, h_i, t_i) to $\mathcal{L}_H$.

C.1. Proof of Theorem 1 (FEME: Confidentiality)

Proof. Our proof close follows the proof structure in [10]. We start with a standard observation derived from a basic

1	κ	t_i	$t_i\tau$	t_ir	t_is_2	$x+\kappa\tau$	$\kappa\lambda_i+t_\pi s_3$	$t_i\tau_1s_3$	$(x+\kappa\tau_1)s_3$	$\frac{1}{b_1}(\lambda_i+t_\rho r')$	$\frac{1}{b_2}(\lambda_i+t_\rho r')$	$\frac{1}{b_1}(\kappa\psi_i+t_\rho r')$	$\frac{1}{b_2}(\kappa\psi_i+t_\rho r')$
μ	$\mu\kappa$	μt_i	$\mu t_i\tau$	μt_ir	μt_is_2	$\mu(x+\kappa\tau)$	$\mu(\kappa\lambda_i+t_\pi s_3)$	$\mu t_i\tau_1s_3$	$\mu(x+\kappa\tau_1)s_3$	$\frac{\mu}{b_1}(\lambda_i+t_\rho r')$	$\frac{\mu}{b_2}(\lambda_i+t_\rho r')$	$\frac{\mu}{b_1}(\kappa\psi_i+t_\rho r')$	$\frac{\mu}{b_2}(\kappa\psi_i+t_\rho r')$
b_1	$b_1\kappa$	b_1t_i	$b_1t_i\tau$	b_1t_ir	$b_1t_is_2$	$b_1(x+\kappa\tau)$	$b_1(\kappa\lambda_i+t_\pi s_3)$	$b_1t_i\tau_1s_3$	$b_1(x+\kappa\tau_1)s_3$	$\lambda_i+t_\rho r'$	$\frac{b_1}{b_2}(\lambda_i+t_\rho r')$	$(\kappa\psi_i+t_\rho r')$	$\frac{b_1}{b_2}(\kappa\psi_i+t_\rho r')$
b_2	$b_2\kappa$	b_2t_i	$b_2t_i\tau$	b_2t_ir	$b_2t_is_2$	$b_2(x+\kappa\tau)$	$b_2(\kappa\lambda_i+t_\pi s_3)$	$b_2t_i\tau_1s_3$	$b_2(x+\kappa\tau_1)s_3$	$\frac{b_2}{b_1}(\lambda_i+t_\rho r')$	–	$\frac{b_2}{b_1}(\kappa\psi_i+t_\rho r')$	–
r	$r\kappa$	rt_i	$rt_i\tau$	t_ir^2	rt_is_2	$r(x+\kappa\tau)$	$r(\kappa\lambda_i+t_\pi s_3)$	$rt_i\tau_1s_3$	$r(x+\kappa\tau_1)s_3$	$\frac{r}{b_1}(\lambda_i+t_\rho r')$	$\frac{r}{b_2}(\lambda_i+t_\rho r')$	$\frac{r}{b_1}(\kappa\psi_i+t_\rho r')$	$\frac{r}{b_2}(\kappa\psi_i+t_\rho r')$
r'	$r'\kappa$	$r't_i$	$r't_i\tau$	t_irr'	$r't_is_2$	$r'(x+\kappa\tau)$	$r'(\kappa\lambda_i+t_\pi s_3)$	$r't_i\tau_1s_3$	$r'(x+\kappa\tau_1)s_3$	$\frac{r'}{b_1}(\lambda_i+t_\rho r')$	$\frac{r'}{b_2}(\lambda_i+t_\rho r')$	$\frac{r'}{b_1}(\kappa\psi_i+t_\rho r')$	$\frac{r'}{b_2}(\kappa\psi_i+t_\rho r')$
s_1	$s_1\kappa$	s_1t_i	$s_1t_i\tau$	s_1t_ir	$s_1t_is_2$	$s_1(x+\kappa\tau)$	$s_1(\kappa\lambda_i+t_\pi s_3)$	$s_1t_i\tau_1s_3$	$s_1(x+\kappa\tau_1)s_3$	$\frac{s_1}{b_1}(\lambda_i+t_\rho r')$	$\frac{s_1}{b_2}(\lambda_i+t_\rho r')$	$\frac{s_1}{b_1}(\kappa\psi_i+t_\rho r')$	$\frac{s_1}{b_2}(\kappa\psi_i+t_\rho r')$
s_3	$s_3\kappa$	s_3t_i	$s_3t_i\tau$	s_3t_ir	$s_3t_is_2$	$s_3(x+\kappa\tau)$	$s_3(\kappa\lambda_i+t_\pi s_3)$	$t_i\tau_1s_3^2$	$(x+\kappa\tau_1)s_3^2$	$\frac{s_3}{b_1}(\lambda_i+t_\rho r')$	$\frac{s_3}{b_2}(\lambda_i+t_\rho r')$	$\frac{s_3}{b_1}(\kappa\psi_i+t_\rho r')$	$\frac{s_3}{b_2}(\kappa\psi_i+t_\rho r')$
$b_1\tau$	$b_1\tau\kappa$	$b_1\tau t_i$	$b_1\tau t_i\tau$	$b_1\tau t_ir$	$b_1\tau t_is_2$	$(x+\kappa\tau)$	$b_1\tau(\kappa\lambda_i+t_\pi s_3)$	$b_1\tau t_i\tau_1s_3$	$b_1\tau(x+\kappa\tau_1)s_3$	$\tau(\lambda_i+t_\rho r')$	$\frac{b_1\tau}{b_2}(\lambda_i+t_\rho r')$	$\tau(\kappa\psi_i+t_\rho r')$	$\frac{b_1\tau}{b_2}(\kappa\psi_i+t_\rho r')$
$b_2\tau$	$b_2\tau\kappa$	$b_2\tau t_i$	$b_2\tau t_i\tau$	$b_2\tau t_ir$	$b_2\tau t_is_2$	$(x+\kappa\tau)$	$b_2\tau(\kappa\lambda_i+t_\pi s_3)$	$b_2\tau t_i\tau_1s_3$	$b_2\tau(x+\kappa\tau_1)s_3$	$\frac{b_2\tau}{b_1}(\lambda_i+t_\rho r')$	–	$\frac{b_2\tau}{b_1}(\kappa\psi_i+t_\rho r')$	–
$b_1s'_2$	$b_1s'_2\kappa$	$b_1s'_2t_i$	$b_1s'_2t_i\tau$	$b_1s'_2t_ir$	$b_1s'_2t_is_2$	$b_1s'_2(x+\kappa\tau)$	$b_1s'_2(\kappa\lambda_i+t_\pi s_3)$	$b_1s'_2t_i\tau_1s_3$	$b_1s'_2(x+\kappa\tau_1)s_3$	$s'_2(\lambda_i+t_\rho r')$	$\frac{b_1s'_2}{b_2}(\lambda_i+t_\rho r')$	$s'_2(\kappa\psi_i+t_\rho r')$	$\frac{b_1s'_2}{b_2}(\kappa\psi_i+t_\rho r')$
$b_2s''_2$	$b_2s''_2\kappa$	$b_2s''_2t_i$	$b_2s''_2t_i\tau$	$b_2s''_2t_ir$	$b_2s''_2t_is_2$	$b_2s''_2(x+\kappa\tau)$	$b_2s''_2(\kappa\lambda_i+t_\pi s_3)$	$b_2s''_2t_i\tau_1s_3$	$b_2s''_2(x+\kappa\tau_1)s_3$	$\frac{b_2s''_2}{b_1}(\lambda_i+t_\rho r')$	$s''_2(\lambda_i+t_\rho r')$	$\frac{b_2s''_2}{b_1}(\kappa\psi_i+t_\rho r')$	$s''_2(\kappa\psi_i+t_\rho r')$
$\alpha+\kappa r$	$(\alpha+\kappa r)\kappa$		$(\alpha+\kappa r)t_i\tau$		$(\alpha+\kappa r)t_is_2$		$(\alpha+\kappa r)(\kappa\lambda_i+t_\pi s_3)$		$(\alpha+\kappa r)\cdot$	$\frac{\alpha+\kappa r}{b_1}(\lambda_i+t_\rho r')$		$\frac{\alpha+\kappa r}{b_1}(\kappa\psi_i+t_\rho r')$	
α	$(\alpha+\kappa r)t_i$		$(\alpha+\kappa r)t_ir$		$(\alpha+\kappa r)(x+\kappa\tau)$		$(\alpha+\kappa r)t_i\tau_1s_3$		$(x+\kappa\tau_1)s_3$	$\frac{\alpha+\kappa r}{b_2}(\lambda_i+t_\rho r')$		$\frac{\alpha+\kappa r}{b_2}(\kappa\psi_i+t_\rho r')$	
$b_1\tau_1s'_3$	$b_1\tau_1s'_3\kappa$		$b_1\tau_1s'_3t_i\tau$		$b_1\tau_1s'_3t_is_2$		$b_1\tau_1s'_3(\kappa\lambda_i+t_\pi s_3)$		$b_1\tau_1s'_3\cdot$	$\tau_1s'_3(\lambda_i+t_\rho r')$		$\tau_1s'_3(\kappa\psi_i+t_\rho r')$	
$x\mu$	$b_1\tau_1s'_3t_i$		$b_1\tau_1s'_3t_ir$		$b_1\tau_1s'_3(x+\kappa\tau)$		$b_1\tau_1^2s'_3t_is_3$		$(x+\kappa\tau_1)s_3$	$\frac{b_1\tau_1s'_3}{b_2}(\lambda_i+t_\rho r')$		$\frac{b_1\tau_1s'_3}{b_2}(\kappa\psi_i+t_\rho r')$	
$b_2\tau_1s''_3$	$b_2\tau_1s''_3\kappa$		$b_2\tau_1s''_3t_i\tau$		$b_2\tau_1s''_3t_is_2$		$b_2\tau_1s''_3(\kappa\lambda_i+t_\pi s_3)$		$b_2\tau_1s''_3\cdot$	$\frac{b_2\tau_1s''_3}{b_1}(\lambda_i+t_\rho r')$		$\frac{b_2\tau_1s''_3}{b_1}(\kappa\psi_i+t_\rho r')$	
$\alpha s+x\mu s_3$	$b_2\tau_1s''_3t_i$		$b_2\tau_1s''_3t_ir$		$b_2\tau_1s''_3(x+\kappa\tau)$		$b_2\tau_1s''_3t_i\tau_1s_3$		$(x+\kappa\tau_1)s_3$	$\tau_1s''_3(\lambda_i+t_\rho r')$		$\tau_1s''_3(\kappa\psi_i+t_\rho r')$	

TABLE 5: Pairing elements in $\mathbb{G}_T$ for the confidentiality (anonymity) proof of FEME
(t_π denotes $t_{\pi(i)}$, and t_ρ denotes $t_{\rho(i)}$)

hybrid argument. In the confidentiality game, $\mathcal{C}$ generates a challenge ciphertext with a component ct_0, which is either $\hat{H}(V) \oplus \phi(\mathsf{msg}_0^*)$ or $\hat{H}(V) \oplus \phi(\mathsf{msg}_1^*)$, where $V = e(g_1, g_2)^{\alpha s + x\mu s_3}$ and $s = s_1 + s_2$. Alternatively, We consider a modified game where V is either $e(g_1, g_2)^{\alpha s + x\mu s_3}$ or $e(g_1, g_2)^{\theta}$, with θ randomly chosen from $\mathbb{Z}_p^*$. We show that the adversary $\mathcal{A}_1$ in the original game can be reduced to an adversary $\mathcal{A}_2$ in the modified game. Since no $\mathcal{A}_2$ has a non-negligible advantage, it implies that $\mathcal{A}_1$ cannot either.

Given that $\mathcal{A}_1$ has an advantage ϵ in the original game, we can construct $\mathcal{A}_2$ as follows. During the challenge phase, after receiving msg_0 and msg_1 from $\mathcal{A}_1$ and V (which is either $V^{(1)} = e(g_1, g_2)^{\alpha s + x\mu s_3}$ or $V^{(2)} = e(g_1, g_2)^{\theta}$ from the challenger $\mathcal{C}$, $\mathcal{A}_2$ flips a coin $\beta \in \{0, 1\}$ and sends $\hat{H}(V) \oplus \phi(\mathsf{msg}_\beta)$ to $\mathcal{A}_1$. Once $\mathcal{A}_1$ outputs a bit β', $\mathcal{A}_2$ outputs 1 if $\beta' = \beta$, or 0 otherwise.

If $V = V^{(1)} = e(g_1, g_2)^{\alpha s + x\mu s_3}$, the challenge is a well-formed FEME ciphertext, and $\mathcal{A}_1$ has an advantage ϵ in correctly guessing $\beta' = \beta$. If $V = V^{(2)} = e(g_1, g_2)^{\theta}$, the challenge becomes independent of msg_0 and msg_1, giving $\mathcal{A}_2$ an advantage of 0. Consequently, we have

$$\begin{aligned} \Pr[\mathcal{A}_2 \text{ win}] &= \Pr[V = V^{(1)}] \cdot \Pr[\beta' = \beta | V^{(1)}] \\ &\quad + \Pr[V = V^{(2)}] \cdot \Pr[\beta' = \beta | V = V^{(2)}] \\ &\leqslant 1/2 \cdot (1/2 + \epsilon) + 1/2 \cdot 1/2 = 1/2 + \epsilon/2, \end{aligned}$$

and the overall advantage of $\mathcal{A}_2$ is $\epsilon/2$. The existence of any successful $\mathcal{A}_1$ implies the existence of a corresponding $\mathcal{A}_2$ with a non-negligible advantage.

Finally, we prove that no such $\mathcal{A}_2$ can distinguish between $e(g_1, g_2)^{\alpha s + x\mu s_3}$ and $e(g_1, g_2)^{\theta}$ in polynomial time. The combination of these results shows that no $\mathcal{A}_1$ can have a non-negligible advantage.

Simulation of the Modified Game. Let $g_1 = \sigma_1(1)$, $g_2 = \sigma_2(1)$ and $e(g_1, g_2) = \sigma_T(1)$. We write g_1^x to denote $\sigma_1(x)$, g_2^y to denote $\sigma_2(y)$ and $e(g_1, g_2)^z$ to denote $\sigma_T(z)$.

• ***Setup***. The challenger $\mathcal{C}$ chooses $\alpha, x, \mu, b_1, b_2, \kappa \xleftarrow{\$} \mathbb{Z}_p^*$, and calculates $Z = e(g_1, g_2)^{\alpha}$, $Y = e(g_1, g_2)^{x\mu}$, $\delta_0 = g_2^{\mu}$, $\delta_1 = g_2^{b_1}$, $\delta_2 = g_2^{b_2}$ and $h = g_1^{\kappa}$. $\mathcal{C}$ sends the master public key $\mathsf{mpk} = (Z, Y, h, \delta_0, \delta_1, \delta_2)$ to $\mathcal{A}$.

• ***Phase 1***. In phase 1, $\mathcal{A}$ can make oracle queries to the random oracle and a key generation oracle as follows.

– *Random oracle* $(\mathcal{O}_H)$. Same as defined above.

– *Attribute encryption key generation oracle* $(\mathcal{O}_{\mathsf{EKGen}})$. When $\mathcal{A}$ makes a key query for an attribute set $\mathcal{S}_{\mathsf{snd}}$, $\mathcal{C}$ picks $\tau \xleftarrow{\$} \mathbb{Z}_p^*$. Then, $\mathcal{C}$ computes

$$\mathsf{ek}_{1,i} = g_1^{t_i\tau}, \quad \mathsf{ek}_2 = g_2^{b_1\tau}, \quad \mathsf{ek}_3 = g_2^{b_2\tau}, \quad \mathsf{ek}_4 = g_1^{x+\kappa\tau}.$$

Then, $\mathcal{C}$ sends to $\mathcal{A}$ the attribute encryption key $\mathsf{EK}_{\mathcal{S}_{\mathsf{snd}}} = (\{n_i\}_{i\in[\ell_1]}, \{\mathsf{ek}_{1,i}\}_{i\in[\ell_1]}, \mathsf{ek}_2, \mathsf{ek}_3, \mathsf{ek}_4)$.

– *Attribute decryption key generation oracle* $(\mathcal{O}_{\mathsf{DKGen}})$. When $\mathcal{A}$ makes a key query for an attribute set $\mathcal{S}_{\mathsf{rcv}}$, $\mathcal{C}$ picks $r \xleftarrow{\$} \mathbb{Z}_p^*$. Then, $\mathcal{C}$ generates the attribute decryption key as

$$\mathsf{dk}_1 = g_1^{\alpha+\kappa r}, \quad \mathsf{dk}_{2,i} = g_1^{t_i r}, \quad \mathsf{dk}_3 = g_2^{r}.$$

Then, $\mathcal{C}$ sends to $\mathcal{A}$ the attribute decryption key $\mathsf{DK}_{\mathcal{S}_{\mathsf{rcv}}} = (\{n_i\}_{i\in[\ell_2]}, \mathsf{dk}_1, \{\mathsf{dk}_{2,i}\}_{i\in[\ell_2]}, \mathsf{dk}_3)$.

– *Policy decryption key generation oracle* $(\mathcal{O}_{\mathsf{PolGen}})$. When $\mathcal{A}$ makes a key query for a policy $\mathbb{A}_{\mathsf{rcv}} = (\mathbf{A}, \rho, \{\Psi_{\rho(i)}\}_{i\in[m_2]})$, $\mathcal{C}$ picks $r' \xleftarrow{\$} \mathbb{Z}_p^*$ and a vector $\mathbf{y} \xleftarrow{\$} \mathbb{Z}_p^{m_2-1}$. Let $\lambda_i = \mathbf{A}_i(\alpha||\mathbf{y})^\top$ and $\psi_i = \mathbf{A}_i(\mu||\mathbf{y})^\top$. Note that the λ_i (resp. ψ_i) are chosen uniformly and independently at random from $\mathbb{Z}_p^*$ subject to the random distribution of α (resp. μ) and $\mathbf{y}$. Then, $\mathcal{C}$ generates the policy decryption key as

$$\mathsf{sk}_1 = g_2^{r'}, \quad \mathsf{sk}_{2,i} = g_1^{\frac{1}{b_1}(\lambda_i + t_{\rho(i)}r')}, \quad \mathsf{sk}_{3,i} = g_1^{\frac{1}{b_2}(\lambda_i + t_{\rho(i)}r')},$$

$$\mathsf{sk}_{4,i} = g_1^{\frac{1}{b_1}(\kappa\psi_i + t_{\rho(i)}r')}, \quad \mathsf{sk}_{5,i} = g_1^{\frac{1}{b_2}(\kappa\psi_i + t_{\rho(i)}r')}.$$

Then, $\mathcal{C}$ sends to $\mathcal{A}$ the policy decryption key $\mathsf{SK}_{\mathbb{A}_{\mathsf{rcv}}} = ((\mathbf{A}, \rho, \{n_{\rho(i)}\}_{i\in[m_2]}), \mathsf{sk}_1, \{\mathsf{sk}_{2,i}, \mathsf{sk}_{3,i}, \mathsf{sk}_{4,i}, \mathsf{sk}_{5,i}\}_{i\in[m_2]})$.

• ***Challenge***. $\mathcal{A}$ outputs the sender's attribute sets $\mathcal{S}^*_{\mathsf{snd}}$, a policy $\mathbb{A}^*_{\mathsf{snd}}$ and two messages $\mathsf{msg}^*_0, \mathsf{msg}^*_1$ of equal length that it intends to challenge. $\mathcal{C}$ checks if $\mathcal{S}^*_{\mathsf{snd}}$ satisfies any of the access policy $\mathbb{A}_{\mathsf{rcv}}$ queried in Phase 1. If yes, $\mathcal{C}$ aborts. Otherwise, $\mathcal{C}$ chooses $s_1, s'_2, s''_2, s'_3, s''_3, \tau' \xleftarrow{\$} \mathbb{Z}^*_p$ and sets $s_2 = s'_2 + s''_2$, $s_3 = s'_3 + s''_3$, $s = s_1 + s_2$. Then $\mathcal{C}$ selects $\lambda_1, \cdots, \lambda_{m_1} \xleftarrow{\$} \mathbb{Z}^*_p$ for encryption of $\mathcal{S}^*_{\mathsf{snd}}$. The challenger selects $\theta \xleftarrow{\$} \mathbb{Z}^*_p$. The challenger flips random coin $b \in \{0,1\}$ to encrypt msg^*_b, and random coin $\beta \in \{0,1\}$ to determine which of the following ciphertext should be created.

If $\beta = 0$, it generates the challenge ciphertext as follows:

$$V = e(g_1, g_2)^{\alpha s + x\mu s_3},\ \mathsf{ct}_0 = \phi(\mathsf{msg}^*_b) \oplus \hat{H}(V),$$

$$\mathsf{ct}_1 = g_2^{s_1},\ \mathsf{ct}_2 = g_2^{s_3},$$

$$\mathsf{ct}_{3,i} = g_1^{\kappa\lambda_i + t_{\pi(i)}s_3},\ \mathsf{ct}_{4,1} = g_2^{b_1 s'_2},\ \mathsf{ct}_{4,2} = g_2^{b_2 s''_2},$$

$$\mathsf{ct}_{5,i} = g_1^{t_i s_2},\ \mathsf{ct}_{6,i} = g_1^{t_i \tau_1 s_3},\ \mathsf{ct}_7 = g_2^{b_1 \tau_1 s'_3},$$

$$\mathsf{ct}_8 = g_2^{b_2 \tau_1 s''_3},\ \mathsf{ct}_9 = g_1^{(x+\kappa\tau_1)s_3},\ \text{where } \tau_1 = \tau + \tau'.$$

Otherwise, it generates $V = e(g_1, g_2)^{\theta}$, and the other ciphertext components are kept the same.

Then, $\mathcal{C}$ sends to adversary $\mathcal{A}$ the ciphertext $\mathsf{CT}_{\mathsf{snd}} = ((\mathbf{M}, \pi, \{n_{\pi(i)}\}_{i\in[m_1]}), \mathsf{ct}_0, \mathsf{ct}_1, \mathsf{ct}_2, (\mathsf{ct}_{3,i})_{i\in[m_1]}, \mathsf{ct}_{4,1}, \mathsf{ct}_{4,2}, \{\mathsf{ct}_{5,i}, \mathsf{ct}_{6,i}\}_{i\in[\ell_1]}, \mathsf{ct}_7, \mathsf{ct}_8, \mathsf{ct}_9)$.

• ***Phase 2***. It is the same as in Phase 1 with the restriction that any input access policy $\mathbb{A}$ are not allowed to satisfy the challenge attribute sets $\mathcal{S}^*_{\mathsf{snd}}$.

• ***Guess***. $\mathcal{A}$ outputs a bit as a guess.

Analysis of $\mathcal{A}$'s Success Probability. To demonstrate that no PPT adversary $\mathcal{A}$ can distinguish between ct_0 in the aforementioned game, we assume the contrary. The only way $\mathcal{A}$'s views could differ is if there exist two distinct terms yielding the same result when $\theta = \delta(\alpha s + x\mu s_3)$ but producing different results when θ is sampled randomly. Let θ_1 and θ_2 be two such terms. Since θ only occurs in $e(g_1, g_2)^{\theta}$, which cannot be paired, $\mathcal{A}$ can only create queries where θ is an additive term. Thus, θ_1 and θ_2 can be written as $\theta_1 = \delta\theta + \theta'_1$ and $\theta_2 = \delta\theta + \theta'_2$ for some θ'_1, θ'_2 that do not contain θ. Based on the assumption that $\theta_1 = \theta_2$ when $\theta = \delta(\alpha s + x\mu s_3)$, we have $\delta_1(\alpha s + x\mu s_3) + \theta'_1 = \delta_2(\alpha s + x\mu s_3) + \theta'_2$. Rearranging this equation gives $\theta'_1 - \theta'_2 = (\delta_2 - \delta_1)(\alpha s + x\mu s_3)$, implying that $\mathcal{A}$ can algebraically construct $e(g_1, g_2)^{\delta(\alpha s + x\mu s_3)}$ for some $\delta \in \mathbb{Z}_q$ using the oracle outputs it has already queried. Below, we show that constructing such an expression is computationally infeasible, giving $\mathcal{A}$ only a negligible advantage in winning the confidentiality game.

To calculate the probability of $\mathcal{A}$ constructing $e(g_1, g_2)^{\delta(\alpha s + x\mu s_3)}$ for some $\delta \in \mathbb{Z}_q$, we perform a case analysis based on the information $\mathcal{A}$ receives from the simulation. For completeness, we first summarize the exponent elements available to $\mathcal{A}$ in groups $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$.

– $\mathbb{G}_1$ elements: 1, κ, t_i, $t_i\tau$, $t_i r$, $t_i s_2$, $x + \kappa\tau$, $\kappa\lambda_i + t_{\pi(i)}s_3$, $t_i\tau_1 s_3$, $(x + \kappa\tau_1)s_3$, $\frac{1}{b_1}(\lambda_i + t_{\rho(i)}r')$, $\frac{1}{b_2}(\lambda_i + t_{\rho(i)}r')$, $\frac{1}{b_1}(\kappa\psi_i + t_{\rho(i)}r')$, $\frac{1}{b_2}(\kappa\psi_i + t_{\rho(i)}r')$.

– $\mathbb{G}_2$ elements: 1, μ, b_1, b_2, r, r', s_1, s_3, $b_1\tau$, $b_2\tau$, $b_1 s'_2$, $b_2 s''_2$, $\alpha + \kappa r$, $b_1\tau_1 s'_3$, $b_2\tau_1 s''_3$.

– $\mathbb{G}_T$ elements: 1, α, $x\mu$.

We now enumerate all possible queries into $\mathbb{G}_T$ using the bilinear map and the group elements available to $\mathcal{A}$, as shown in Table 5. Note that the blue element $\alpha s + x\mu s_3$ in Table 5 is not used in this proof.

$\mathcal{A}$ can compute arbitrary linear combinations of these terms, and we will demonstrate that none can take the form $\delta(\alpha s + x\mu s_3)$, where $s = s_1 + s_2$.

(1) Consider how to construct $e(g_1, g_2)^{\delta\alpha s_1}$ for some δ. From Table 5, the only possibility is that $\mathcal{A}$ could create $\lambda_i s_1$ using terms such as $s_1(\kappa\lambda_i + t_{\pi(i)}s_3)$, $s_1\kappa$, $s_1 t_i$, $s_1 t_i \tau$, $s_1 t_i r$, $s_1 t_i s_2$, $s_1(x + \kappa\tau)$, $s_1 t_i \tau_1 s_3$, and $s_1(x + \kappa\tau_1)s_3$. However, $\lambda_i s_1$ cannot be combined through addition or subtraction with any other elements in $\mathbb{G}_1$, $\mathbb{G}_2$, or $\mathbb{G}_T$. Hence, constructing $\delta\alpha s_1$ in $\mathbb{G}_T$ is impossible for $\mathcal{A}$.

(2) Now, consider constructing $e(g_1, g_2)^{\delta\alpha s_2}$ for some δ, where $s_2 = s'_2 + s''_2$. According to Table 5, $\mathcal{A}$ cannot combine $s_2 = s'_2 + s''_2$ via simple addition or subtraction within the elements of $\mathbb{G}_1$, $\mathbb{G}_2$, or $\mathbb{G}_T$. The only way $\mathcal{A}$ could generate a term involving αs_2 is by pairing $\frac{1}{b_1} \cdot (\lambda_i + t_{\rho(i)}r')$ with $b_1\tau_1 s'_3$ and pairing $\frac{1}{b_2} \cdot (\lambda_i + t_{\rho(i)}r')$ with $b_2\tau_1 s''_3$, producing $(\lambda_i + t_{\rho(i)}r')s'_2$ and $(\lambda_i + t_{\rho(i)}r')s''_2$, which combine to form $(\lambda_i + t_{\rho(i)}r')(s'_2 + s''_2) = \lambda_i s_2 + t_{\rho(i)}r's_2$ in $\mathbb{G}_T$.

For $\mathcal{A}$ to construct $\delta\alpha s_2$ in $\mathbb{G}_T$, it must first construct $t_{\rho(i)}r's_2$ and then cancel out the term of $\lambda_i s_2$. From Table 5, $\mathcal{A}$ can deduce $t_{\rho(i)}r's_2$ by pairing $t_{\rho(i)}s_2$ (derived from $t_i s_2$) with r'. However, canceling $\lambda_i s_2$ by reconstructing λ_i as α is impossible. The reason is that the input access policy $\mathbb{A}$ cannot be satisfied by the attribute sets $\mathcal{S}^*$. Therefore, constructing $\delta\alpha s_2$ in $\mathbb{G}_T$ is infeasible for $\mathcal{A}$.

(3) Finally, consider constructing $e(g_1, g_2)^{\delta x\mu s_3}$ for some δ, where $s_3 = s'_3 + s''_3$. As shown in Table 5, combining $s_3 = s'_3 + s''_3$ through addition or subtraction within the elements of $\mathbb{G}_1$, $\mathbb{G}_2$, or $\mathbb{G}_T$ is impossible. The only way $\mathcal{A}$ could generate a term involving xs_3 is by pairing $\frac{1}{b_1} \cdot (\kappa\psi_i + t_{\rho(i)}r')$ with $b_1\tau_1 s'_3$ and pairing $\frac{1}{b_2} \cdot (\kappa\psi_i + t_{\rho(i)}r')$ with $b_2\tau_1 s''_3$, yielding $(\kappa\psi_i + t_{\rho(i)}r')\tau_1 s'_3$ and $(\kappa\psi_i + t_{\rho(i)}r')\tau_1 s''_3$, which combine to produce $(\kappa\psi_i + t_{\rho(i)}r')\tau_1 s_3$ in $\mathbb{G}_T$.

Thus, $\mathcal{A}$ can only create $x\psi_i s_3$ from terms such as $(\kappa\psi_i + t_{\rho(i)}r')\tau_1 s_3$, $s_3\kappa$, $s_3 t_i$, and $s_3(x + \kappa\tau)$. However, combining $x\psi_i s_3$ via addition or subtraction within the elements of $\mathbb{G}_1$, $\mathbb{G}_2$, or $\mathbb{G}_T$ is impossible. Therefore, constructing $\delta x\mu s_3$ in $\mathbb{G}_T$ is infeasible for $\mathcal{A}$.

Analysis of Simulation Failure. Let n represent the total number of group elements $\mathcal{A}$ receives from its oracle queries to the hash function, groups $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$, the bilinear map e, and its interaction with the confidentiality game. We show that the views of $\mathcal{A}$ for $\beta = 0$ and $\beta = 1$ are identically distributed, except with probability $O(n^2/q)$, based on the randomness in the variable values chosen during the simulation. This probability arises from an accidental collision, where two distinct polynomials in the GGM evaluate to the same value. As indicated in Table 5, the polynomial's maximum degree is 5. By the Schwartz-Zippel lemma [22], [34], the probability of such a collision occurring is $O(1/q)$.

Using a union bound, the probability of a collision across all n queries is at most $O(n^2/q)$, which is negligible when q is exponentially large in the secret parameter κ.

In conclusion, $\mathcal{A}$ only holds a negligible advantage in the modified game, implying that it also has a negligible advantage in the confidentiality game.

This completes the proof of Theorem 1. ■

C.2. Proof of Theorem 2 (FEME: Anonymity)

Proof. The logic flow of this proof is similar to the one for the Theorem 1. Firstly, we consider a modified security game that implies anonymity. Next, we bound $\mathcal{A}$'s success probability and analyze the simulation failure.

In the anonymity game, the ciphertext components related to the two challenged attribute sets and the two challenged access policies are $\mathsf{ct}_{3,i} = h^{\mathbf{M}_i(s_1||\mathbf{v})^\top} \cdot H(\Psi_{\pi(i)})^{s_3}$, $\mathsf{ct}_{5,i} = H(u_i)^{s_2}$ and $\mathsf{ct}_{6,i} = H(u_i)^{\tau_1}$, where $\tau_1 = \tau + \tau'$. Similarly, we can simulate them as $\mathsf{ct}_{3,i} = h^{\mathbf{M}_i(s_1||\mathbf{v})^\top} \cdot H(\Psi_{\pi(i)})^{s_3} = g_1^{\kappa\lambda_i + t_{\pi(i)}s_3}$, $\mathsf{ct}_{5,i} = H(u_i)^{s_2} = g_1^{t_i s_2}$ and $\mathsf{ct}_{6,i} = H(u_i)^{\tau_1} = g_1^{t_i \tau_1 s_3}$, where λ_i are chosen uniformly and independently at random from $\mathbb{Z}_p^*$ subject to the random distribution of s_1 and $\mathbf{v}$.

The challenger produces a challenge ciphertext element $\mathsf{ct}_{3,i}$ either $g_1^{\kappa\lambda_i + t_{\pi(i),0}s_3}$ or $g_1^{\kappa\lambda_i + t_{\pi(i),1}s_3}$ (given two challenge tag sets $\{t_{\pi(i),0}\}_i$, $\{t_{\pi(i),1}\}_i$, using $g_1^{t_{\pi(i),0}}$, $g_1^{t_{\pi(i),1}}$ as the querying results of $H(\Psi_{\pi(i)})$ from the random oracle, respectively), generates $\mathsf{ct}_{5,i}$ either $g_1^{t_{i,0}\cdot s_2}$ or $g_1^{t_{i,1}\cdot s_2}$, creates $\mathsf{ct}_{6,i}$ either $g_1^{t_{i,0}\cdot\tau_1 s_3}$ or $g_1^{t_{i,1}\cdot\tau_1 s_3}$ (given two challenge tag sets $\{t_{i,0}\}_i$, $\{t_{i,1}\}_i$, using $g_1^{t_{i,0}}$, $g_1^{t_{i,1}}$ as the querying results of $H(u_i)$ from the random oracle, respectively).

We can instead consider a modified experiment in which the challenger randomly samples a bit $\beta \xleftarrow{\$} \{0,1\}$ and $\theta_1, \theta_2, \theta_3 \xleftarrow{\$} \mathbb{Z}_p^*$. The generated challenge ciphertext element $\mathsf{ct}_{3,i}$ is either $g_1^{\kappa\lambda_i + t_{\pi(i),\beta}s_3}$ or $g_1^{\theta_1}$, the element $\mathsf{ct}_{5,i}$ is either $g_1^{t_{i,\beta}\cdot s_2}$ or $g_1^{\theta_2}$, and $\mathsf{ct}_{6,i}$ is either $g_1^{t_{i,\beta}\cdot\tau_1 s_3}$ or $g_1^{\theta_3}$.

Therefore, assume that $\mathcal{A}$ has an advantage ϵ in winning the anonymity game. Then, $\mathcal{A}$ has an advantage $\epsilon/2$ in distinguishing $g_1^{\kappa\lambda_i + t_{\pi(i),\beta}s_3}$ from $g_1^{\theta_1}$, an advantage $\epsilon/2$ in distinguishing $g_1^{t_{i,\beta}\cdot s_2}$ from $g_1^{\theta_2}$, and an advantage $\epsilon/2$ in distinguishing $g_1^{t_{i,\beta}\cdot\tau_1 s_3}$ from $g_1^{\theta_3}$. For instance, the probability of distinguishing $g_1^{t_{i,0}\cdot\tau_1 s_3}$ from $g_1^{\theta_3}$ is equal to that of distinguishing $g_1^{t_{i,1}\cdot\tau_1 s_3}$ from $g_1^{\theta_3}$. We call them the three tuples for simplicity in the following proof.

Simulation of the Modified Game. The adversary $\mathcal{A}$ aims to distinguish the three tuples.

• **Setup**. Same as defined in the proof of Theorem 1.

• **Phase 1**. In phase 1, $\mathcal{A}$ can make oracle queries to the random oracle and a key generation oracle as follows.

– *Random oracle* ($\mathcal{O}_H$). Same as defined above.

– *Key generation oracles* ($\mathcal{O}_{\mathsf{EKGen}}, \mathcal{O}_{\mathsf{DKGen}}, \mathcal{O}_{\mathsf{PolGen}}$). Same as defined in the proof of Theorem 1.

• **Challenge**. $\mathcal{A}$ outputs $\mathcal{S}_0^* = \{u_{i,0}\}_{i\in[m]} = \{n_i, v_{i,0}\}_{i\in[\ell_1]}$, $\mathcal{S}_1^* = \{u_{i,1}\}_{i\in[\ell_1]} = \{n_i, v_{i,1}\}_{i\in[\ell_1]}$, and two access policies $\mathbb{A}^*_{\mathsf{snd}_0} = (\mathbf{M}, \pi, \Psi_{\pi(i),0} = \{n_{\pi(i)}, v_{\pi(i),0}\}_{i\in[m_1]})$, $\mathbb{A}^*_{\mathsf{snd}_1} = (\mathbf{M}, \pi, \Psi_{\pi(i),1} = \{n_{\pi(i)}, v_{\pi(i),1}\}_{i\in[m_1]})$ that it intends to attack. Note that $\mathcal{S}_0^*, \mathcal{S}_1^*$ have the same attribute names $\{n_i\}_{i\in[\ell_1]}$, and $\mathbb{A}^*_{\mathsf{snd}_0}$, $\mathbb{A}^*_{\mathsf{snd}_1}$ have the same attribute names $\{n_{\pi(i)}\}_{i\in[m_1]}$. $\mathcal{C}$ checks whether $\mathcal{S}_0^*$ or $\mathcal{S}_1^*$ satisfies any of the access policy $\mathbb{A}$ queried in Phase 1. If yes, $\mathcal{C}$ rejects $\mathcal{S}_0^*$, $\mathcal{S}_1^*$. $\mathcal{C}$ also checks whether any of the attribute set $\mathcal{S}$ queried in Phase 1 satisfies $\mathbb{A}^*_{\mathsf{snd}_0}$ or $\mathbb{A}^*_{\mathsf{snd}_1}$. If yes, $\mathcal{C}$ rejects $\mathbb{A}^*_{\mathsf{snd}_0}$, $\mathbb{A}^*_{\mathsf{snd}_1}$. Otherwise, $\mathcal{C}$ chooses $\theta, s_1, s_2', s_2'', s_3', s_3'', \tau' \xleftarrow{\$} \mathbb{Z}_p^*$ and sets $s_2 = s_2' + s_2''$, $s_3 = s_3' + s_3''$, $s = s_1 + s_2$. Then $\mathcal{C}$ selects $\beta \xleftarrow{\$} \{0,1\}$ for encryption one set of attributes, and flips a coin $b \in \{0,1\}$.

If $b = 0$, it generates the challenge ciphertext as follows:

$$V = e(g_1, g_2)^{\alpha s + x\mu s_3},\ \mathsf{ct}_0 = \phi(\mathsf{msg}) \oplus \hat{H}(V),\ \mathsf{ct}_1 = g_2^{s_1},$$

$$\mathsf{ct}_2 = g_2^{s_3},\ \mathsf{ct}_{3,i} = g_1^{\kappa\lambda_i + t_{\pi(i),\beta}s_3},\ \mathsf{ct}_{4,1} = g_2^{b_1 s_2'},\ \mathsf{ct}_{4,2} = g_2^{b_2 s_2''},$$

$$\mathsf{ct}_{5,i} = g_1^{t_{i,\beta}s_2},\ \mathsf{ct}_{6,i} = g_1^{t_{i,\beta}\tau_1 s_3},\ \mathsf{ct}_7 = g_2^{b_1\tau_1 s_3'},$$

$$\mathsf{ct}_8 = g_2^{b_2\tau_1 s_3''},\ \mathsf{ct}_9 = g_1^{(x+\kappa\tau_1)s_3},\ \text{where } \tau_1 = \tau + \tau'.$$

Otherwise, it generates $\mathsf{ct}_{3,i} = g_1^{\theta_1}$, $\mathsf{ct}_{5,i} = g_1^{\theta_2}$, $\mathsf{ct}_{6,i} = g_1^{\theta_3}$, and the other ciphertext components are kept the same.

Then, $\mathcal{C}$ sends to adversary $\mathcal{A}$ the ciphertext $\mathsf{CT}_{\mathsf{snd}} = ((\mathbf{M}, \pi, \{n_{\pi(i)}\}_{i\in[m_1]}), \mathsf{ct}_0, \mathsf{ct}_1, \mathsf{ct}_2, (\mathsf{ct}_{3,i})_{i\in[m_1]}, \mathsf{ct}_{4,1}, \mathsf{ct}_{4,2}, \{\mathsf{ct}_{5,i}, \mathsf{ct}_{6,i}\}_{i\in[\ell_1]}, \mathsf{ct}_7, \mathsf{ct}_8, \mathsf{ct}_9)$.

• **Phase 2**. It is the same as in Phase 1 with the restriction that any input access policy $\mathbb{A}$ are not allowed to satisfy the challenge attribute sets $\mathcal{S}_0^*$ and $\mathcal{S}_1^*$.

• ***Guess***. $\mathcal{A}$ outputs a bit as a guess.

Analysis of $\mathcal{A}$'s Success Probability. For simplicity, we denote $t_{\pi(i),\beta}$ as $t_{\pi(i)}$, and $t_{i,\beta}$ as t_i in all further paragraphs. Suppose that $\mathcal{A}$ can algebraically construct $e(g_1,g_2)^{\delta_1(\kappa\lambda_i + t_{\pi(i)}s_3)}$, $e(g_1,g_2)^{\delta_2 t_i s_2}$, $e(g_1,g_2)^{\delta_3 t_i\tau_1 s_3}$ for some $\delta_1, \delta_2, \delta_3 \in \mathbb{Z}_q$ using all oracle outputs it has already queried, then it can use them to distinguish the three tuples.

To calculate the probability of $\mathcal{A}$ constructing $e(g_1,g_2)^{\delta_1(\kappa\lambda_i + t_{\pi(i)}s_3)}$, $e(g_1,g_2)^{\delta_2 t_i s_2}$, $e(g_1,g_2)^{\delta_3 t_i\tau_1 s_3}$ for some $\delta_1, \delta_2, \delta_3 \in \mathbb{Z}_q$, we perform a case analysis based on the information $\mathcal{A}$ receives from the simulation.

For completeness, we list all possible queries into $\mathbb{G}_T$ using the bilinear map and group elements available to $\mathcal{A}$, as shown in Table 5. The violet elements in Table 5 should be excluded for this anonymity proof, they are used in the confidentiality proof. $\mathcal{A}$ can compute arbitrary linear combinations of these terms. Below, we show that constructing these expressions is computationally infeasible, giving $\mathcal{A}$ only a negligible advantage in winning the anonymity game.

(1) Consider how to construct $e(g_1,g_2)^{\delta_1(\kappa\lambda_i + t_{\pi(i)}s_3)}$ for some δ_1. According to Table 5, $\mathcal{A}$ cannot combine $s_3 = s_3' + s_3''$ via simple addition or subtraction within the elements of $\mathbb{G}_1$, $\mathbb{G}_2$, or $\mathbb{G}_T$.

To generate a term involving $\kappa\lambda_i$ is by pairing $\frac{1}{b_1}\cdot(\lambda_i + t_{\rho(i)}r')$ with $b_1\tau_1 s_3'$ and pairing $\frac{1}{b_2}\cdot(\lambda_i + t_{\rho(i)}r')$ with $b_2\tau_1 s_3''$, yielding $(\lambda_i + t_{\rho(i)}r')\tau_1 s_3'$ and $(\lambda_i + t_{\rho(i)}r')\tau_1 s_3''$, which combine to produce $(\lambda_i + t_{\rho(i)}r')\tau_1 s_3$ in $\mathbb{G}_T$.

To generate a term involving $t_i s_3$ is by pairing $\frac{1}{b_1}\cdot(\kappa\psi_i + t_{\rho(i)}r')$ with $b_1\tau_1 s_3'$ and pairing $\frac{1}{b_2}\cdot(\kappa\psi_i + t_{\rho(i)}r')$ with

$b_2\tau_1 s_3''$, yielding $(\kappa\psi_i + t_{\rho(i)}r')\tau_1 s_3'$ and $(\kappa\psi_i + t_{\rho(i)}r')\tau_1 s_3''$, which combine to produce $(\kappa\psi_i + t_{\rho(i)}r')\tau_1 s_3$ in $\mathbb{G}_T$.

However, $\kappa\lambda_i + t_{\pi(i)}s_3$ cannot be derived through addition or subtraction of $(\lambda_i + t_{\rho(i)}r')\tau_1 s_3$, $(\kappa\psi_i + t_{\rho(i)}r')\tau_1 s_3$ with any other elements in $\mathbb{G}_1$, $\mathbb{G}_2$, or $\mathbb{G}_T$. Hence, constructing $\delta_1(\kappa\lambda_i + t_{\pi(i)}s_3)$ in $\mathbb{G}_T$ is impossible for $\mathcal{A}$.

(2) Now, consider constructing $e(g_1, g_2)^{\delta_2 t_i s_2}$ for some δ_2. According to Table 5, $\mathcal{A}$ cannot combine $s_2 = s_2' + s_2''$ via simple addition or subtraction within the elements of $\mathbb{G}_1$, $\mathbb{G}_2$, or $\mathbb{G}_T$.

The first way $\mathcal{A}$ could generate a term involving $t_i s_2$ is by pairing $\frac{1}{b_1} \cdot (\lambda_i + t_{\rho(i)}r')$ with $b_1 s_2'$ and pairing $\frac{1}{b_2} \cdot (\lambda_i + t_{\rho(i)}r')$ with $b_2 s_2''$, producing $(\lambda_i + t_{\rho(i)}r')s_2'$ and $(\lambda_i + t_{\rho(i)}r')s_2''$, which combine to $(\lambda_i + t_{\rho(i)}r')(s_2' + s_2'') = \lambda_i s_2 + r' t_{\rho(i)} s_2$ in $\mathbb{G}_T$. However, $t_i s_2$ cannot be derived through addition or subtraction of $\lambda_i s_2 + r' t_{\rho(i)} s_2$ with any other elements in $\mathbb{G}_1$, $\mathbb{G}_2$, or $\mathbb{G}_T$. Hence, constructing $\delta_2 t_i s_2$ in $\mathbb{G}_T$ is impossible for $\mathcal{A}$.

Another way $\mathcal{A}$ could generate a term involving $t_i s_2$ is by pairing $\frac{1}{b_1} \cdot (\kappa\psi_i + t_{\rho(i)}r')$ with $b_1 s_2'$ and pairing $\frac{1}{b_2} \cdot (\kappa\psi_i + t_{\rho(i)}r')$ with $b_2 s_2''$, producing $(\kappa\psi_i + t_{\rho(i)}r')s_2'$ and $(\kappa\psi_i + t_{\rho(i)}r')s_2''$, which combine to $(\kappa\psi_i + t_{\rho(i)}r')(s_2' + s_2'') = \kappa\psi_i s_2 + r' t_{\rho(i)} s_2$ in $\mathbb{G}_T$. However, $t_i s_2$ cannot be derived through addition or subtraction of $\kappa\psi_i s_2 + r' t_{\rho(i)} s_2$ with any other elements in $\mathbb{G}_1$, $\mathbb{G}_2$, or $\mathbb{G}_T$. Hence, constructing $\delta_2 t_i s_2$ in $\mathbb{G}_T$ is impossible for $\mathcal{A}$.

(3) Finally, consider constructing $e(g_1, g_2)^{\delta_3 t_i \tau_1 s_3}$ for some δ_3. According to Table 5, $\mathcal{A}$ cannot combine $s_3 = s_3' + s_3''$ via simple addition or subtraction within the elements of $\mathbb{G}_1$, $\mathbb{G}_2$, or $\mathbb{G}_T$.

The first way $\mathcal{A}$ could generate a term involving $t_i \tau_1 s_3$ is by pairing $\frac{1}{b_1} \cdot (\lambda_i + t_{\rho(i)}r')$ with $b_1 \tau_1 s_3'$ and pairing $\frac{1}{b_2} \cdot (\lambda_i + t_{\rho(i)}r')$ with $b_2 \tau_1 s_3''$, yielding $(\lambda_i + t_{\rho(i)}r')\tau_1 s_3'$ and $(\lambda_i + t_{\rho(i)}r')\tau_1 s_3''$, which combine to produce $(\lambda_i + t_{\rho(i)}r')\tau_1 s_3 = \lambda_i \cdot \tau_1 s_3 + r' \cdot t_i \tau_1 s_3$ in $\mathbb{G}_T$. However, $t_i \tau_1 s_3$ cannot be derived through addition or subtraction of $\lambda_i \cdot \tau_1 s_3 + r' \cdot t_{\rho(i)} \tau_1 s_3$ with any other elements in $\mathbb{G}_1$, $\mathbb{G}_2$, or $\mathbb{G}_T$. Hence, constructing $\delta_3 t_i \tau_1 s_3$ in $\mathbb{G}_T$ is impossible for $\mathcal{A}$.

Another way $\mathcal{A}$ could generate a term involving $t_i \tau_1 s_3$ is by pairing $\frac{1}{b_1} \cdot (\kappa\psi_i + t_{\rho(i)}r')$ with $b_1 \tau_1 s_3'$ and pairing $\frac{1}{b_2} \cdot (\kappa\psi_i + t_{\rho(i)}r')$ with $b_2 \tau_1 s_3''$, yielding $(\kappa\psi_i + t_{\rho(i)}r')\tau_1 s_3'$ and $(\kappa\psi_i + t_{\rho(i)}r')\tau_1 s_3''$, which combine to produce $(\kappa\psi_i + t_{\rho(i)}r')\tau_1 s_3$ in $\mathbb{G}_T$. However, $t_i \tau_1 s_3$ cannot be derived through addition or subtraction of $(\kappa\psi_i + t_{\rho(i)}r')\tau_1 s_3$ with any other elements in $\mathbb{G}_1$, $\mathbb{G}_2$, or $\mathbb{G}_T$. Hence, constructing $\delta_3 t_i \tau_1 s_3$ in $\mathbb{G}_T$ is impossible for $\mathcal{A}$.

Analysis of Simulation Failure. Same as that in the proof for Theorem 1.

In conclusion, $\mathcal{A}$ only holds a negligible advantage in the modified game, implying that it also has a negligible advantage in the anonymity game.

This completes the proof of Theorem 2. ■

C.3. Proof of Theorem 3 (FEME: Authenticity)

To prove the authenticity of FEME, we show that an adversary cannot create a valid ciphertext that passes decryption without the proper attribute encryption keys. The challenger sets up the system, providing the adversary with master public keys and simulating key generation oracles. The adversary can query these oracles to obtain encryption and decryption keys for chosen attribute sets and policies, except for those matching the final target policy. The proof relies on a contradiction: assuming the adversary produces a valid ciphertext under conditions not allowed by the oracles, we demonstrate this would imply the adversary must have queried the sender's encryption key with attributes matching the target policy, violating the security model's constraints. This involves showing that certain polynomial relations between group elements must hold, which leads to the contradiction, proving that any valid forgery is computationally infeasible.

Proof. The interaction between the adversary $\mathcal{A}$ and challenger $\mathcal{C}$ proceeds as follows.

• ***Setup***. The challenger $\mathcal{C}$ chooses $\alpha, x, \mu, b_1, b_2, \kappa \xleftarrow{\$} \mathbb{Z}_p^*$, and calculates $Z = e(g_1, g_2)^{\alpha}$, $Y = e(g_1, g_2)^{x\mu}$, $\delta_0 = g_2^{\mu}$, $\delta_1 = g_2^{b_1}$, $\delta_2 = g_2^{b_2}$ and $h = g_1^{\kappa}$. $\mathcal{C}$ sends the master public key $\mathsf{mpk} = (Z, Y, h, \delta_0, \delta_1, \delta_2)$ to $\mathcal{A}$.

• ***Phase 1***. In phase 1, $\mathcal{A}$ can make oracle queries to the random oracle and a key generation oracle as follows.

− *Random oracle* ($\mathcal{O}_H$). Same as defined above.

− *Attribute encryption key generation oracle* ($\mathcal{O}_{\mathsf{EKGen}}$). When $\mathcal{A}$ makes a encryption key query for an attribute set $\mathcal{S}_{\mathsf{snd}}$, $\mathcal{C}$ picks $\tau_j \xleftarrow{\$} \mathbb{Z}_p^*$ for the j-th query. Then, $\mathcal{C}$ generates the attribute encryption key as

$$\mathsf{ek}_{1,i} = h_i^{\tau_j}, \quad \mathsf{ek}_2 = \delta_1^{\tau_j}, \quad \mathsf{ek}_3 = \delta_2^{\tau_j}, \quad \mathsf{ek}_4 = g_1^{x} \cdot h^{\tau_j}.$$

Then, $\mathcal{C}$ sends to $\mathcal{A}$ the attribute encryption key $\mathsf{EK}_{\mathcal{S}_{\mathsf{snd}}} = (\{n_i\}_{i\in[\ell_1]}, \{\mathsf{ek}_{1,i}\}_{i\in[\ell_1]}, \mathsf{ek}_2, \mathsf{ek}_3, \mathsf{ek}_4)$.

− *Attribute decryption key generation oracle* ($\mathcal{O}_{\mathsf{DKGen}}$). When $\mathcal{A}$ makes a key query for an attribute set $\mathcal{S}_{\mathsf{rcv}}$, $\mathcal{C}$ picks $r \xleftarrow{\$} \mathbb{Z}_p^*$. Then, $\mathcal{C}$ generates the attribute decryption key as

$$\mathsf{dk}_1 = g_1^{\alpha} \cdot h^{r}, \quad \mathsf{dk}_{2,i} = h_i^{r}, \quad \mathsf{dk}_3 = g_2^{r}.$$

Then, $\mathcal{C}$ sends to $\mathcal{A}$ the attribute decryption key $\mathsf{DK}_{\mathcal{S}_{\mathsf{rcv}}} = (\{n_i\}_{i\in[\ell_2]}, \mathsf{dk}_1, \{\mathsf{dk}_{2,i}\}_{i\in[\ell_2]}, \mathsf{dk}_3)$.

− *Policy decryption key generation oracle* ($\mathcal{O}_{\mathsf{PolGen}}$). When $\mathcal{A}$ makes a key query for a policy $\mathbb{A}_{\mathsf{rcv}} = (\mathbf{A}, \rho, \{\Psi_{\rho(i)}\}_{i\in[m_2]})$, $\mathcal{C}$ picks $r' \xleftarrow{\$} \mathbb{Z}_p^*$ and a vector $\mathbf{y} \xleftarrow{\$} \mathbb{Z}_p^{m_2-1}$. Let $\lambda_i = \mathbf{A}_i(\alpha||\mathbf{y})^{\top}$ and $\psi_i = \mathbf{A}_i(\mu||\mathbf{y})^{\top}$. Note that the λ_i (resp. ψ_i) are chosen uniformly and independently at random from $\mathbb{Z}_p^*$ subject to the random distribution of α (resp. μ) and $\mathbf{y}$. Then, $\mathcal{C}$ generates the policy decryption key as

$$\mathsf{sk}_1 = g_2^{r'}, \quad \mathsf{sk}_{2,i} = (g_1^{\lambda_i} \cdot h_{\rho(i)}^{r'})^{\frac{1}{b_1}}, \quad \mathsf{sk}_{3,i} = (g_1^{\lambda_i} \cdot h_{\rho(i)}^{r'})^{\frac{1}{b_2}},$$

$$\mathsf{sk}_{4,i} = (h^{\psi_i} \cdot h_{\rho(i)}^{r'})^{\frac{1}{b_1}}, \quad \mathsf{sk}_{5,i} = (h^{\psi_i} \cdot h_{\rho(i)}^{r'})^{\frac{1}{b_2}}.$$

Then, $\mathcal{C}$ sends to $\mathcal{A}$ the policy decryption key $\mathsf{SK}_{\mathbb{A}_{\mathsf{rcv}}} = ((\mathbf{A}, \rho, \{n_{\rho(i)}\}_{i\in[m_2]}), \mathsf{sk}_1, \{\mathsf{sk}_{2,i}, \mathsf{sk}_{3,i}, \mathsf{sk}_{4,i}, \mathsf{sk}_{5,i}\}_{i\in[m_2]})$.

• ***Forgery***. The attacker $\mathcal{A}$ sends $(\mathsf{CT}^*_{\mathsf{snd}}, \mathcal{S}^*_{\mathsf{snd}}, \mathbb{A}^*_{\mathsf{rcv}})$ to $\mathcal{C}$, where $\mathcal{S}^*_{\mathsf{snd}} \models \mathbb{A}^*_{\mathsf{rcv}}$. The restriction is that the attribute encryption key of $\mathcal{S}^*_{\mathsf{snd}}$ has not been queried in $\mathcal{O}_{\mathsf{EKGen}}$, and the policy decryption key of $\mathbb{A}^*_{\mathsf{rcv}}$ and any other $\mathbb{A}_{\mathsf{rcv}}$

satisfying $\mathcal{S}^*_{\mathsf{snd}} \models \mathbb{A}_{\mathsf{rcv}}$ have not been queried in $\mathcal{O}_{\mathsf{PolGen}}$. We must prove that under the restrictions, the adversary $\mathcal{A}$ cannot forge a valid ciphertext $\mathsf{CT}^*_{\mathsf{snd}}$.

We use proof by contradiction to assume that $\mathsf{CT}^*_{\mathsf{snd}}$ is a valid ciphertext. The adversary $\mathcal{A}$ has access to the group elements provided in the master public key $\mathsf{mpk} = (Z, Y, h, \delta_0, \delta_1, \delta_2)$, $h_i = H(u_i)$ obtained from $\mathcal{O}_H$, and $\mathsf{EK}_{\mathcal{S}_{\mathsf{snd}}}$ obtained from $\mathcal{O}_{\mathsf{EKGen}}$, as they are the input of Enc for $\mathsf{CT}^*_{\mathsf{snd}}$ generation. Let $\mathsf{EK}^{(j)}_{\mathcal{S}_{\mathsf{snd}}} = (\{n^{(j)}_i\}_{i\in[\ell_1]}, \{\mathsf{ek}^{(j)}_{1,i}\}_{i\in[\ell_1]}, \mathsf{ek}^{(j)}_2, \mathsf{ek}^{(j)}_3, \mathsf{ek}^{(j)}_4)$ be the output of the j-th query to $\mathcal{O}_{\mathsf{EKGen}}$. Denote q_e as the query time to $\mathcal{O}_{\mathsf{EKGen}}$. In GGM, the only way for the adversary to generate new group elements is to use the existing exponent elements available to $\mathcal{A}$ in groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$. This means that there are known scalars $\big(a, b, b', b'', c, c', c'', \{d_i\}_{i\in[m_1]}, e, \{f_{1,j}, f_{2,j}, f_{3,j}, f_{4,j}\}_{j\in[q_e]}\big)$ such that:

$$V = Z^{a+b} \cdot Y^c,\ \mathsf{ct}_0 = \phi(\mathsf{msg}) \oplus \hat{H}(V),$$

$$\mathsf{ct}^*_1 = g_2^a,\ \mathsf{ct}^*_2 = g_2^c,\ \mathsf{ct}^*_{3,i} = h^{d_i} \cdot (h_{\pi(i)})^c,$$

$$\mathsf{ct}^*_{4,1} = \delta_1^{b'},\ \mathsf{ct}^*_{4,2} = \delta_2^{b''},\ \mathsf{ct}^*_{5,i} = h_i^b,$$

$$\begin{aligned}
\mathsf{ct}^*_{6,i} &= \textstyle\prod_{j=1}^{q_e} (\mathsf{ek}^{(j)}_{1,i})^{f_{1,j}\cdot c} \cdot h_i^{e\cdot c},\\
\mathsf{ct}^*_7 &= \textstyle\prod_{j=1}^{q_e} (\mathsf{ek}^{(j)}_2)^{f_{2,j}\cdot c'} \cdot \delta_1^{e\cdot c'},\\
\mathsf{ct}^*_8 &= \textstyle\prod_{j=1}^{q_e} (\mathsf{ek}^{(j)}_3)^{f_{3,j}\cdot c''} \cdot \delta_2^{e\cdot c''},\\
\mathsf{ct}^*_9 &= \textstyle\prod_{j=1}^{q_e} (\mathsf{ek}^{(j)}_4)^{f_{4,j}\cdot c} \cdot h^{e\cdot c}.
\end{aligned}$$

In the ciphertext $\mathsf{CT}^*_{\mathsf{snd}}$ construction, it implicit sets that $a = s_1$, $b = s_2$, $c = s_3$, $b' = s'_2$, $b'' = s''_2$, $c' = s'_3$, $c'' = s''_3$, where $b = b' + b''$ and $c = c' + c''$. It also implies that $d_i = \mathbf{M}_i(s_1||\mathbf{v})^\top$ and $e = \tau'$.

Recall that $\mathsf{CT}^*_{\mathsf{snd}}$ is a valid ciphertext on $\mathcal{S}^*_{\mathsf{snd}}$ and $\mathcal{S}^*_{\mathsf{snd}} \models \mathbb{A}^*_{\mathsf{rcv}}$. According to the correctness proof (shown in the full version [20]), we know that:

$$\frac{e(\prod_{i\in I_2}(\mathsf{sk}_{4,\rho(i)})^{\omega_i}, \mathsf{ct}_7)e(\prod_{i\in I_2}(\mathsf{sk}_{5,\rho(i)})^{\omega_i}, \mathsf{ct}_8)}{e(\mathsf{ct}_9, \delta_0)e(\prod_{i\in I_2}(\mathsf{ct}_{6,\rho(i)})^{\omega_i}, \mathsf{sk}_1)} = Y^{-s_3},$$

where $\sum_{i\in I_2}(\mathbf{A}_i(\mu||\mathbf{y})^\top \cdot \omega_i) = \mu$.

Based on this equation, we derive that

$$\begin{aligned}
&\frac{e(\prod_{i\in I_2}(\mathsf{sk}_{4,\rho(i)})^{\omega_i}, \mathsf{ct}_7)e(\prod_{i\in I_2}(\mathsf{sk}_{5,\rho(i)})^{\omega_i}, \mathsf{ct}_8)}{e(\mathsf{ct}_9, \delta_0)e(\prod_{i\in I_2}(\mathsf{ct}_{6,\rho(i)})^{\omega_i}, \mathsf{sk}_1)}\\
=\ &\frac{e(\prod_{i\in I_2}((h^{\psi_i}\cdot h^{r'}_{\rho(i)})^{\frac{1}{b_1}})^{\omega_i}, \prod_{j=1}^{q_e}(\mathsf{ek}^{(j)}_2)^{f_{2,j}c'}\cdot\delta_1^{e\cdot c'})}{e(\prod_{j=1}^{q_e}(\mathsf{ek}^{(j)}_4)^{f_{4,j}c}\cdot h^{e\cdot c}, \delta_0)}\\
&\cdot\frac{e(\prod_{i\in I_2}((h^{\psi_i}\cdot h^{r'}_{\rho(i)})^{\frac{1}{b_2}})^{\omega_i}, \prod_{j=1}^{q_e}(\mathsf{ek}^{(j)}_3)^{f_{3,j}c''}\cdot\delta_2^{e\cdot c''})}{e(\prod_{i\in I_2}(\prod_{j=1}^{q_e}(\mathsf{ek}^{(j)}_{1,\rho(i)})^{f_{1,j}c}\cdot h^{e\cdot c}_{\rho(i)})^{\omega_i}, g_2^{r'})}\\
=\ &\frac{e(\prod_{i\in I_2}(h^{\psi_i}\cdot h^{r'}_{\rho(i)})^{\frac{\omega_i}{b_1}}, \prod_{j=1}^{q_e}(\delta_1^{\tau_j})^{f_{2,j}c'}\cdot\delta_1^{e\cdot c'})}{e(\prod_{j=1}^{q_e}(g_1^x\cdot h^{\tau_j})^{f_{4,j}c}\cdot h^{e\cdot c}, g_2^{\mu})}\\
&\cdot\frac{e(\prod_{i\in I_2}(h^{\psi_i}\cdot h^{r'}_{\rho(i)})^{\frac{\omega_i}{b_2}}, \prod_{j=1}^{q_e}(\delta_2^{\tau_j})^{f_{3,j}c''}\cdot\delta_2^{e\cdot c''})}{e(\prod_{i\in I_2}(\prod_{j=1}^{q_e}(h^{\tau_j}_{\rho(i)})^{f_{1,j}c}\cdot h^{e\cdot c}_{\rho(i)})^{\omega_i}, g_2^{r'})}\\
=\ &\frac{e(g_1^{\frac{\kappa}{b_1}\sum_{i\in I_2}\psi_i\omega_i}\cdot g_1^{\frac{r'}{b_1}\sum_{i\in I_2}\omega_i t_{\rho(i)}}, g_2^{c'b_1(\sum_{j=1}^{q_e}f_{2,j}\tau_j+e)})}{e(g_1^{cx\cdot\sum_{j=1}^{q_e}f_{4,j}}\cdot g_1^{c(\sum_{j=1}^{q_e}f_{4,j}\tau_j+e)\kappa}, g_2^{\mu})}\\
&\cdot\frac{e(g_1^{\frac{\kappa}{b_2}\sum_{i\in I_2}\psi_i\omega_i}\cdot g_1^{\frac{r'}{b_2}\sum_{i\in I_2}\omega_i t_{\rho(i)}}, g_2^{c''b_2(\sum_{j=1}^{q_e}f_{3,j}\tau_j+e)})}{e(g_1^{c(\sum_{j=1}^{q_e}f_{1,j}\tau_j+e)\sum_{i\in I_2}\omega_i t_{\rho(i)}}, g_2^{r'})}\\
=\ &e(g_1, g_2)^{-x\mu\cdot c} = Y^{-c},
\end{aligned}$$

where $\sum_{i\in I_2}\psi_i\cdot\omega_i = \mu$, and $\psi_i = \mathbf{A}_i(\mu||\mathbf{y})^\top$.

Then, we obtain the following polynomial relation:

$$\begin{aligned}
&(\frac{\kappa}{b_1}\sum_{i\in I_2}\psi_i\omega_i + \frac{r'}{b_1}\sum_{i\in I_2}\omega_i t_{\rho(i)})\cdot(c'b_1(\textstyle\sum_{j=1}^{q_e}f_{2,j}\tau_j + e))\\
&+(\frac{\kappa}{b_2}\sum_{i\in I_2}\psi_i\omega_i + \frac{r'}{b_2}\sum_{i\in I_2}\omega_i t_{\rho(i)})\cdot(c''b_2(\textstyle\sum_{j=1}^{q_e}f_{3,j}\tau_j + e))\\
&-(cx\cdot\textstyle\sum_{j=1}^{q_e}f_{4,j} + c(\sum_{j=1}^{q_e}f_{4,j}\tau_j + e)\kappa)\cdot\mu\\
&-(c(\textstyle\sum_{j=1}^{q_e}f_{1,j}\tau_j + e)\sum_{i\in I_2}\omega_i t_{\rho(i)})\cdot r'\\
=\ &-x\mu c
\end{aligned}$$

Let us rearrange the left side of this equation as follows.

$$\begin{aligned}
&(\frac{\kappa}{b_1}\sum_{i\in I_2}\psi_i\omega_i + \frac{r'}{b_1}\sum_{i\in I_2}\omega_i t_{\rho(i)})\cdot(c'b_1(\textstyle\sum_{j=1}^{q_e}f_{2,j}\tau_j + e))\\
&+(\frac{\kappa}{b_2}\sum_{i\in I_2}\psi_i\omega_i + \frac{r'}{b_2}\sum_{i\in I_2}\omega_i t_{\rho(i)})\cdot(c''b_2(\textstyle\sum_{j=1}^{q_e}f_{3,j}\tau_j + e))\\
&-(cx\cdot\textstyle\sum_{j=1}^{q_e}f_{4,j} + c(\sum_{j=1}^{q_e}f_{4,j}\tau_j + e)\kappa)\cdot\mu\\
&-(c(\textstyle\sum_{j=1}^{q_e}f_{1,j}\tau_j + e)\sum_{i\in I_2}\omega_i t_{\rho(i)})\cdot r'\\
=\ &(\kappa c'\textstyle\sum_{j=1}^{q_e}f_{2,j}\tau_j(\displaystyle\sum_{i\in I_2}\psi_i\omega_i) + r'c'\textstyle\sum_{j=1}^{q_e}f_{2,j}\tau_j(\displaystyle\sum_{i\in I_2}\omega_i t_{\rho(i)})\\
&\quad+\kappa c'e\sum_{i\in I_2}\psi_i\omega_i + r'c'e\sum_{i\in I_2}\omega_i t_{\rho(i)})\\
&+(\kappa c''\textstyle\sum_{j=1}^{q_e}f_{3,j}\tau_j(\displaystyle\sum_{i\in I_2}\psi_i\omega_i) + r'c''\textstyle\sum_{j=1}^{q_e}f_{3,j}\tau_j(\displaystyle\sum_{i\in I_2}\omega_i t_{\rho(i)})\\
&\quad+\kappa c''e\sum_{i\in I_2}\psi_i\omega_i + r'c''e\sum_{i\in I_2}\omega_i t_{\rho(i)})\\
&-(\textstyle\sum_{j=1}^{q_e}f_{4,j}\cdot x\mu c + c(\sum_{j=1}^{q_e}f_{4,j}\tau_j + e)\kappa\cdot\mu)\\
&-(cr'(\textstyle\sum_{j=1}^{q_e}f_{1,j}\tau_j + e)\sum_{i\in I_2}\omega_i t_{\rho(i)})\\
=\ &\kappa(ce + c'\textstyle\sum_{j=1}^{q_e}f_{2,j}\tau_j + c''\sum_{j=1}^{q_e}f_{3,j}\tau_j)(\displaystyle\sum_{i\in I_2}\psi_i\omega_i)\\
&+r'(ce + c'\textstyle\sum_{j=1}^{q_e}f_{2,j}\tau_j + c''\sum_{j=1}^{q_e}f_{3,j}\tau_j)(\displaystyle\sum_{i\in I_2}\omega_i t_{\rho(i)})\\
&-(\textstyle\sum_{j=1}^{q_e}f_{4,j}\cdot x\mu c + \kappa\cdot c\mu(\sum_{j=1}^{q_e}f_{4,j}\tau_j + e))\\
&-(r'(c\textstyle\sum_{j=1}^{q_e}f_{1,j}\tau_j + ce)\sum_{i\in I_2}\omega_i t_{\rho(i)})\\
=\ &\kappa[(ce + c'\textstyle\sum_{j=1}^{q_e}f_{2,j}\tau_j + c''\sum_{j=1}^{q_e}f_{3,j}\tau_j)\mu\\
&\qquad-c\mu(\textstyle\sum_{j=1}^{q_e}f_{4,j}\tau_j + e))]\\
&+r'(-c\sum_{j=1}^{q_e}f_{1,j}\tau_j + c'\sum_{j=1}^{q_e}f_{2,j}\tau_j + c''\sum_{j=1}^{q_e}f_{3,j}\tau_j)(\sum_{i\in I_2}\omega_i t_{\rho(i)})\\
&-\textstyle\sum_{j=1}^{q_e}f_{4,j}\cdot x\mu c\\
=\ &(-c\textstyle\sum_{j=1}^{q_e}f_{4,j}\tau_j + c'\sum_{j=1}^{q_e}f_{2,j}\tau_j + c''\sum_{j=1}^{q_e}f_{3,j}\tau_j)\kappa\mu\\
&+r'(-c\sum_{j=1}^{q_e}f_{1,j}\tau_j + c'\sum_{j=1}^{q_e}f_{2,j}\tau_j + c''\sum_{j=1}^{q_e}f_{3,j}\tau_j)(\sum_{i\in I_2}\omega_i t_{\rho(i)})\\
&-\textstyle\sum_{j=1}^{q_e}f_{4,j}\cdot x\mu c
\end{aligned}$$

Since the right side of the equation is $-x\mu c$, we must have

$$-c\sum_{j=1}^{q_e} f_{4,j}\tau_j + c'\sum_{j=1}^{q_e} f_{2,j}\tau_j + c''\sum_{j=1}^{q_e} f_{3,j}\tau_j = 0, \quad (1)$$

$$-c\sum_{j=1}^{q_e} f_{1,j}\tau_j + c'\sum_{j=1}^{q_e} f_{2,j}\tau_j + c''\sum_{j=1}^{q_e} f_{3,j}\tau_j = 0, \quad (2)$$

$$\sum\nolimits_{j=1}^{q_e} f_{4,j} = 1. \quad (3)$$

Combining equations (1)-(3), we derive that

$$\sum\nolimits_{j=1}^{q_e} f_{1,j} = \sum\nolimits_{j=1}^{q_e} f_{4,j} = 1.$$

We arrange equation (1) as

$$\sum\nolimits_{j=1}^{q_e}(-cf_{4,j} + c'f_{2,j} + c''f_{3,j})\tau_j = 0.$$

According to linear algebraic theory, we deduce that $-cf_{4,j} + c'f_{2,j} + c''f_{3,j} = 0$ for all $j \in [q_e]$. As $c = c' + c''$, we have

$$\begin{aligned} -(c'+c'')f_{4,j} + c'f_{2,j} + c''f_{3,j} &= 0 \\ c'(f_{2,j} - f_{4,j}) + c''(f_{3,j} - f_{4,j}) &= 0 \end{aligned}$$

Then, we deduce that $f_{2,j} = f_{3,j} = f_{4,j}$ for all $j \in [q_e]$. Similarly, we deduce that $f_{1,j} = f_{2,j} = f_{3,j}$ for all $j \in [q_e]$ from equation (2). Then, $f_{1,j} = f_{2,j} = f_{3,j} = f_{4,j}$ for all $j \in [q_e]$. Under this situation, we have

$$\mathsf{EK}^{(j)}_{\mathcal{S}_{\mathsf{snd}}} = (\{n_i^{(j)}\}_{i\in[\ell_1]}, \{\mathsf{ek}^{(j)}_{1,i}\}_{i\in[\ell_1]}, \mathsf{ek}^{(j)}_2, \mathsf{ek}^{(j)}_3, \mathsf{ek}^{(j)}_4)$$

are created for the same $\mathcal{S}^*_{\mathsf{snd}}$ for all $j \in [q_e]$, where $\mathcal{S}^*_{\mathsf{snd}} \models \mathbb{A}^*_{\mathsf{rcv}}$ (the deduction of this step is deterred to the end of the proof). It indicates that the ciphertext $\mathsf{CT}^*_{\mathsf{snd}}$ must be constructed as

$$\begin{aligned} \mathsf{ct}^*_{6,i} &= \prod_{j=1}^{q_e}(H(u_i)^{\tau_j})^{f_j\cdot c}\cdot h_i^{e\cdot c} = (H(u_i)^{\tau_0}\cdot h_i^e)^c, \\ \mathsf{ct}^*_7 &= \prod_{j=1}^{q_e}(\delta_1^{\tau_j})^{f_j\cdot c'}\cdot \delta_1^{e\cdot c'} = \left(\delta_1^{\tau_0}\cdot\delta_1^e\right)^{c'}, \\ \mathsf{ct}^*_8 &= \prod_{j=1}^{q_e}(\delta_2^{\tau_j})^{f_j\cdot c''}\cdot \delta_2^{e\cdot c''} = \left(\delta_2^{\tau_0}\cdot\delta_2^e\right)^{c''}, \\ \mathsf{ct}^*_9 &= \prod_{j=1}^{q_e}(g_1^x h^{\tau_j})^{f_j\cdot c}\cdot h^{e\cdot c} = \left((g_1^x h^{\tau_0})\cdot h^e\right)^c, \end{aligned}$$

where $f_j = f_{1,j} = f_{2,j} = f_{3,j} = f_{4,j}$ and $\tau_0 = \sum_{j=1}^{q_e}\tau_j f_j$.

We can deduce that

$$\mathsf{ct}^*_{6,i} = (\mathsf{ek}^*_{1,i}\cdot h_i^e)^c,\ \mathsf{ct}^*_7 = (\mathsf{ek}^*_2\cdot\delta_1^e)^{c'},$$

$$\mathsf{ct}^*_8 = (\mathsf{ek}^*_3\cdot\delta_2^e)^{c''},\ \mathsf{ct}^*_9 = (\mathsf{ek}^*_4\cdot h^e)^c,$$

where $\mathsf{ek}^*_{1,i} = H(u_i)^{\tau_0}$, $\mathsf{ek}^*_2 = \delta_1^{\tau_0}$, $\mathsf{ek}^*_3 = \delta_2^{\tau_0}$, $\mathsf{ek}^*_4 = g_1^x h^{\tau_0}$. Since the scalers (c, c', c'', e) are selected by the adversary, $\mathcal{A}$ can easily calculate $\mathsf{EK}^*_{\mathsf{snd}} = (\{n_i^*\}_{i\in[\ell_1]}, \{\mathsf{ek}^*_{1,i}\}_{i\in[\ell_1]}, \mathsf{ek}^*_2, \mathsf{ek}^*_3, \mathsf{ek}^*_4)$ for $\mathcal{S}^*_{\mathsf{snd}}$ and $\mathcal{S}^*_{\mathsf{snd}} \models \mathbb{A}^*_{\mathsf{rcv}}$, which contradicts with the constraint.

Therefore, the adversary cannot symbolically produce a valid forgery, and the *authenticity* of FEME is proved. □

In the following, we proved that $\mathsf{EK}^{(j)}_{\mathcal{S}_{\mathsf{snd}}}$ are created for the same $\mathcal{S}^*_{\mathsf{snd}}$.

(1) Let $q_e = 1$. We have $f_{1,j} = f_{2,j} = f_{3,j} = f_{4,j} = f_j = 1$ and

$$\mathsf{ct}^*_{6,i} = (\mathsf{ek}_{1,i})^c\cdot h_i^{e\cdot c},\ \mathsf{ct}^*_7 = (\mathsf{ek}_2)^{c'}\cdot\delta_1^{e\cdot c'},$$

$$\mathsf{ct}^*_8 = (\mathsf{ek}_3)^{c''}\cdot\delta_2^{e\cdot c''},\ \mathsf{ct}^*_9 = (\mathsf{ek}_4)^c\cdot h^{e\cdot c}.$$

If the challenge ciphertext $\mathsf{CT}^*_{\mathsf{snd}}$ is valid, the adversary can directly calculate $\mathsf{EK}^*_{\mathsf{snd}}$ for $\mathcal{S}^*_{\mathsf{snd}}$ and $\mathcal{S}^*_{\mathsf{snd}} \models \mathbb{A}^*_{\mathsf{rcv}}$, which contradicts with the constraint.

(2) Let $q_e = 2$. We have $f_{1,j} = f_{2,j} = f_{3,j} = f_{4,j} = f_j$ and $\sum_{j=1}^2 f_j = f_1 + f_2 = 1$.

(2.1) We firstly prove that $\mathcal{S}^{(1)}_{\mathsf{snd}} = \{u_i^{(1)}\}_{i\in[\ell_1^{(1)}]}$ and $\mathcal{S}^{(2)}_{\mathsf{snd}} = \{u_i^{(2)}\}_{i\in[\ell_1^{(2)}]}$ must have the same number of attributes, i.e., $\ell_1^{(1)} = \ell_1^{(2)}$.

Suppose that $\mathcal{S}^{(1)}_{\mathsf{snd}}$ and $\mathcal{S}^{(2)}_{\mathsf{snd}}$ has different number of attributes and $\ell_1^{(1)} > \ell_1^{(2)}$. We have

$$\mathsf{ct}^*_{6,i} = \left(H(u_i^{(1)})^{\tau_1 f_1}\cdot h_i^e\right)^c,\ \text{for } i = \ell_1^{(1)}.$$

Therefore, $\tau_0 = \tau_1 f_1$. Since $\tau_0 = \sum_{j=1}^2\tau_j f_j$, it can be deduced that $f_2 = 0$. Since $f_1 + f_2 = 1$, we have $f_1 = 1$. Then,

$$\mathsf{ct}^*_{6,i} = (\mathsf{ek}^*_{1,i}\cdot h_i^e)^c,\ \mathsf{ct}^*_7 = (\mathsf{ek}^*_2\cdot\delta_1^e)^{c'},$$

$$\mathsf{ct}^*_8 = (\mathsf{ek}^*_3\cdot\delta_2^e)^{c''},\ \mathsf{ct}^*_9 = (\mathsf{ek}^*_4\cdot h^e)^c.$$

$\mathcal{A}$ can easily calculate $\mathsf{EK}^*_{\mathsf{snd}}$ for $\mathcal{S}^*_{\mathsf{snd}}$ and $\mathcal{S}^*_{\mathsf{snd}} \models \mathbb{A}^*_{\mathsf{rcv}}$, which contradicts with the constraint.

(2.2) Next, we prove that $\mathcal{S}^{(1)}_{\mathsf{snd}} = \mathcal{S}^{(2)}_{\mathsf{snd}}$. If $\mathcal{S}^{(1)}_{\mathsf{snd}} \neq \mathcal{S}^{(2)}_{\mathsf{snd}}$, there must exist some certain attribute $u_i^{(1)} \neq u_i^{(2)}$, where $i \in [\ell_1^{(2)}]$. Put $u_i^{(1)}$ and $u_i^{(2)}$ into the random oracle to get the hash values $h_i^{(1)} = H(u_i^{(1)}) = g_1^{t_i^{(1)}}$ and $h_i^{(2)} = H(u_i^{(2)}) = g_1^{t_i^{(2)}}$, where $t_i^{(1)}, t_i^{(2)} \xleftarrow{\$} \mathbb{Z}_p^*$ and $t_i^{(1)} \neq t_i^{(2)}$.

The ciphertext $\mathsf{CT}^*_{\mathsf{snd}}$ is constructed as

$$\begin{aligned} \mathsf{ct}^*_{6,i} &= \prod_{j=1}^{2}(H(u_i^{(j)})^{\tau_j})^{f_j\cdot c}\cdot h_i^{e\cdot c}, \\ \mathsf{ct}^*_7 &= \prod_{j=1}^{2}(\delta_1^{\tau_j})^{f_j\cdot c'}\cdot \delta_1^{e\cdot c'} = \left(\delta_1^{\tau_0}\cdot\delta_1^e\right)^{c'}, \\ \mathsf{ct}^*_8 &= \prod_{j=1}^{2}(\delta_2^{\tau_j})^{f_j\cdot c''}\cdot \delta_2^{e\cdot c''} = \left(\delta_2^{\tau_0}\cdot\delta_2^e\right)^{c''}, \\ \mathsf{ct}^*_9 &= \prod_{j=1}^{2}(g_1^x h^{\tau_j})^{f_j\cdot c}\cdot h^{e\cdot c} = \left((g_1^x h^{\tau_0})\cdot h^e\right)^c, \end{aligned}$$

where $\tau_0 = \sum_{j=1}^2\tau_j f_j$.

If the ciphertext $\mathsf{CT}^*_{\mathsf{snd}}$ is valid, it must have

$$\mathsf{ct}^*_{6,i} = \prod_{j=1}^{2}(H(u_i^{(j)})^{\tau_j})^{f_j\cdot c}\cdot h_i^{e\cdot c} = (H(\bar{u}_i)^{\tau_0}\cdot h_i^e)^c$$

for some attribute $\bar{u}_i$. Let $\bar{h}_i = H(\bar{u}_i) = g_1^{\bar{t}_i}$, where $\bar{t}_i \xleftarrow{\$} \mathbb{Z}_p^*$. We have

$$\begin{aligned}\mathsf{ct}_{6,i}^* &= \prod\nolimits_{j=1}^{2} (H(u_i^{(j)})^{\tau_j})^{f_j \cdot c} \cdot h_i^{e \cdot c} \\ &= (g_1^{\sum_{j=1}^{2} t^{(j)} \cdot \tau_j f_j} \cdot h_i^e)^c \\ &= (g^{\bar{t}_i \tau_0} \cdot h_i^e)^c = (H(\bar{u}_i)^{\tau_0} \cdot h_i^e)^c.\end{aligned}$$

Since $\tau_0 = \sum_{j=1}^{2} \tau_j f_j$ and $\sum_{j=1}^{2} f_j = f_1 + f_2 = 1$, we can deduce that

$$\begin{aligned} & \sum\nolimits_{j=1}^{2} t_i^{(j)} \cdot f_j \tau_j = \bar{t}_i \tau_0 \\ \Rightarrow \quad & t_i^{(1)} \tau_1 f_1 + t_i^{(2)} \tau_2 f_2 = \bar{t}_i (\tau_1 f_1 + \tau_2 f_2) \\ \Rightarrow \quad & (t_i^{(1)} - \bar{t}_i) f_1 \tau_1 + (t_i^{(2)} - \bar{t}_i) f_2 \tau_2 = 0.\end{aligned}$$

As τ_1, τ_2 are random numbers in $\mathbb{Z}_p^*$, we have $(t_i^{(1)} - \bar{t}_i) f_1 = 0$ and $(t_i^{(2)} - \bar{t}_i) f_2 = 0$. Since $f_1 + f_2 = 1$, we have $t_i^{(1)} = \bar{t}_i$ and $f_1 = 1$, $f_2 = 0$, or $f_1 = 0$, $f_2 = 1$ and $t_i^{(2)} = \bar{t}_i$. In either way, if the challenge ciphertext $\mathsf{CT}_{\mathsf{snd}}^*$ is valid, the adversary can directly calculate $\mathsf{EK}_{\mathsf{snd}}^*$ for $\mathcal{S}_{\mathsf{snd}}^*$ and $\mathcal{S}_{\mathsf{snd}}^* \models \mathbb{A}_{\mathsf{rcv}}^*$, which contradicts with the constraint.

(3) Let $q_e \geqslant 3$. We have $f_{1,j} = f_{2,j} = f_{3,j} = f_{4,j} = f_j$ and $\sum_{j=1}^{q_e} f_j = 1$.

(3.1) We firstly prove that $\mathcal{S}_{\mathsf{snd}}^{(j)}$ must have the same number of attributes, where $j \in [q_3]$.

Suppose that $\mathcal{S}_{\mathsf{snd}}^{(1)}$, $\mathcal{S}_{\mathsf{snd}}^{(2)}$, ... $\mathcal{S}_{\mathsf{snd}}^{(q_e)}$ has different number of attributes and $\ell_1^{(1)} > \ell_1^{(2)} > \cdots > \ell_1^{(q_e)}$. We deduce that

$$\begin{aligned}\mathsf{ct}_{6,\ell_1^{(1)}}^* &= \big(H(u^{(1)})^{\tau_1 f_1} \cdot h_{\ell_1^{(1)}}^e\big)^c = \big(H(u^{(1)})^{\tau_0} \cdot h_{\ell_1^{(1)}}^e\big)^c, \\ & \qquad\qquad\qquad\qquad \text{where } h_{\ell_1^{(1)}} = H(u^{(1)}), \\ \mathsf{ct}_{6,\ell_1^{(2)}}^* &= \big(H(\hat{u}^{(1)})^{\tau_1 f_1} \cdot H(\hat{u}^{(2)})^{\tau_2 f_2} \cdot h_{\ell_1^{(2)}}^e\big)^c \\ &= \big(H(\bar{u})^{\tau_0} \cdot h_{\ell_1^{(2)}}^e\big)^c, \qquad \text{where } h_{\ell_1^{(2)}} = H(\bar{u}), \\ & \cdots \\ \mathsf{ct}_{6,\ell_1^{(q_e)}}^* &= \big(H(\tilde{u}^{(1)})^{\tau_1 f_1} \cdots H(\check{u}^{(q_e)})^{\tau_{q_e} f_{q_e}} \cdot h_{\ell_1^{(3)}}^e\big)^c, \\ &= \big(H(\check{u})^{\tau_0} \cdot h_{\ell_1^{(3)}}^e\big)^c, \qquad \text{where } h_{\ell_1^{(3)}} = H(\check{u}), \\ \mathsf{ct}_7^* &= \prod_{j=1}^{q_e} (\delta_1^{\tau_j})^{f_j \cdot c'} \cdot \delta_1^{e \cdot c'} = \big(\delta_1^{\tau_0} \cdot \delta_1^e\big)^{c'}, \\ \mathsf{ct}_8^* &= \prod_{j=1}^{q_e} (\delta_2^{\tau_j})^{f_j \cdot c''} \cdot \delta_2^{e \cdot c''} = \big(\delta_2^{\tau_0} \cdot \delta_2^e\big)^{c''}, \\ \mathsf{ct}_9^* &= \prod_{j=1}^{q_e} (g_1^x h^{\tau_j})^{f_j \cdot c} \cdot h^{e \cdot c} = \big((g_1^x h^{\tau_0}) \cdot h^e\big)^c,\end{aligned}$$

where $f_j = f_{1,j} = f_{2,j} = f_{3,j} = f_{4,j}$.

Therefore, we have $\tau_0 = \tau_1 f_1$ and $\tau_0 = \sum_{j=1}^{q_e} \tau_j f_j$. It can be deduced that $f_2 = f_3 = \cdots = f_{q_e} = 0$. Since $\sum_{j=1}^{q_e} f_j = 1$, we have $f_1 = 1$. Then,

$$\mathsf{ct}_{6,i}^* = (\mathsf{ek}_{1,i}^* \cdot h_i^e)^c, \ \mathsf{ct}_7^* = (\mathsf{ek}_2^* \cdot \delta_1^e)^{c'},$$

$$\mathsf{ct}_8^* = (\mathsf{ek}_3^* \cdot \delta_2^e)^{c''}, \ \mathsf{ct}_9^* = (\mathsf{ek}_4^* \cdot h^e)^c.$$

$\mathcal{A}$ can easily calculate $\mathsf{EK}_{\mathsf{snd}}^*$ for $\mathcal{S}_{\mathsf{snd}}^*$ and $\mathcal{S}_{\mathsf{snd}}^* \models \mathbb{A}_{\mathsf{rcv}}^*$, which contradicts with the constraint.

(3.2) Next, we prove that $\mathcal{S}_{\mathsf{snd}}^{(1)} = \cdots = \mathcal{S}_{\mathsf{snd}}^{(q_e)}$. If $\mathcal{S}_{\mathsf{snd}}^{(1)} \neq \mathcal{S}_{\mathsf{snd}}^{(2)}$, there must exist some certain attribute $u_i^{(1)} \neq \cdots \neq u_i^{(q_e)}$, where $i \in [\ell_1^{(3)}]$. Put $u_i^{(1)}, \cdots, u_i^{(q_e)}$ into the random oracle to get the hash values $h_i^{(1)} = H(u_i^{(1)}) = g_1^{t_i^{(1)}}, \cdots, h_i^{(q_e)} = H(u_i^{(q_e)}) = g_1^{t_i^{(q_e)}}$, where $t_i^{(1)}, \cdots, t_i^{(q_e)} \xleftarrow{\$} \mathbb{Z}_p^*$ and $t_i^{(1)} \neq \cdots \neq t_i^{(q_e)}$.

The ciphertext $\mathsf{CT}_{\mathsf{snd}}^*$ is constructed as

$$\begin{aligned}\mathsf{ct}_{6,i}^* &= \prod_{j=1}^{q_e} (H(u_i^{(j)})^{\tau_j})^{f_j \cdot c} \cdot h_i^{e \cdot c}, \\ \mathsf{ct}_7^* &= \prod_{j=1}^{q_e} (\delta_1^{\tau_j})^{f_j \cdot c'} \cdot \delta_1^{e \cdot c'} = \big(\delta_1^{\tau_0} \cdot \delta_1^e\big)^{c'}, \\ \mathsf{ct}_8^* &= \prod_{j=1}^{q_e} (\delta_2^{\tau_j})^{f_j \cdot c''} \cdot \delta_2^{e \cdot c''} = \big(\delta_2^{\tau_0} \cdot \delta_2^e\big)^{c''}, \\ \mathsf{ct}_9^* &= \prod_{j=1}^{q_e} (g_1^x h^{\tau_j})^{f_j \cdot c} \cdot h^{e \cdot c} = \big((g_1^x h^{\tau_0}) \cdot h^e\big)^c,\end{aligned}$$

where $\tau_0 = \sum_{j=1}^{q_e} \tau_j f_j$.

If the ciphertext $\mathsf{CT}_{\mathsf{snd}}^*$ is valid, it must have

$$\mathsf{ct}_{6,i}^* = \prod_{j=1}^{q_e} (H(u_i^{(j)})^{\tau_j})^{f_j \cdot c} \cdot h_i^{e \cdot c} = (H(\bar{u}_i)^{\tau_0} \cdot h_i^e)^c$$

for some attribute $\bar{u}_i$. Let $\bar{h}_i = H(\bar{u}_i) = g_1^{\bar{t}_i}$, where $\bar{t}_i \xleftarrow{\$} \mathbb{Z}_p^*$. We have

$$\begin{aligned}\mathsf{ct}_{6,i}^* &= \prod\nolimits_{j=1}^{q_e} (H(u_i^{(j)})^{\tau_j})^{f_j \cdot c} \cdot h_i^{e \cdot c} \\ &= (g_1^{\sum_{j=1}^{q_e} t^{(j)} \cdot \tau_j f_j} \cdot h_i^e)^c \\ &= (g^{\bar{t}_i \tau_0} \cdot h_i^e)^c = (H(\bar{u}_i)^{\tau_0} \cdot h_i^e)^c.\end{aligned}$$

Since $\tau_0 = \sum_{j=1}^{q_e} \tau_j f_j$ and $\sum_{j=1}^{q_e} f_j = 1$, we can deduce that

$$\begin{aligned} & \sum\nolimits_{j=1}^{q_e} t_i^{(j)} f_j \tau_j = \bar{t}_i \tau_0 = \sum\nolimits_{j=1}^{q_e} \bar{t}_i \tau_j f_j \\ \Rightarrow \quad & \sum\nolimits_{j=1}^{q_e} (t_i^{(j)} - \bar{t}_i) f_j \tau_j = 0.\end{aligned}$$

As $\tau_1, \cdots, \tau_{q_e}$ are random numbers in $\mathbb{Z}_p^*$, we have $(t_i^{(j)} - \bar{t}_i) f_j = 0$ for $j \in [q_e]$. Since $\sum_{j=1}^{q_e} f_j = 1$, we have $t_i^{(j)} = \bar{t}_i$, $f_j = 1$ for some $j \in [q_e]$, and $f_{j'} = 0$, for the $j' \in [q_e]$ and $j' \neq j$. In either way, if the challenge ciphertext $\mathsf{CT}_{\mathsf{snd}}^*$ is valid, the adversary can directly calculate $\mathsf{EK}_{\mathsf{snd}}^*$ for $\mathcal{S}_{\mathsf{snd}}^*$ and $\mathcal{S}_{\mathsf{snd}}^* \models \mathbb{A}_{\mathsf{rcv}}^*$, which contradicts with the constraint.

Therefore, we prove that EK are created for the same $\mathcal{S}_{\mathsf{snd}}^*$ for all $j \in [q_e]$, where $\mathcal{S}_{\mathsf{snd}}^* \models \mathbb{A}_{\mathsf{rcv}}^*$.

This completes the proof of Theorem 3. ■