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

Yang Yang*, Guomin Yang*, Yingjiu Li†, Pengfei Wu*, Rui Shi‡, Minming Huang*,
Jian Weng§, HweeHwa Pang*, Robert H. Deng*
*Singapore Management University, Singapore
({yyang, gmyang, pfwu, mmhuang, hhpang, robertdeng}@smu.edu.sg)
†University of Oregon, USA (yingjiul@uoregon.edu)
‡Hainan University, China (shir@hainanu.edu.cn)
§Jinan University, Guangzhou, China (cryptjweng@gmail.com)

***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× faster for encryption and 6.23× 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.

I. Introduction

Service discovery (SD) protocols, including Wi-Fi [10], AirDrop [11], and BLE [12], are integral to modern wireless networks but lack robust privacy safeguards. This exposes them to tracking, linkability, and identity exposure attacks, where adversaries monitor device presence, track movements, and link sessions, leading to profiling and privacy breaches [22], [29], [41]. Despite the adoption of protocols like DNS-SD [14], mDNS [34], SSDP [27], and UPnP [15], their use of cleartext advertisements and lack of authentication lead to spoofing, MitM, and DoS attacks [17], [44], [45].

Existing privacy-enhancing protocols such as PrivateDrop [31] and WTSB [47] still fall short, as they lack policy-controlled access and attribute hiding, leaving users vulnerable to tracking and impersonation [42]. Similarly, CBN [22] provides anonymous client authentication but fails to protect service providers from spoofing.

These limitations underscore the need for privacy-preserving SD mechanisms aligned with global standards. Regulations such as RFC 7258 [28], ISO/IEC 29184 [7], and GDPR [3] mandate confidentiality and privacy-by-design; NIST SP 800-63B [4] and ETSI TS 103 465 [5] advocate bilateral access control; and RFC 6973 [26], the NIST Privacy Framework [8], and OECD Guidelines [2] emphasize bilateral anonymity. Sender authentication is equally essential, mandated by NIST SP 800-53 [9], ISO/IEC 29115 [1], and ETSI EN 303 645 [6].

Recently, Yang et al. [49] proposed PriSrv (NDSS'24), addressing major privacy and usability challenges in service discovery. Its dual-layer architecture enables only authorized clients to discover services, protecting sensitive information during interactions. Unlike protocols such as AirDrop and BLE that lack strong privacy guarantees, PriSrv supports bilateral policy control through anonymous credential-based matchmaking encryption (ACME), providing mutual authentication and defending against MitM attacks, tracking, and profiling.

However, PriSrv inherits limitations from ACME. It reveals public attributes in the outer layer, potentially enabling tracking. Its binary attribute vector model (each attribute can only represent 1 or 0) restricts expressiveness and increases computation with large attribute sets. ACME's small-universe design requires system rebuilds to add new attributes, hindering scalability. Additionally, large ciphertexts lead to high communication overhead, and pre-issued anonymous credentials introduce management complexity.

To overcome these issues, we propose a fast and expressive matchmaking encryption (FEME) scheme, which is also of

Network and Distributed System Security (NDSS) Symposium 2026
23-27 February 2026, San Diego, CA, USA
ISBN 979-8-9919276-8-0
https://dx.doi.org/10.14722/ndss.2026.230087
www.ndss-symposium.org

independent interest for advancing matchmaking encryption (ME). Unlike prior Identity-Based ME (IBME) schemes [16], [25] that support only equality policies, FEME enables expressive policies with arbitrary strings and solves an open problem posed in CRYPTO'19 and ASIACRYPT'22. Additionally, FEME is significantly faster in encryption and decryption than the existing ME scheme supporting expressive policy control [49].

To further ensure robust authentication and privacy, FEME introduces a novel *double re-randomization and binding technique*, which prevents encryption key extraction, thwarts ciphertext forgery and component-mixing attacks, and conceals sensitive attribute values. FEME achieves bilateral access control, bilateral anonymity, and sender authentication in the context of expressive policies with high efficiency. It adopts a *partially hidden access structure* [33], where only attribute names are exposed while attribute values remain concealed, enabling efficient policy matching without revealing sensitive information. Combined with a *randomness splitting technique* [36], FEME offers a practical and privacy-preserving solution for expressive matchmaking encryption.

Building on the strengths of FEME, PriSrv+ overcomes the limitations of its predecessor, PriSrv, and introduces new capabilities. It enhances usability by eliminating the reliance on anonymous credentials and the associated overhead of credential issuance and revocation. PriSrv+ supports expressive bilateral policy control and flexible attribute representation, lifting the constraints of binary vectors and small-universe designs in PriSrv. It also significantly reduces communication overhead, shrinking broadcast sizes by up to 87.33%, which boosts scalability and performance in low-bandwidth settings. Additionally, PriSrv+ improves privacy by concealing all attribute values during service discovery, providing a robust and efficient solution for privacy-preserving 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. FEME offers up to 7.62× faster encryption and 6.23× faster decryption compared with ACME, making PriSrv+ suitable for wireless 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.

• *Formal Security and Privacy Guarantees*. Rigorous formal security proofs demonstrate that FEME satisfies confidentiality, anonymity, and authenticity. PriSrv+ is proven to be 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.

II. Related Work

A. Service Discovery Protocols

Service discovery (SD) protocols such as Wi-Fi [10], AirDrop [11], and Bluetooth Low Energy (BLE) [12] facilitate the automatic detection and advertisement of services and devices in dynamic networks, streamlining device interactions. However, these protocols pose significant privacy risks, particularly for users wishing to safeguard sensitive or identifying information. Studies show that about 90% of users view the exposure of device names as a privacy threat [32], enabling adversaries to infer personal data such as location, mobility, and user profiles [41], [42], [47], [51]. For example, device names in public Wi-Fi can allow Internet Service Providers (ISPs) to track users [22], while attackers in IoT networks can analyze service data to reveal user routines [29].

Most existing SD protocols, including DNS-SD [14], mDNS [34], SSDP [27], and UPnP [15], lack strong privacy safeguards, leaving them vulnerable to man-in-the-middle (MitM), spoofing, and denial-of-service (DoS) attacks [17], [45]. These risks are exacerbated by the use of cleartext broadcasts in Wi-Fi and BLE, which expose device identifiers and enable adversarial tracking and profiling [44]. Although protocols like CBN [22] support anonymous client authentication, they offer insufficient protection for service providers, who remain exposed to impersonation and MitM attacks.

Protocols such as AirDrop [11], PrivateDrop [31], and WTSB [47] introduce encryption and authentication to enhance service discovery privacy. While improving mutual

authentication and anonymity, they still suffer from tracking, MitM, and DoS vulnerabilities due to incomplete privacy features, such as selective attribute disclosure and multi-show unlinkability [17], [42]. For example, reliance on certificates in AirDrop and PrivateDrop may allow attackers to link sessions and track users [31].

Yang et al. introduced PriSrv [49], a private SD protocol that allows service providers and clients to define fine-grained access control policies, enabling mutual authentication while concealing private information. PriSrv leverages Anonymous Credential-based Matchmaking Encryption (ACME) to support bilateral policy control, selective attribute disclosure, and multishow unlinkability. However, its large message size results in high transmission overhead and reception delays, limiting its effectiveness in low-bandwidth networks like BLE and congested Wi-Fi. Additionally, the exposure of public attributes may lead to 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.

B. Matchmaking Encryption (ME)

Matchmaking Encryption (ME) was introduced by Ateniese et al. [16] in CRYPTO'19 as a new encryption paradigm enabling both sender and receiver to specify policies that must be mutually satisfied for successful decryption. In ME, the sender with identity or attribute σ defines a policy $\mathbb{R}$, and the receiver with ρ defines $\mathbb{S}$; decryption succeeds only if σ satisfies $\mathbb{S}$ and ρ satisfies $\mathbb{R}$. Ateniese et al. also instantiated Identity-Based Matchmaking Encryption (IBME) in the random oracle model, where equality-based identities are used, and sender authentication is achieved via embedded encryption keys.

Francati et al. [30] extended IBME to the standard model using non-standard assumptions and NIZK proofs, while Chen et al. [25] constructed IBME under standard assumptions. Despite providing data privacy and authenticity, these schemes are limited to equality-based policies and 1-to-1 data sharing.

To support one-to-many data sharing in ME, Sun et al. [43] and Yang et al. [48] in TIFS'23 proposed privacy-aware ME (PSME) and certificateless ME (CLME), respectively—extending IBME to multi-user settings via identity-based broadcast encryption. Wu et al. [46] introduced fuzzy IBME (FBME), enabling decryption when the overlap between sender and receiver attributes exceeds a threshold. However, FBME's threshold-based policies have limited expressiveness and incur high decryption costs.

Recently, Yang et al. [49] in NDSS'24 developed ACME, an anonymous credential-based ME scheme with flexible bilateral policy control. Despite its utility, ACME suffers from large ciphertext size and a small-universe construction that requires binary attribute vectors—leading to large vector sizes and increased computation. In contrast, FEME supports monotonic Boolean policies with an unrestricted attribute universe, allowing arbitrary strings as attributes. It also improves performance, reducing ciphertext size by 87.33% and achieving up to 7.62× faster encryption and 6.23× faster decryption.

III. Preliminary

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

A. 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).

The notation $s \xleftarrow{\$} S$ indicates that s is uniformly sampled from set 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. 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 $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$.

B. Access Structure

Definition 1 (Access Structure [18])**.** *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., $\mathbb{A} \subseteq 2^{\{P_1, \cdots, P_n\}} \setminus \{\varnothing\}$. 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.

C. Linear Secret Sharing Scheme (LSSS)

Definition 2 (Linear Secret Sharing Scheme (LSSS) [18])**.** *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 $\rho(i)$.*

LSSS possesses the linear reconstruction property [18]. Let Π be an LSSS for the access structure $\mathbb{A}$, and $S \in \mathbb{A}$ 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 be the i-th row of $\mathbf{A}$, we similarly have $\sum_{i \in I} \omega_i A_i = (1, 0, \cdots, 0)$. These constants $\{\omega_i\}$ are computable in time polynomial [18] in the size of $\mathbf{A}$. Notably, 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 [18], 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.

D. Partially Hidden Access Structure

In a partially hidden access structure [33], 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 exposed in the policy.

We first 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 a 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 $\mathcal{T}_{\text{name}} = (n_{\pi(1)}, \cdots, n_{\pi(m)})$. It 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.

IV. 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 ME techniques.

A. 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) [19], [23] 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.

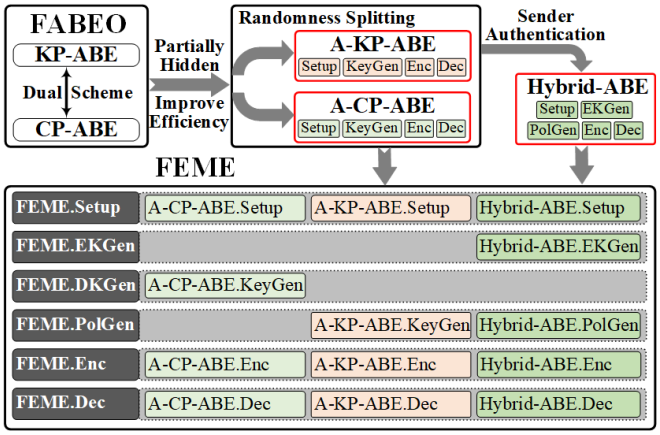


Fig. 1: Technical Roadmap of FEME

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 build on FABEO [38], 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 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.

B. Novelty of FEME

Distinct from Existing ABE-Based Solutions. FEME achieves bilateral access control, bilateral anonymity, and

[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.

sender authentication—features not simultaneously supported by existing ABE-based schemes such as FABEO [38], FEASE [36], and FABESA [35]. These works optimize ABE efficiency or enable unilateral anonymity for searchable encryption but lack bilateral policy matching and sender authentication, both of which are critical for privacy-preserving service discovery.

Technical Challenges and Innovations. Designing an ME scheme for real-time service discovery presents significant challenges beyond traditional ABE systems. Existing ABE schemes, including combinations of CP-ABE and KP-ABE, cannot enforce *sender authentication*—a critical requirement in ME. Without sender authentication, malicious entities can forge ciphertexts with fabricated attributes, enabling impersonation, spoofing, and injection attacks that threaten both security and availability. Prior techniques such as partial policy hiding and randomness splitting, used in works like FEASE and FABESA, fall short of defending against these advanced threats in bilateral settings.

To overcome these limitations, FEME introduces a novel *double re-randomization and binding* technique that ensures both efficiency and robust sender authentication. The *first-level re-randomization* randomizes encryption key components using a shared factor, preventing adversaries from extracting or reusing the sender's encryption key. The *second-level re-randomization* applies additional independent randomness to specific ciphertext components while enforcing a constraint across them. This enforces *binding* between encryption key-derived components and other ciphertext elements, ensuring they cannot be mixed or tampered with to create forged messages. Moreover, the second-level re-randomization conceals sensitive attribute values to resist attribute guessing attacks. Together, these mechanisms provide strong protection against impersonation, ciphertext injection, and attribute guessing attacks—enabling secure, private, and authenticated service discovery in adversarial settings.

C. 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 balance privacy and efficiency, we adopt a *partially hidden access structure* (§III-D) that separates each attribute into a visible *attribute name* and a concealed *attribute value*, protecting sensitive information. Since attribute names—visible in A-KP-ABE or A-CP-ABE ciphertexts—are typically less sensitive, this design enables significant efficiency gains. Our A-CP-ABE and A-KP-ABE constructions minimize costly pairing and exponentiation operations, accelerating policy matching and improving suitability for resource-constrained environments.

(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 [38]), 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 equality $e(\mathsf{ct}_{1,u}, g_2) = e(H(u), \mathsf{ct}_2)$. 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 $\frac{1}{b_2}$. It ensures that the attribute set remains concealed, and the modified components $\mathsf{ct}_{1,i}$, ct_2, and

A-KP-ABE: Anonymous 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 $\alpha, b_1, b_2 \xleftarrow{\$} \mathbb{Z}_p^*$ and a hash function $H : \{0,1\}^* \to \mathbb{G}_1$. 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 $\mathsf{msk} := (\alpha, b_1, b_2)$.

$\underline{\mathsf{KeyGen}(\mathsf{msk}, \mathbb{A} = (\mathbf{A}, \rho, \{\Psi_{\rho(i)}\}_{i\in[m]})) \to \mathsf{SK}_\mathbb{A}}$. 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 $\mathsf{sk}_1 = g_2^{r'}$, $\mathsf{sk}_{2,i} = \big(g_1^{\mathbf{A}_i(\alpha||\mathbf{y})^\top} \cdot H(\Psi_{\rho(i)})^{r'}\big)^{\frac{1}{b_1}}$, $\mathsf{sk}_{3,i} = \big(g_1^{\mathbf{A}_i(\alpha||\mathbf{y})^\top} \cdot H(\Psi_{\rho(i)})^{r'}\big)^{\frac{1}{b_2}}$. Output $\mathsf{SK}_\mathbb{A} := ((\mathbf{A}, \rho, \{n_{\rho(i)}\}_{i\in[m]}), \mathsf{sk}_1, \{\mathsf{sk}_{2,i}, \mathsf{sk}_{3,i}\}_{i\in[m]})$.

$\underline{\mathsf{Enc}(\mathcal{S} = \{u_i\}_{i\in[\ell]} = \{\langle n_i, v_i\rangle\}_{i\in[\ell]}, \mathsf{msg}) \to \mathsf{CT}_\mathcal{S}}$. 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 $\mathsf{CT}_\mathcal{S} := (\{n_i\}_{i\in[\ell]}, \mathsf{ct}_0, \{\mathsf{ct}_{1,i}\}_{i\in[\ell]}, \mathsf{ct}_2, \mathsf{ct}_3)$.

$\underline{\mathsf{Dec}(\mathsf{SK}_\mathbb{A}, \mathsf{CT}_\mathcal{S}) \to \mathsf{msg}/\perp}$. 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

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

Fig. 2: A-KP-ABE Scheme

A-CP-ABE: Anonymous CP-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 $\alpha \xleftarrow{\$} \mathbb{Z}_p^*$, $h \xleftarrow{\$} \mathbb{G}_1$ and a hash function $H : \{0,1\}^* \to \mathbb{G}_1$. Compute $Z = e(g_1, g_2)^\alpha$. Output the master public key $\mathsf{mpk} := (\mathcal{G}, H, Z, h)$ and master secret key $\mathsf{msk} := \alpha$.

$\underline{\mathsf{KeyGen}(\mathsf{msk}, \mathcal{S} = \{u_i\}_{i\in[\ell]} = \{\langle n_i, v_i\rangle\}_{i\in[\ell]}) \to \mathsf{SK}_\mathcal{S}}$. 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 $\mathsf{SK}_\mathcal{S} := (\{n_i\}_{i\in[\ell]}, \mathsf{sk}_1, \{\mathsf{sk}_{2,i}\}_{i\in\ell}, \mathsf{sk}_3)$.

$\underline{\mathsf{Enc}(\mathbb{A} = (\mathbf{M}, \pi, \{\Psi_{\pi(i)}\}_{i\in[m]}), \mathsf{msg}) \to \mathsf{CT}_\mathbb{A}}$. 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 $\mathsf{CT} := ((\mathbf{M}, \pi, \{n_{\pi(i)}\}_{i\in[m]}), \mathsf{ct}_0, \mathsf{ct}_1, \mathsf{ct}_2, \{\mathsf{ct}_{3,i}\}_{i\in[m]})$.

$\underline{\mathsf{Dec}(\mathsf{SK}_\mathcal{S}, \mathsf{CT}_\mathbb{A}) \to \mathsf{msg}/\perp}$. 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

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

Fig. 3: A-CP-ABE Scheme

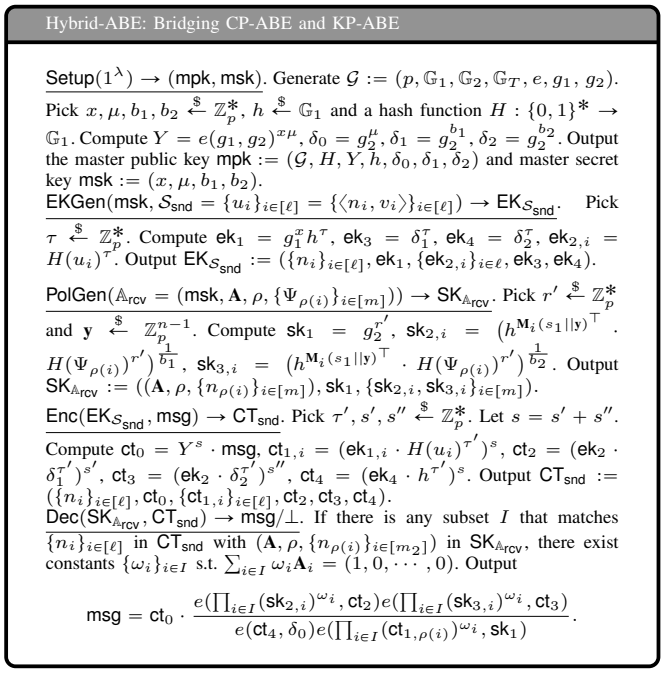


Fig. 4: Hybrid-ABE Scheme

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, its 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 its secret key components $\mathsf{sk}_{1,j} = g_2^{r'[j]}$, $\mathsf{sk}_{2,i} = g_1^{\mathbf{A}_i(\alpha||\mathbf{y})^\top} \cdot H(\Psi_{\rho(i)})^{r'[\eta(i)]}$, 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 (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.

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

FEME: Fast and Expressive Matchmaking Encryption

1. $\mathsf{Setup}(1^\lambda) \to (\mathsf{mpk}, \mathsf{msk})$. // 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 $H : \{0,1\}^* \to \mathbb{G}_1$, $\hat{H} : \mathbb{G}_T \to \{0,1\}^{l_0}$, and a polynomial-time computable padding function $\phi : \{0,1\}^n \to \{0,1\}^{\ell_0}$. 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. $\mathsf{EKGen}(\mathsf{msk}, \mathcal{S}_{\mathsf{snd}} = \{u_i\}_{i\in[\ell_1]} = \{\langle n_i, v_i\rangle\}_{i\in[\ell_1]}) \to \mathsf{EK}_{\mathcal{S}_{\mathsf{snd}}}$. // 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: $\mathsf{ek}_{1,i} = H(u_i)^\tau$ for $i \in [\ell_1]$, $\mathsf{ek}_2 = \delta_1^\tau$, $\mathsf{ek}_3 = \delta_2^\tau$, $\mathsf{ek}_4 = g_1^x h^\tau$. 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. $\mathsf{DKGen}(\mathsf{msk}, \mathcal{S}_{\mathsf{rcv}} = \{u_i\}_{i\in[\ell_2]} = \{\langle n_i, v_i\rangle\}_{i\in[\ell_2]}) \to \mathsf{DK}_{\mathcal{S}_{\mathsf{rcv}}}$. // 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: $\mathsf{dk}_1 = g_1^\alpha h^r$, $\mathsf{dk}_{2,i} = H(u_i)^r$, $\mathsf{dk}_3 = g_2^r$. 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. $\mathsf{PolGen}(\mathsf{msk}, \mathbb{A}_{\mathsf{rcv}} = (\mathbf{A}, \rho, \{\Psi_{\rho(i)}\}_{i\in[m_2]})) \to \mathsf{SK}_{\mathbb{A}_{\mathsf{rcv}}}$. // 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. $\mathsf{Enc}(\mathsf{EK}_{\mathcal{S}_{\mathsf{snd}}}, \mathbb{A}_{\mathsf{snd}} = (\mathbf{M}, \pi, \{\Psi_{\pi(i)}\}_{i\in[m_1]}), \mathsf{msg}) \to \mathsf{CT}_{\mathsf{snd}}$. // 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. $\mathsf{Dec}(\mathsf{DK}_{\mathcal{S}_{\mathsf{rcv}}}, \mathsf{SK}_{\mathbb{A}_{\mathsf{rcv}}}, \mathsf{CT}_{\mathsf{snd}}) \to \mathsf{msg}/\perp$: // 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 $\perp$.

Fig. 5: FEME: Fast and Expressive Matchmaking Encryption Scheme

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 [16] (Crypto'19)	×	×	√	√	√	√	×	√
IBME [30] (IndoCrypt'21)	×	×	√	√	×/√	√	×	√
IBME [25] (AsiaCrypt'21)	×	×	√	√	√	√	×	√
FBME [46] (TIFS'23)	×	×	×	√	√	√	×	√
PSME [43] (TIFS'23)	×	×	×	√	√	√	×	√
CLME [48] (TIFS'23)	×	×	√	√	√	√	×	√
ACME [49] (NDSS'24)	√	×	×	√	√	×	√	×
FEME	√	√	√	√	√	√	√	√

TABLE I: Comparison of Matchmaking Encryption (ME) Schemes

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 $(\mathsf{ct}_1, \mathsf{ct}_2, \{\mathsf{ct}_{3,i}\}_{i\in[m_2]})$ 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 Hybrid-ABE. Here, m_1 denotes the number of rows in sender's access matrix $\mathbf{M}$, and ℓ_1 the number of attributes in sender's attribute set.

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

D. FEME Construction

Following the above technical details, we describe the construction of FEME in Fig. 5. The security models for FEME, detailed in Appendix B, outline its confidentiality, anonymity, and authenticity properties.

In FEME, a key generation center (KGC) runs Setup to generate the master public and secret keys, incorporating an efficiently computable and invertible padding function ϕ that enables integrity checks, thereby ensuring authenticated message encryption and robustness against unauthorized modifications [16]. To enable secure communication, the KGC executes EKGen to create the sender's attribute encryption key, and $\mathsf{DKGen/PolGen}$ 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.

A core innovation of FEME is its *double re-randomization and binding technique*, which ensures secure and efficient sender authentication. *First re-randomization* applies a shared random value τ' to encryption key components $(\mathsf{ek}_{1,i}, \mathsf{ek}_2, \mathsf{ek}_3, \mathsf{ek}_4)$, generating ciphertext components $(\mathsf{ct}_{6,i}, \mathsf{ct}_7, \mathsf{ct}_8, \mathsf{ct}_9)$, preventing adversaries from extracting valid encryption keys. *Second re-randomization* utilizes random values s_3, s_3', s_3'', ensuring attackers cannot generate new valid ciphertexts via mimicry. The same s_3 *binds* encryption key-derived components with ciphertext elements $(V, \mathsf{ct}_2, \mathsf{ct}_{3,i})$, preventing attackers from mixing components from different ciphertexts. To conceal sensitive attribute values and prevent attribute guessing attacks, (s_3', s_3'') are applied to further re-randomize ciphertext components $(\mathsf{ct}_7, \mathsf{ct}_8)$.

FEME's partially hidden access structure enhances decryption efficiency by revealing only attribute names while concealing values. This allows the receiver to pre-filter unmatched ciphertexts without computation. In Dec, the receiver checks whether the sender's attribute names satisfy its policy and vice versa. If either check fails, decryption aborts with output $\perp$. Otherwise, full decryption proceeds to verify attribute values and padding integrity. The message is returned only if both policies are satisfied and the padding is valid; otherwise, $\perp$ ensures ciphertext integrity.

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 proof of Theorem 1 is deferred to Appendix C, and the proofs of Theorems 2-3 are deferred to the full version [50].

E. Comparative Advantages of FEME

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

Expressiveness. Both FEME and ACME [49] support monotonic Boolean formula-based access structures, unlike other schemes limited to identity-based matching. However, ACME's small-universe design restricts it to a fixed attribute set and relies on binary vectors, leading to longer vectors and higher computational costs for expressive policies. FEME supports an unrestricted attribute universe, allowing any arbitrary string as an attribute.

Security and Privacy. All schemes ensure data confidentiality. Francati et al. [30] provide an IBME without

authenticity and another using NIZK for authenticity. Except for ACME, all schemes preserve attribute privacy or identity privacy. ACME reveals outer-layer public attributes due to its dual-layer design.

Usability. IBME [16], [25], [30], FBME [46], PSME [43], and CLME [48] require *pre-registration pairing*, where the sender must know the receiver's identity or attributes beforehand. This tight coupling restricts flexible and real-time service discovery, as these schemes rely on identity-based or broadcast encryption. ACME [49] avoids such pairing but incurs extra overhead due to its dependence on anonymous credentials for sender authentication. In contrast, FEME *eliminates* both pre-registration pairing and external credential management by integrating sender authentication directly into ciphertexts, offering higher usability and scalability.

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

V. 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 [49], 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 journalist network operated by an NGO as an example, where the provider's policy is "(Journalist Type: Investigative **AND** Focus Area: Government Corruption **AND** Journalist Affiliation: Independent Media) **OR** (Role: Whistleblower **AND** Level: High Threat)," which is transformed to its partially hidden form "(Journalist Type **AND** Focus Area **AND** Journalist Affiliation) **OR** (Role **AND** 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 "(Network Type: Investigative **AND** Affiliation: NGO-Backed) **OR** (Jurisdiction: EU **AND** Support: Protection Available)". 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 "(Network Type **AND** Affiliation) **OR** (Jurisdiction **AND** Support)" 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.

A. Security and Threat Models of PriSrv+

Security Model. The security model of PriSrv+ follows that of PriSrv (Appendix C in full version) [49], 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 the other participant's attribute values unless they satisfy each other's access policies.

Threat Model. Similar to PriSrv, PriSrv+ assumes a fully trusted Key Generation Center (KGC) for key distribution, which does not participate in service discovery. Service providers and clients are considered *untrustworthy* and may attempt to gain unauthorized information or disrupt the protocol. Malicious providers may impersonate the other providers, track clients, or inject forged ciphertexts. Malicious clients may impersonate users or launch excessive requests to overwhelm providers. The mitigation of excessive requests is further discussed in Section V-D.

Following the Canetti-Krawczyk model for authenticated key exchange (AKE) [20], [21] and the service discovery model in [47], PriSrv+ considers a strong adversary capable of controlling public communications—eavesdropping, injecting, modifying, replaying, or interleaving messages across sessions. The adversary, which may be external, a rogue service provider, or a compromised client, can launch attacks including spoofing, impersonation, MitM, and DoS. Their goals include breaking authenticated key exchange and exposing sensitive information for tracking and inference.

B. PriSrv+ Construction

Fig. 6 presents the PriSrv+ protocol, comprising a privacy-preserving service broadcast phase and an anonymous mutual authentication phase. Key differences from PriSrv (highlighted in blue) include: (1) PriSrv+ uses the sender's encryption key EK for authentication, replacing PriSrv's anonymous credentials; and (2) PriSrv+ adopts FEME for bilateral policy control, whereas PriSrv employs ACME.

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 a lifetime (e.g., 30

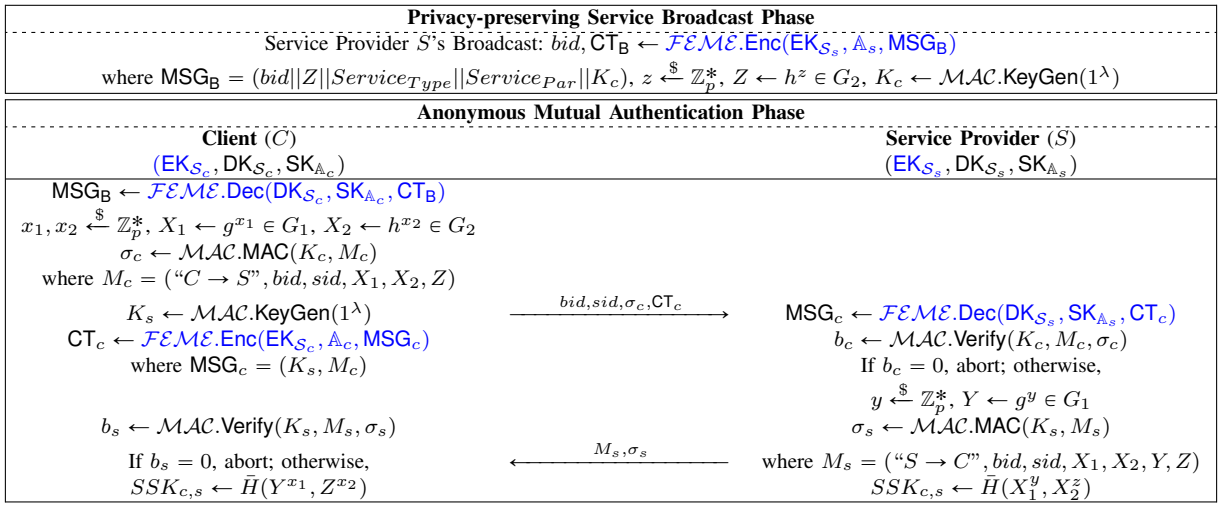


Fig. 6: PriSrv+ Protocol

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 $SSK_{c,s}$.

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.

C. Comparative Advantages of PriSrv+

In Table I of [49], PriSrv is shown to be the only protocol among 10 SD protocols—including 7 standard ones (DNS-SD [14], mDNS [34], SSDP [27], UPnP [15], Wi-Fi [10], BLE [12], AirDrop [11]) and 3 privacy-preserving ones (PrivateDrop [31], CBN [22], WTSB [47])—that achieves both high privacy and usability. Rather than revisiting prior comparisons, we directly compare PriSrv+ with PriSrv in terms of expressiveness, security and privacy, and usability. Table II summarizes the results.

In terms of *expressiveness*, PriSrv+ supports LSSS-defined policies with a large-universe construction, allowing any arbitrary string (e.g., a postal address) as an attribute. In contrast, PriSrv's small-universe design restricts it to a fixed attribute set, requiring a full system rebuild to add new attributes. Moreover, PriSrv represents attributes as binary vectors (1 or 0), resulting in longer vectors and increased encryption/decryption overhead for broader attribute sets.

Protocol	Expressiveness			Security and Privacy				Usability		
	Monotonic Policy	Arbitrary Attribute	Large Universe	Privacy Broadcast	Mutual Authn.	Bilateral Anon.	Pub. Attri. Hidden	No Pre-reg. Pairing	No 3rd-party Dependence	No In-advance ID Issuance
PriSrv [49]	√	×	×	√	√	√	×	√	√	×
PriSrv+	√	√	√	√	√	√	√	√	√	√

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

In terms of *security and privacy*, both protocols provide similar privacy protections. However, PriSrv discloses public attributes and access policies in its outer layer due to its dual-layer design. PriSrv+ mitigates this exposure by using a partially hidden access structure that separates attribute names from values, revealing only names during matching while keeping values confidential.

In terms of *usability*, neither protocol requires pre-registered pairings or third-party support for service discovery. However, PriSrv depends on anonymous credential issuance, adding management overhead. PriSrv+ eliminates this requirement by directly leveraging the sender's encryption key, simplifying deployment without compromising functionality.

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

D. Discussions

Justifying Attribute Concealment in PriSrv+. Although PriSrv could be modified to conceal all attributes and policies within its inner layer, this would substantially increase overhead. Its dual-layer architecture relies on public attributes in the outer layer to efficiently filter mismatched services without decryption. Concealing all attributes would force clients to decrypt every broadcast, incurring significant latency. PriSrv+ addresses this limitation using FEME, which reveals only attribute names while concealing values. This preserves fast filtering and eliminates tracking risks, leading to up to 3.55× faster encryption and 6.23× faster decryption compared to PriSrv, even with its outer-layer filtering mechanism.

Need for Expressive Attributes. Modern service discovery in IoT, smart cities, enterprise systems, and 5G networks requires expressive attributes—e.g., building names, project codes, or service categories—not supported by fixed identifier sets. PriSrv+ accommodates these scenarios through support for arbitrary string attributes, enabling fine-grained policy enforcement beyond traditional service discovery.

Mitigating DoS Attacks. To resist Denial-of-Service (DoS) attacks from excessive attribute submissions, PriSrv+ can incorporate a *flexible Proof-of-Work (PoW) mechanism*. The service provider embeds a difficulty level in the broadcast identifier (bid), and clients must compute a session ID (sid) such that $H(bid, sid, \sigma_c, \mathsf{CT}_c)$ satisfies this level (e.g., number of leading zeros). This imposes minimal overhead on legitimate users and service providers while significantly raising the attack cost. Dynamic difficulty adjustment and complementary measures—such as attribute limits and request throttling—further enhance resilience without harming usability.

VI. Implementation and Evaluation

For a fair comparison, we used the same benchmarks, crypto library, elliptic curves, and parameters as PriSrv [49] 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 [24] for the MAC scheme. Our source code is available at [37].

A. Evaluation of FEME and ACME

Table III 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 [49], 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 III: Performance of FEME and ACME (on Desktop)

Table III shows that FEME's computation (upper part) and communication costs (lower part) are substantially lower

[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-cfr g-pairing-friendly-curves.

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 IV: Performance of PriSrv+ and PriSrv (on Four Platforms)

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 exponentiation costs 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 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.

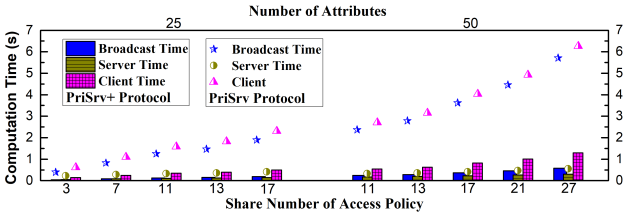
Overall, FEME significantly outperforms ACME, except in the |EK| size of EKGen.

B. Evaluation of PriSrv+ and PriSrv

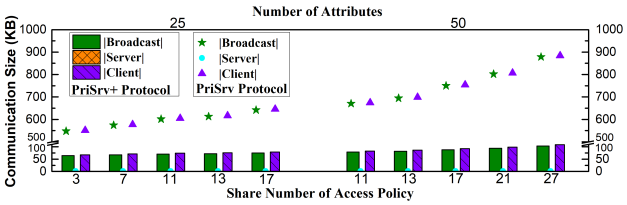
To comprehensively evaluate PriSrv+ and PriSrv [49], we conducted tests on four platforms (Table IV) using the same parameters as Table III across three elliptic curves. The platforms include a desktop (Intel Core i9-7920X), a laptop (Intel Core i5-10210U), a smartphone (ARM Cortex @2.4GHz), and a Raspberry Pi (ARM Cortex @1.5GHz), covering both high-performance and mobile environments relevant to wireless service discovery.

PriSrv+ consistently outperforms PriSrv in computation overhead. In the broadcast phase (upper Table IV), PriSrv+ averages 54.336 ms across platforms, compared to PriSrv's 443.868 ms—7.17× faster. In the mutual authentication phase (lower Table), PriSrv+ averages 233.181 ms vs. PriSrv's 1008.02 ms, achieving a 3.32× improvement. The total computation time of PriSrv+ is 287.517 ms, well under one second and perceived as an "immediate response" [31], [49], while PriSrv takes 1451.88 ms, making PriSrv+ 4.05× faster overall.

Table IV 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

Fig. 7: Comparison of PriSrv+ and PriSrv

We implemented two protocols in a real-world wireless environment using an open-source Wi-Fi Alliance project [10] 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) compares the computation costs of PriSrv+ and PriSrv under complex policies with more attributes. Bars show PriSrv+'s broadcast time (T_B), server time (T_S), and client time (T_C) in the anonymous authentication phase; symbols represent the same for PriSrv. We vary the number of attributes (top x-axis) and policy share number (bottom x-axis), where the latter reflects the number of shares required to reconstruct a secret [18] (i.e., the edge number in an equivalent access

tree). Both protocols are tested with identical attribute and policy sizes to ensure fairness.

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 (|Broadcast|, |Server|, |Client|), with PriSrv shown as symbols. The server's communication cost during the authentication phase is constant and identical for both protocols at |Server| = 0.82 KB. PriSrv+ has average communication costs of |Broadcast| = 79.623 KB and |Client| = 83.64 KB, while PriSrv averages |Broadcast| = 677.488 KB and |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.

Optimized Efficiency and Scalability. PriSrv+ ensures computational efficiency with overhead growing proportionally to the attribute number, consistent with standard ABE schemes. Unlike conventional ABE systems that suffer from performance degradation with flexible attributes, PriSrv+ supports an unrestricted attribute universe with minimal overhead. Experiments show up to 7.17× faster service broadcasts and 3.32× faster mutual authentication than PriSrv, even with complex policies. These gains result from encryption and decryption optimizations, enabling expressive attribute matching while maintaining real-time performance in wireless networks.

C. Interoperability

PriSrv+ is an enhanced version of PriSrv, improving expressiveness, privacy, and usability. Like PriSrv, it is interoperable with existing SD protocols like mDNS, BLE, EAP, AirDrop.

Integration Approaches. PriSrv+ can be integrated through two approaches. The first places PriSrv+ at the application layer, allowing it to work with existing lower-layer protocols. If the payload exceeds protocol limits, the lower layers handle segmentation and reassembly without altering PriSrv+'s logic. The second replaces lower-layer protocols with PriSrv+, requiring specific adaptations. We illustrate the first approach using mDNS and BLE, and the second with EAP and AirDrop.

Privacy-Enhanced mDNS and BLE. PriSrv+ can integrate into the Vanadium framework [13] to develop privacy-enhanced mDNS and BLE. Vanadium provides service discovery APIs for protocols like mDNS [34] and BLE [12]. mDNS, often paired with DNS-SD [14], supports additional attributes in TXT records (up to 65,535 bytes). PriSrv+ broadcasts (bid, CT_B) using 64,622 bytes on BN256 within a single TXT record, whereas PriSrv requires nine TXT records (to transmit 531,996 bytes), reducing the mDNS packet size by 88.89%.

BLE's standard 31-byte payload challenges the transmission of large ciphertexts. Using the BLE Attribute Protocol (ATT) and PDU Segmentation, if the payload exceeds BLE's limit, it is segmented into multiple PDUs and reassembled by the receiver. PriSrv+ achieves an average 87.73% reduction for clients compared to PriSrv, enabling efficient segmentation and limited fragmentation.

Privacy-Enhanced EAP. PriSrv+ enhances privacy in EAP by integrating encrypted broadcasts and responses. An access point (AP) acts as a pass-through for interactions between the client and service provider. The provider broadcasts privacy-preserving service information, including a broadcast identifier and FEME-generated ciphertext, matching EAP's initial authentication request. A successful decryption by the client prompts an encrypted reply, which the provider decrypts and then responds with an authentication tag containing DH shares and a MAC. The client verifies the tag, computes a secret session key, and signals success, while the provider computes the same key for a secure session. EAP messages are then encapsulated in EAPOL frames and sent as RADIUS packets. Compared to PriSrv, PriSrv+ reduces communication costs by 86.64% during mutual authentication.

Privacy-Enhanced Apple AirDrop. AirDrop uses BLE to broadcast a hashed service provider identity for detecting nearby clients, followed by a TLS handshake that exposes identities via cleartext certificate exchange. Using the PrivateDrop mechanism [31], PriSrv+ improves 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. Apple may act as a key generation center, producing the necessary secret keys alongside existing iCloud certificates.

VII. Limitations and Future Work

While PriSrv+ introduces notable advancements in private service discovery, certain limitations remain.

Trusted Authority for Attribute Assignment. Although PriSrv+ eliminates credential issuance and revocation burdens of PriSrv, it still depends on a *trusted authority* for attribute assignment during registration—a common but centralized assumption in ABE systems. *Future work* could explore decentralized alternatives to reduce this trust dependency.

Revocation Mechanism. PriSrv+ does not include an integrated revocation scheme, but it can adopt existing ABE-based approaches such as *key updates*, *attribute expiration*, *proxy re-encryption*, and *server-assisted revocation*. *Future research* may focus on lightweight, scalable revocation strategies suited to dynamic service discovery settings.

Mitigating Client Abuse. Strong anonymity in PriSrv+ may allow clients to misuse services. Although not addressed natively, PriSrv+ can incorporate ABE-based *traitor tracing* techniques, such as embedding user-specific identifiers in keys or using traceable ciphertexts. *Key-insulated architectures* and dynamic revocation can further mitigate abuse. *Future work* should explore integrating such accountability mechanisms without compromising user privacy.

VIII. Conclusion

PriSrv+ significantly advances privacy-preserving service discovery in wireless networks by introducing Fast and Expressive Matchmaking Encryption (FEME). It overcomes the limitations of prior schemes by enabling expressive bilateral access control while enhancing efficiency, security, privacy, and usability. Evaluations demonstrate notable gains in performance and reduced communication overhead, making it well-suited for resource-constrained devices. With formal security guarantees and compatibility with existing wireless protocols, PriSrv+ effectively meets the privacy and security demands of wireless service discovery environments.

Acknowledgment

The authors thank the anonymous reviewers for their valuable comments and insightful suggestions.

This research is supported by the National Research Foundation, Singapore and Infocomm Media Development Authority under its Trust Tech Funding Initiative, AXA Research Fund, National Natural Science Foundation of China (No. 62372110, 62332007, U22B2028), Fujian Provincial Natural Science of Foundation (No. 2023J02008), Lee Kong Chian Chair Professorship, University of Oregon School of Law, Consumer Protection Research Grant 2025-2026 (under Grant No. 4236D0), National Science Foundation (No. 2112471), Werner Siemens-Stiftung (WSS) as part of the Centre for Cyber Trust (CEYT), Science and Technology Major Project of Tibetan Autonomous Region of China (No. XZ202201ZD0006G), Open Research Fund of Machine Learning and Cyber Security Interdiscipline Research Engineering Center of Jiangsu Province (No. SDGC2131), National Joint Engineering Research Center of Network Security Detection and Protection Technology, Guangdong Key Laboratory of Data Security and Privacy Preserving, Guangdong Hong Kong Joint Laboratory for Data Security and Privacy Protection, and Engineering Research Center of Trustworthy AI, Ministry of Education. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore and Infocomm Media Development Authority.

Ethics Considerations

This work presents PriSrv+, a cryptographic protocol for privacy-preserving service discovery. It does not involve human subjects, real-world deployments, or personal data—only synthetic datasets and simulated attributes were used. PriSrv+ is designed to enhance user privacy and prevent tracking and profiling attacks, without introducing or exploiting system vulnerabilities. All cryptographic techniques follow established models, and no responsible disclosure was required. We conducted this research in accordance with ethical principles outlined in the Menlo Report and believe it contributes positively to secure communications.

References

[1] ISO/IEC 29115:2013 - Entity authentication assurance framework. 2013.
[2] OECD Guidelines on the Protection of Privacy and Transborder Flows of Personal Data. 2013.
[3] Regulation (EU) 2016/679 - General Data Protection Regulation (GDPR). 2016.
[4] NIST Special Publication 800-63B - Digital Identity Guidelines. 2017.
[5] ETSI TS 103 465 V1.1.1 - Cyber Security for IoT. 2019.
[6] ETSI EN 303 645 V2.1.1 - Cyber Security for Consumer IoT. 2020.
[7] ISO/IEC 29184:2020 - Online privacy notices and consent. 2020.
[8] NIST Privacy Framework: A Tool for Improving Privacy Through Enterprise Risk Management. 2020.
[9] NIST Special Publication 800-53 - Security and Privacy Controls for Federal Information Systems. 2020.
[10] Wi-fi. https://w1.fi, 2023.
[11] Airdrop. https://support.apple.com/en-us/HT204144, 2024.
[12] Bluetooth. https://www.bluetooth.com, 2024.
[13] Vanadium. https://vanadium.github.io/, 2024.
[14] DNS-SD. RFC 6763. Dns-based service discovery, 2013.
[15] UPnP. RFC 6970. Universal plug and play (upnp) internet gateway device - port control protocol interworking function, 2013.
[16] Giuseppe Ateniese, Danilo Francati, David Nunez, and Daniele Venturi. Match me if you can: Matchmaking encryption and its applications. In *CRYPTO*, 2019.
[17] 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.
[18] Amos Beimel. Secure schemes for secret sharing and key distribution. *PhD thesis, Israel Institute of Technology, Technion*, 1996.
[19] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *IEEE S&P*, 2007.
[20] Ran Canetti and Hugo Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In *EUROCRYPT*, pages 453–474. Springer, 2001.
[21] Ran Canetti and Hugo Krawczyk. Security analysis of ike's signature-based key-exchange protocol. In *CRYPTO*, pages 143–161. Springer, 2002.
[22] Aldo Cassola, Erik-Oliver Blass, and Guevara Noubir. Authenticating privately over public wi-fi hotspots. In *CCS*, 2015.
[23] Melissa Chase. Multi-authority attribute based encryption. In *Theory of cryptography conference*, pages 515–534. Springer, 2007.
[24] Melissa Chase, Sarah Meiklejohn, and Greg Zaverucha. Algebraic macs and keyed-verification anonymous credentials. In *CCS*, 2014.
[25] Jie Chen, Yu Li, Jinming Wen, and Jian Weng. Identity-based matchmaking encryption from standard assumptions. In *ASIACRYPT*. Springer, 2022.
[26] A. Cooper and H. Tschofenig. Privacy Considerations for Internet Protocols. *RFC 6973*, 2013.
[27] SSDP. IETF draft-cai-ssdp-v1 03. Simple service discovery protocol/1.0 operating without on arbiter, 1999.
[28] S. Farrell and H. Tschofenig. Pervasive Monitoring Is an Attack. *RFC 7258*, 2014.
[29] Kassem Fawaz, Kyu-Han Kim, and Kang G Shin. Protecting privacy of ble device users. In *USENIX Security*, 2016.
[30] Danilo Francati, Alessio Guidi, Luigi Russo, and Daniele Venturi. Identity-based matchmaking encryption without random oracles. *INDOCRYPT*, 2021.
[31] Alexander Heinrich, Matthias Hollick, Thomas Schneider, Milan Stute, and Christian Weinert. Privatedrop: Practical privacy-preserving authentication for apple airdrop. In *USENIX Security*, 2021.
[32] Bastian Könings, Christoph Bachmaier, Florian Schaub, and Michael Weber. Device names in the wild: Investigating privacy risks of zero configuration networking. In *MDM*, 2013.
[33] Junzuo Lai, Robert H Deng, and Yingjiu Li. Expressive cp-abe with partially hidden access structures. In *Proceedings of the 7th ACM symposium on information, computer and communications security*, pages 18–19, 2012.
[34] mDNS. RFC 6762. Multicast dns, 2013.
[35] Long Meng, Liqun Chen, Yangguang Tian, and Mark Manulis. Fabesa: Fast (and anonymous) attribute-based encryption under standard assumption. In *CCS*, pages 4688–4702, 2024.

[36] Long Meng, Liqun Chen, Yangguang Tian, Mark Manulis, and Suhui Liu. Fease: Fast and expressive asymmetric searchable encryption. In *USENIX Security 24*, pages 2545–2562, 2024.
[37] PriSrv+. https://github.com/PriSrv-Plus, 2025.
[38] Doreen Riepel and Hoeteck Wee. Fabeo: Fast attribute-based encryption with optimal security. In *CCS*, 2022.
[39] Jacob T Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *Journal of the ACM*, 1980.
[40] Victor Shoup. Lower bounds for discrete logarithms and related problems. In *EUROCRYPT*, 1997.
[41] 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.
[42] 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.
[43] 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.
[44] 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.
[45] 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.
[46] Axin Wu, Weiqi Luo, Jian Weng, Anjia Yang, and Jinghang Wen. Fuzzy identity-based matchmaking encryption and its application. *IEEE TIFS*, 2023.
[47] David J Wu, Ankur Taly, Asim Shankar, and Dan Boneh. Privacy, discovery, and authentication for the internet of things. In *ESORICS*, 2016.
[48] Ningbin Yang, Chunming Tang, and Debiao He. A lightweight certificateless multi-user matchmaking encryption for mobile devices: Enhancing security and performance. *IEEE TIFS*, 2023.
[49] 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).
[50] Yang Yang, Guomin Yang, Yingjiu Li, Rui Shi, Minming Huang, Jian Weng, HweeHwa Pang, and Robert H Deng. Prisrv+: Privacy and usability-enhanced wireless service discovery with fast and expressive matchmaking encryption (full version). *Cryptology ePrint Archive*, 2025. https://eprint.iacr.org/2025/1584.
[51] 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.
[52] Richard Zippel. Probabilistic algorithms for sparse polynomials. In *ISSAC*, 1979.

APPENDIX A
ARTIFACT APPENDIX

A. Description & Requirements

This artifact includes the source code, benchmark suite, and experimental scripts used to evaluate the PriSrv+ protocol and its core component, FEME (Fast and Expressive Matchmaking Encryption). It enables the reproduction of key results from Section VI of the paper, demonstrating the performance advantages of PriSrv+. All necessary components, including source code, detailed documentation, and synthetic datasets, are available in a publicly accessible repository.

1) How to access:

- **Repository:** https://github.com/PriSrv-Plus
- **DOI:** To be assigned upon camera-ready submission

2) Hardware dependencies:

- CPU: Intel i5/i7/i9 or equivalent, 4-core minimum
- RAM: 8 GB minimum, 16 GB recommended
- Storage: 1 GB

3) Software dependencies:

- Operating System: Linux (Ubuntu 20.04+ recommended)
- Programming Languages: C/C++
- Libraries: MIRACL cryptographic library (included)
- Tools: gcc/g++, make

4) Benchmarks: Synthetic attribute sets and access policies are included to reproduce the benchmarks demonstrating PriSrv+'s performance efficiency. Generation scripts for these benchmarks are provided.

B. Artifact Installation & Configuration

1) Clone the repository: [xleftmargin=-12mm] git clone https://github.com/PriSrv-Plus.git cd PriSrv-Plus
2) Install dependencies: [xleftmargin=-12mm] sudo apt-get install build-essential
3) Build the artifact: [xleftmargin=-12mm] make algorithm

C. Experiment Workflow

Execute the provided benchmark and test scripts to reproduce the results: [xleftmargin=4.5mm] make test

D. Major Claims

- (C1): FEME achieves efficient encryption and decryption performance as evidenced by Section VI (Table III).
- (C2): PriSrv+ demonstrates efficient privacy-preserving broadcasts and anonymous mutual authentication, confirmed through experiments detailed in Section VI (Table IV, Figure 7).

E. Evaluation

1) Experiment (E1): FEME: Performance Evaluation [30 human-minutes + 1 compute-hour]

[How to]: Validate performance results of FEME.

[Preparation]: Install and configure the artifact as described above.

[Execution]: Run the benchmark script: [xleftmargin=10mm] make test

[Results]: Review terminal outputs for encryption and decryption metrics.

2) Experiment (E2): PriSrv+: Privacy-Preserving Broadcast and Anonymous Mutual Authentication [30 human-minutes + 1 compute-hour]

[How to]: Validate performance results.

[Preparation]: Use the configuration from previous steps.

[Execution]: Execute the provided privacy broadcast scripts: [xleftmargin=10mm] make test

[Results]: Review terminal outputs for broadcast and mutual authentication metrics.

F. Artifact Availability

The full artifact (source code, benchmarks, scripts, and documentation) is permanently archived on Zenodo:
https://doi.org/10.5281/zenodo.16945935

APPENDIX B
SECURITY MODEL OF FEME

The security models for FEME outline 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.

Definition 3. *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* $\mathsf{Adv}^{\mathsf{conf}}_{\mathcal{FEME}}(\lambda) \stackrel{\mathsf{def}}{=}$

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

$\leqslant \nu(\lambda)$, *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 4. *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* $\mathsf{Adv}^{\mathsf{anon}}_{\mathcal{FEME}}(\lambda) \stackrel{\mathsf{def}}{=}$

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

$\leqslant \nu(\lambda)$, *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 5. *A FEME scheme $\mathcal{FEME}$ satisfies authenticity if for any PPT adversary $\mathcal{A}$, there exists a negligible function ν such that* $\mathsf{Adv}^{\mathsf{auth}}_{\mathcal{FEME}}(\lambda) \stackrel{\mathsf{def}}{=}$

$$\mathsf{Pr}\left[\begin{array}{l} (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda) \\ (\mathsf{CT}_{\mathsf{snd}}, \mathbb{A}_{\mathsf{rcv}}, \mathcal{S}_{\mathsf{rcv}}) \leftarrow \mathcal{A}^{\mathcal{O}_1,\mathcal{O}_2,\mathcal{O}_3}(\mathsf{mpk}) \\ \mathsf{DK}_{\mathcal{S}_{\mathsf{rcv}}} \leftarrow \mathsf{DKGen}(\mathsf{msk}, \mathcal{S}_{\mathsf{rcv}}) \\ \mathsf{SK}_{\mathbb{A}_{\mathsf{rcv}}} \leftarrow \mathsf{PolGen}(\mathsf{msk}, \mathbb{A}_{\mathsf{rcv}}) \\ \mathsf{msg} = \mathsf{Dec}(\mathsf{DK}_{\mathcal{S}_{\mathsf{rcv}}}, \mathsf{SK}_{\mathbb{A}_{\mathsf{rcv}}}, \mathsf{CT}_{\mathsf{snd}}) \\ \forall \mathcal{S}_{\mathsf{snd}} \in \mathcal{Q}_{\mathcal{O}_1} : (\mathcal{S}_{\mathsf{snd}} \nVdash \mathbb{A}_{\mathsf{rcv}}) \bigwedge (\mathsf{msg} \neq \perp) \end{array} \right]$$

$\leqslant \nu(\lambda)$, *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 $\mathcal{S} \nVdash \mathbb{A}$ denotes that attributes $\mathcal{S}$ does not satisfy $\mathbb{A}$.*

APPENDIX C
SECURITY PROOFS OF FEME

Generic Group Model (GGM). We use an extended GGM for bilinear groups, as outlined in [40]. 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 $\mathcal{L}_H$.

A. Proof of Theorem 1 (FEME: Confidentiality)

Proof. Our proof close follows the proof structure in [19]. We start with a standard observation derived from a basic 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

1	κ	t_i	$t_i\tau$	t_ir	t_is_2	$x+\kappa\tau$	$\kappa\lambda_i+t_\pi s_3$	$t_i\tau_1 s_3$	$(x+\kappa\tau_1)s_3$	$\frac{1}{b_1}(\lambda_i+t_\rho r')$	$\frac{1}{b_2}(\lambda_i+t_\rho r')$	$\frac{1}{b_1}(\kappa\psi_i+t_\rho r')$	$\frac{1}{b_2}(\kappa\psi_i+t_\rho r')$
μ	$\mu\kappa$	μt_i	$\mu t_i\tau$	μt_ir	μt_is_2	$\mu(x+\kappa\tau)$	$\mu(\kappa\lambda_i+t_\pi s_3)$	$\mu t_i\tau_1 s_3$	$\mu(x+\kappa\tau_1)s_3$	$\frac{\mu}{b_1}(\lambda_i+t_\rho r')$	$\frac{\mu}{b_2}(\lambda_i+t_\rho r')$	$\frac{\mu}{b_1}(\kappa\psi_i+t_\rho r')$	$\frac{\mu}{b_2}(\kappa\psi_i+t_\rho r')$
b_1	$b_1\kappa$	b_1t_i	$b_1t_i\tau$	b_1t_ir	$b_1t_is_2$	$b_1(x+\kappa\tau)$	$b_1(\kappa\lambda_i+t_\pi s_3)$	$b_1t_i\tau_1 s_3$	$b_1(x+\kappa\tau_1)s_3$	$\lambda_i+t_\rho r'$	$\frac{b_1}{b_2}(\lambda_i+t_\rho r')$	$(\kappa\psi_i+t_\rho r')$	$\frac{b_1}{b_2}(\kappa\psi_i+t_\rho r')$
b_2	$b_2\kappa$	b_2t_i	$b_2t_i\tau$	b_2t_ir	$b_2t_is_2$	$b_2(x+\kappa\tau)$	$b_2(\kappa\lambda_i+t_\pi s_3)$	$b_2t_i\tau_1 s_3$	$b_2(x+\kappa\tau_1)s_3$	$\frac{b_2}{b_1}(\lambda_i+t_\rho r')$	-	$\frac{b_2}{b_1}(\kappa\psi_i+t_\rho r')$	-
r	$r\kappa$	rt_i	$rt_i\tau$	t_ir^2	rt_is_2	$r(x+\kappa\tau)$	$r(\kappa\lambda_i+t_\pi s_3)$	$rt_i\tau_1 s_3$	$r(x+\kappa\tau_1)s_3$	$\frac{r}{b_1}(\lambda_i+t_\rho r')$	$\frac{r}{b_2}(\lambda_i+t_\rho r')$	$\frac{r}{b_1}(\kappa\psi_i+t_\rho r')$	$\frac{r}{b_2}(\kappa\psi_i+t_\rho r')$
r'	$r'\kappa$	$r't_i$	$r't_i\tau$	$t_ir'r$	$r't_is_2$	$r'(x+\kappa\tau)$	$r'(\kappa\lambda_i+t_\pi s_3)$	$r't_i\tau_1 s_3$	$r'(x+\kappa\tau_1)s_3$	$\frac{r'}{b_1}(\lambda_i+t_\rho r')$	$\frac{r'}{b_2}(\lambda_i+t_\rho r')$	$\frac{r'}{b_1}(\kappa\psi_i+t_\rho r')$	$\frac{r'}{b_2}(\kappa\psi_i+t_\rho r')$
s_1	$s_1\kappa$	s_1t_i	$s_1t_i\tau$	s_1t_ir	$s_1t_is_2$	$s_1(x+\kappa\tau)$	$s_1(\kappa\lambda_i+t_\pi s_3)$	$s_1t_i\tau_1 s_3$	$s_1(x+\kappa\tau_1)s_3$	$\frac{s_1}{b_1}(\lambda_i+t_\rho r')$	$\frac{s_1}{b_2}(\lambda_i+t_\rho r')$	$\frac{s_1}{b_1}(\kappa\psi_i+t_\rho r')$	$\frac{s_1}{b_2}(\kappa\psi_i+t_\rho r')$
s_3	$s_3\kappa$	s_3t_i	$s_3t_i\tau$	s_3t_ir	$s_3t_is_2$	$s_3(x+\kappa\tau)$	$s_3(\kappa\lambda_i+t_\pi s_3)$	$t_i\tau_1 s_3^2$	$(x+\kappa\tau_1)s_3^2$	$\frac{s_3}{b_1}(\lambda_i+t_\rho r')$	$\frac{s_3}{b_2}(\lambda_i+t_\rho r')$	$\frac{s_3}{b_1}(\kappa\psi_i+t_\rho r')$	$\frac{s_3}{b_2}(\kappa\psi_i+t_\rho r')$
$b_1\tau$	$b_1\tau\kappa$	$b_1\tau t_i$	$b_1\tau t_i\tau$	$b_1\tau t_ir$	$b_1\tau t_is_2$	$(x+\kappa\tau)$	$b_1\tau(\kappa\lambda_i+t_\pi s_3)$	$b_1\tau t_i\tau_1 s_3$	$b_1\tau(x+\kappa\tau_1)s_3$	$\tau(\lambda_i+t_\rho r')$	$\frac{b_1\tau}{b_2}(\lambda_i+t_\rho r')$	$\tau(\kappa\psi_i+t_\rho r')$	$\frac{b_1\tau}{b_2}(\kappa\psi_i+t_\rho r')$
$b_2\tau$	$b_2\tau\kappa$	$b_2\tau t_i$	$b_2\tau t_i\tau$	$b_2\tau t_ir$	$b_2\tau t_is_2$	$(x+\kappa\tau)$	$b_2\tau(\kappa\lambda_i+t_\pi s_3)$	$b_2\tau t_i\tau_1 s_3$	$b_2\tau(x+\kappa\tau_1)s_3$	$\frac{b_2\tau}{b_1}(\lambda_i+t_\rho r')$	-	$\frac{b_2\tau}{b_1}(\kappa\psi_i+t_\rho r')$	-
b_1s_2'	$b_1s_2'\kappa$	$b_1s_2't_i$	$b_1s_2't_i\tau$	$b_1s_2't_ir$	$b_1s_2't_is_2$	$b_1s_2'(x+\kappa\tau)$	$b_1s_2'(\kappa\lambda_i+t_\pi s_3)$	$b_1s_2't_i\tau_1 s_3$	$b_1s_2'(x+\kappa\tau_1)s_3$	$s_2'(\lambda_i+t_\rho r')$	$\frac{b_1s_2'}{b_2}(\lambda_i+t_\rho r')$	$s_2'(\kappa\psi_i+t_\rho r')$	$\frac{b_1s_2'}{b_2}(\kappa\psi_i+t_\rho r')$
b_2s_2''	$b_2s_2''\kappa$	$b_2s_2''t_i$	$b_2s_2''t_i\tau$	$b_2s_2''t_ir$	$b_2s_2''t_is_2$	$b_2s_2''(x+\kappa\tau)$	$b_2s_2''(\kappa\lambda_i+t_\pi s_3)$	$b_2s_2''t_i\tau_1 s_3$	$b_2s_2''(x+\kappa\tau_1)s_3$	$\frac{b_2s_2''}{b_1}(\lambda_i+t_\rho r')$	$s_2''(\lambda_i+t_\rho r')$	$\frac{b_2s_2''}{b_1}(\kappa\psi_i+t_\rho r')$	$s_2''(\kappa\psi_i+t_\rho r')$
$\alpha+\kappa r$	$(\alpha+\kappa r)\kappa$	$(\alpha+\kappa r)t_i\tau$		$(\alpha+\kappa r)t_is_2$	$(\alpha+\kappa r)(\kappa\lambda_i+t_\pi s_3)$	$(\alpha+\kappa r)\cdot$	$\frac{\alpha+\kappa r}{b_1}(\lambda_i+t_\rho r')$		$\frac{\alpha+\kappa r}{b_1}(\kappa\psi_i+t_\rho r')$				
α	$(\alpha+\kappa r)t_i$	$(\alpha+\kappa r)t_ir$		$(\alpha+\kappa r)(x+\kappa\tau)$	$(\alpha+\kappa r)t_i\tau_1 s_3$	$(x+\kappa\tau_1)s_3$	$\frac{\alpha+\kappa r}{b_2}(\lambda_i+t_\rho r')$		$\frac{\alpha+\kappa r}{b_2}(\kappa\psi_i+t_\rho r')$				
$b_1\tau_1s_3'$	$b_1\tau_1s_3'\kappa$	$b_1\tau_1s_3't_i\tau$		$b_1\tau_1s_3't_ir$	$b_1\tau_1s_3't_is_2$	$b_1\tau_1s_3'\cdot$	$\tau_1s_3'(\lambda_i+t_\rho r')$		$\tau_1s_3'(\kappa\psi_i+t_\rho r')$				
$x\mu$	$b_1\tau_1s_3't_i$	$b_1\tau_1s_3't_ir$		$b_1\tau_1s_3'(x+\kappa\tau)$	$b_1\tau_1^2s_3't_is_3$	$(x+\kappa\tau_1)s_3$	$\frac{b_1\tau_1s_3'}{b_2}(\lambda_i+t_\rho r')$		$\frac{b_1\tau_1s_3'}{b_2}(\kappa\psi_i+t_\rho r')$				
$b_2\tau_1s_3''$	$b_2\tau_1s_3''\kappa$	$b_2\tau_1s_3''t_i\tau$		$b_2\tau_1s_3''t_ir$	$b_2\tau_1s_3''t_is_2$	$b_2\tau_1s_3''\cdot$	$\frac{b_2\tau_1s_3''}{b_1}(\lambda_i+t_\rho r')$		$\frac{b_2\tau_1s_3''}{b_1}(\kappa\psi_i+t_\rho r')$				
$\alpha s+x\mu s_3$	$b_2\tau_1s_3''t_i$	$b_2\tau_1s_3''t_ir$		$b_2\tau_1s_3''(x+\kappa\tau)$	$b_2\tau_1s_3''t_i\tau_1s_3$	$(x+\kappa\tau_1)s_3$	$\tau_1s_3''(\lambda_i+t_\rho r')$		$\tau_1s_3''(\kappa\psi_i+t_\rho r')$				

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

challenge becomes independent of msg_0 and msg_1, giving $\mathcal{A}_2$ an advantage of 0. Consequently, we have

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

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

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

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

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

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

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

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

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 msg_0^*, 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:

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

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

$$\mathsf{ct}_{3,i}=g_1^{\kappa\lambda_i+t_{\pi(i)}s_3},\ \mathsf{ct}_{4,1}=g_2^{b_1s_2'},\ \mathsf{ct}_{4,2}=g_2^{b_2s_2''},$$

$$\mathsf{ct}_{5,i}=g_1^{t_is_2},\ \mathsf{ct}_{6,i}=g_1^{t_i\tau_1s_3},\ \mathsf{ct}_7=g_2^{b_1\tau_1s_3'},$$

$$\mathsf{ct}_8=g_2^{b_2\tau_1s_3''},\ \mathsf{ct}_9=g_1^{(x+\kappa\tau_1)s_3},\ \text{where}\ \tau_1=\tau+\tau'.$$

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

• ***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 V. Note that the blue element $\alpha s + x\mu s_3$ in Table V 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 V, 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 V, $\mathcal{A}$ 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 V, $\mathcal{A}$ 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 V, 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 $x s_3$ 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 $\mathbb{G}_T$.

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 V, the polynomial's maximum degree is 5. By the Schwartz-Zippel lemma [39], [52], 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. ■

Due to length limitation, the proofs of Theorems 2-3 are deferred to the full version [50].