PriSrv: Privacy-Enhanced and Highly Usable Service Discovery in Wireless Communications

Yang Yang[1], Robert H. Deng[1], Guomin Yang[1], Yingjiu Li[2], HweeHwa Pang[1], Minming Huang[1], Rui Shi[3], Jian Weng[4]
1. School of Computing and Information Systems, Singapore Management University, Singapore
2. Department of Computer Science, University of Oregon, USA
3. Beijing Electronic Science and Technology Institute, Beijing, China
4. College of Information Science and Technology, Jinan University, Guangzhou, China

***Abstract*—Service discovery is essential in wireless communications. However, existing service discovery protocols provide no or very limited privacy protection for service providers and clients, and they often leak sensitive information (e.g., service type, client's identity and mobility pattern), which leads to various network-based attacks (e.g., spoofing, man-in-the-middle, identification and tracking). In this paper, we propose a private service discovery protocol, called PriSrv, which allows a service provider and a client to respectively specify a fine-grained authentication policy that the other party must satisfy before a connection is established. PriSrv consists of a private service broadcast phase and an anonymous mutual authentication phase with bilateral control, where the private information of both parties is hidden beyond the fact that a mutual match to the respective authentication policy occurred. As a core component of PriSrv, we introduce the notion of anonymous credential-based matchmaking encryption (ACME), which exerts dual-layer matching in one step to simultaneously achieve bilateral flexible policy control, selective attribute disclosure and multi-show unlinkability. As a building block of ACME, we design a fast anonymous credential (FAC) scheme to provide constant size credentials and efficient show/verification mechanisms, which is suitable for privacy-enhanced and highly usable service discovery in wireless networks.**

We present a concrete PriSrv protocol that is interoperable with popular wireless communication protocols, such as Wi-Fi Extensible Authentication Protocol (EAP), mDNS, BLE and Airdrop, to offer privacy-enhanced protection. We present formal security proof of our protocol and evaluate its performance on multiple hardware platforms: desktop, laptop, mobile phone and Raspberry Pi. PriSrv accomplishes private discovery and secure connection in less than 0.973 s on the first three platforms, and in less than 2.712 s on Raspberry Pi 4B. We also implement PriSrv into IEEE 802.1X in the real network to demonstrate its practicality.

I. Introduction

Service discovery (SD) protocols, such as Wi-Fi [1], AirDrop [2], and BLE [3], are essential components of networking systems that enable devices and services to dynamically discover and communicate with each other in a network environment. They facilitate the automatic detection and advertisement of available services, making it easier for devices to locate and interact with desired resources. However, there is a lack of highly usable approaches to sufficiently protect identification and private information in protocol executions, especially for privacy-concerned parties. A survey [4] showed that about 90% users considered the exposure of device names from wireless network services as a privacy risk, as such exposure may lead to adversarial inference of users' private information such as mobility patterns, profiles, and locations [5], [6], [7], [8]. For instance, in public Wi-Fi, ISP could easily identify a person via the announced device names [9]. In IoT networks, an attacker may infer a user's regular routine by collecting the service data from user's smart devices [10]. Several vulnerabilities spanning from Wi-Fi, BLE to Apple Wireless Direct Link (AWDL) are discovered in recently years which lead to tracking, DoS, and MitM attacks on iOS and macOS [8].

On the other hand, users prefer high usability in accessing wireless network services, which include no pre-registered pairing, no third-party dependence for service discovery, and low computation and communication overheads. A major barrier in increasing user satisfaction for accessing wireless network services is the technical difficulty of elevating privacy protection without sacrificing high usability in wireless network protocols. Existing privacy-aware wireless network protocols and other related works fail to overcome this barrier as they either leak private information [11], [12], [13] or violate high usability requirements in protocol executions [14].

Our objective is to develop a privacy-enhanced and highly usable service discovery protocol between wireless network service access point (service provider) and client to enable them to discover each other within range and establish a secure communication channel only if they meet each other's connection requirements. The challenges to achieve this objective are three folds: (1) ensure that services are only discoverable by an authorized set of clients; (2) enable clients to filter out unauthorized services without heavy computation; (3) allow both service provider and client to specify policies the other party must satisfy in order for their private information to be revealed. In certain service discovery protocols, such as AirDrop and BLE, both service provider and client are wireless devices, which necessitates reciprocal privacy protection.

We propose a dual-layer architecture to solve this problem, which includes an outer layer and an inner layer. In the outer layer, each service provider or client is associated with a set of

This is the full version of the research work published in Network and Distributed System Security (NDSS) Symposium 2024.
https://dx.doi.org/10.14722/ndss.2024.24174

public attributes (such as domain name) that can be revealed to everyone and a public authentication policy, which are used for fast bilateral policy matching without decryption. Specifically, a service provider broadcasts a ciphertext encrypted by its policy and public attributes. A client first checks whether its public attributes match with the service provider's public policy, and vice versa, which filters the mismatch services accordingly. If and only if their public attributes satisfy the bilateral policy, the client can decrypt the ciphertext. In the inner layer, each party is associated with a set of private attributes (such as device ID) that are only revealed to the intended peers. Only when the decryption of the outer layer ciphertext is successful, the client can recover the private attributes of the service provider, which allows the client to authenticate the service provider by verifying the authenticity of the latter's attributes, including both public attributes and private attributes. The service provider authenticates the client using the same mechanism. Then, they establish a session key using a secure key agreement protocol to enable secure communication between them. By applying the above dual-layer architecture, PriSrv builds a private-enhanced service discovery protocol with high usability.

A. *Privacy Enhancement and High Usability Requirements*

To mitigate the leakage of any private information in service discovery, SD protocols should meet the following privacy enhancement requirements.

1. Private Service Broadcast. Service contents broadcasted by service providers must be both confidential and unforgeable, preventing unintended clients from learning service content and enabling the detection of bogus service providers broadcasting fraudulent services.

2. Mutual Authentication. Service providers and clients authenticate each other in a secure manner to ensure that the private information of both parties will not be leaked to any unauthenticated entity.

3. Bilateral Anonymity. Both service providers and clients remain anonymous to a third-party during protocol execution, and no third-party can identify the private information of the involved parties.

4. Bilateral Flexible Policy Control. Both service providers and clients can specify fine-grained access policies for authorized peers and simultaneously check the satisfaction of policies from both sides, which guarantees that private information of both sides are only exposed to their authorized peers.

5. Selective Attribute Disclosure. It refers to the ability of an entity (either service provider or client) to choose which specific attributes they disclose to the other, while keeping other attributes undisclosed. It allows each entity to share only the necessary and relevant information while maintaining control over their private information.

6. Multi-Show Unlinkability. It allows a user to prove possession of a credential or attributes without revealing their identity or linking their actions across multiple sessions.

In addition to these privacy enhancement requirements, SD protocols are expected to meet the following high usability requirements.

1. No Pre-registered Pairing. Clients are not required to subscribe to or share a secret key with any service providers beforehand. It allows clients to discover and connect to service providers seamlessly without any manual setup or configuration.

2. No Third-party Dependency during Service Discovery Process. Service discovery should not depend on any external services such as a third-party server or a directory provider during protocol execution. Protocols relying on external servers presume a reliable Internet connection for mobile devices. However, this presumption may not hold in wireless communications (e.g., BLE communications).

3. No In-advance Identity Issuance. Users are not required to register to a third-party to obtain identity certification documents, such as certificates, credentials, etc. In-advance identity issuance has less impact on the usability of service discovery process since it occurs only once before the execution of SD protocol. We note that PriSrv requires in-advance identity issuance.

B. *Contributions*

We propose PriSrv, a service discovery protocol, to meet both privacy enhancement and high usability requirements. The main contributions of this work are summarized as follows.

- **A New Privacy-Enhanced Service Discovery Protocol with High Usability**. PriSrv is the first privacy-enhanced and highly usable service discovery protocol that can be integrated into a wide range of wireless applications.
- **Anonymous Credential-based Matchmaking Encryption (ACME)**. We propose a novel cryptographic primitive called anonymous credential-based matchmaking encryption (ACME). ACME supports bilateral fine-grained policies and selective attribute disclosure for private mutual authentication in service discovery. ACME outperforms the matchmaking encryption (ME) in CRYPTO'19 [15] in terms of functionality and efficiency. This is a contribution of independent interest for the advancement of matchmaking encryption.
- **Fast Anonymous Credential**. As a building block of ACME, we propose a fast anonymous credential (FAC) scheme to support anonymous authentication with selective attribute disclosure and multi-show unlinkability. A comprehensive comparison with existing anonymous credential schemes demonstrates its superior efficiency for credential showing and verification with constant and small credential.
- **Interoperability with Existing Protocols**. To demonstrate interoperability, we present concrete methods for integrating PriSrv with mainstream service discovery protocols including Extensible Authentication Protocol (EAP), mDNS, BLE and AirDrop. Through experimentation, we show the applicability and effectiveness of PriSrv in real-world scenarios.
- **Formal Security Proofs**. We provide formal security proofs for the security and privacy properties of PriSrv in

a security model that captures various attack vectors, such as intercepting, tampering with channel messages, replaying, injecting data packets, and interleaving messages among different sessions in realistic settings.

• **Deployment on Multiple Platforms in Real Networks**: The performance of PriSrv is evaluated on multiple hardware platforms, including desktop, laptop, mobile phone and Raspberry Pi, in the Wi-Fi WPA-Enterprise framework. Our experiments demonstrate the efficiency of PriSrv across different platforms. The private service broadcast phase in PriSrv takes less than 0.483 seconds, and the anonymous mutual authentication phase takes less than 0.973 seconds on the first three devices. The delays stay well below 1 second, which humans perceive as an "immediate response" [16], [17]. While on Raspberry Pi, the delays are 1.189 and 2.712 seconds for private broadcast and mutual authenticationon, respectively, which demonstrates additional costs on IoT devices.

II. Related Work

A variety of protocols have been developed for service discover in network environments. As shown in Table I, none of them, except PriSrv, satisfy all privacy enhancement requirements.

In particular, the protocols DNS-SD [18], mDNS [19], SSDP [20], UPnP [21] and CBN [9] do not meet any privacy enhancement requirement. First, DNS-based Service Discovery (DNS-SD) [18] utilizes the Domain Name System (DNS) to enable service discovery. It allows service providers to advertise their services by registering them with a DNS server, and clients can discover these services by querying the DNS server, which is widely used in local networks and the Internet. Second, multicast DNS (mDNS) [19] enables service discovery in local networks without the need for a central DNS server, and allows service providers to announce their services using multicast DNS packets, and clients can resolve and discover these services directly. Third, Simple Service Discovery Protocol (SSDP) [20] is designed based on the Internet protocol suite for advertisement and discovery of network services and presence information. Fourth, Universal Plug and Play (UPnP) [21] permits networked devices, such as personal computers, printers, Internet gateways, Wi-Fi access points and mobile devices to seamlessly discover each other's presence on the network and establish functional network services. Lastly, CBN scheme [9] requires clients to subscribe to service providers so that service providers can unilaterally authenticate clients anonymously for service discovery.

The above SD protocols are vulnerable to man-in-the-middle (MitM) attacks, spoofing attacks and denial-of-service (DoS) attacks due to the lack of proper privacy protection. Bai et al. [12] launched MitM attacks against mDNS and illustrated how a malicious device can impersonate a printer by spoofing its mDNS hostname. According to Wang et al. [22], UPnP is vulnerable to DoS attacks: a device receiving a request from a potentially spoofed control point may respond to the supposed requester, unknowingly contributing to the amplification and intensification of the attack. CBN scheme [9] only requires clients to anonymously authenticate to service providers in a private manner, while the authentication/anonymity of service providers and private broadcast are not supported, making it vulnerable to MitM attacks and spoofing attacks.

Among these protocols, DNS-SD relies on DNS records to advertise and discover services within a network. CBN scheme relies on a pre-registration pairing mechanism: service provider maintains a directory to control the access of subscribers while clients are required to register to service providers beforehand, where the size of directory grows linearly with the number of clients.

Although Wi-Fi [1] and BLE [3] support mutual authentication, they dissatisfy other privacy enhancement requirements, including private broadcast, bilateral anonymity, bilateral flexible policy control, selective attribute disclosure and multi-show unlinkability. Wi-Fi [1] enables devices to discover and connect to services available on a local-area network. Bluetooth Low Energy (BLE) [3] is designed for low-power devices, such as IoT devices and wearable devices, to advertise their available services, allowing other devices to discover and connect to them for data exchange and interaction.

A common problem of Wi-Fi and BLE is that the private information of service providers and clients is advertised publicly in wireless network, which may induce user identification, impersonation attacks and spoofing attacks. A survey [4] indicated that 59% investigated devices periodically announce their owners' real names for Wi-Fi network, which is deemed as a privacy risky by about 90% users. A deep-learning-based identification mechanism (with accuracy over 80%) was demonstrated in [23] to identify mobile devices from broadcast and multicast packets. Na et al. [11] proposed Wi-attack to leverage the wide-deployed Wi-Fi devices (such as Wi-Fi APs) to conduct poisonous impersonation attacks, where the vulnerability is caused by the open nature of these cleartext advertisements. Similarly, BLE-equipped devices consistently advertise their unique identifiers in cleartext [10], making them vulnerable to BLE Spoofing Attacks (BLESA) [24].

Revealing of device identifiers in Wi-Fi and BLE is a stepping stone toward advanced attacks such as user profiling and tracking [10]. Large-scale tracking attack in real-time can be mounted by deploying multiple low-cost Wi-Fi and BLE nodes throughout an area. This allows adversaries to infer additional user information such as home and work locations, movement patterns and behavior profiling, which are useful for targeted tracking [25].

AirDrop [2], PrivateDrop [16] and WTSB [5] employ encryption and authentication mechanisms to protect communications in service discovery. AirDrop [2] is an SD protocol for file-sharing on Apple devices, which utilizes a combination of Wi-Fi and Bluetooth technologies to enable devices in close proximity to discover each other and share files wirelessly. AirDrop and PrivateDrop need to establish TLS connection with client and server certificates for authentication. PrivateDrop realizes *private* mutual authentication for AirDrop by protecting device identifiers in an optimized private set intersection

SD Protocols	**Privacy Enhancement**						**High Usability**		
	Private Broadcast	Mutual Authn.	Bilateral Anon.	Bilateral Flex. Pol. Ctrl.	Sel. Attr. Disclosure	Multi-Show Unlinkability	No Pre-reg. Pairing	No 3rd-party Dependence	No In-advance ID Issuance
DNS-SD [18]	×	×	×	×	×	×	√	×	×
mDNS [19]	×	×	×	×	×	×	√	√	×
SSDP [20]	×	×	×	×	×	×	√	√	√
UPnP [21]	×	×	×	×	×	×	√	√	√
Wi-Fi [1]	×	√	×	×	×	×	√	√	×
BLE [3]	×	√	×	×	×	×	√	√	√
AirDrop [2]	×	√	×	×	×	×	√	√	×
PrivateDrop [16]	×	√	√	×	×	×	√	√	×
CBN [9]	×	×	×	×	×	×	×	√	×
WTSB [5]	√	√	√	×	×	×	√	√	×
PriSrv	√	√	√	√	√	√	√	√	×

TABLE I: Comparison of Service Discovery Protocols

protocol [16]. WTSB [5] realizes private service discovery by leveraging prefix encryption (a variant of identity-based encryption) and standard digital signature-based key exchange protocol. WTSB [5] supports private broadcast, mutual authentication and bilateral anonymity.

However, these SD protocols (AirDrop, PrivateDrop [16] and WTSB [5]) suffer from MitM attacks, DoS attacks, impersonation attacks or user tracking attacks due to the lack of privacy enhanced properties, such as bilateral policy control, selective attribute disclosure and multi-show unlinkability. The attacker is able to link multiple sessions using client and server certificates in AirDrop and PrivateDrop protocols. Stute et al. [7] exposed several security and privacy vulnerabilities in Apple Wireless Direct Link (AWDL) ranging from design flaws to implementation bugs leading to (i) MitM attacks enabling stealthy modification of files transmitted via AirDrop, (ii) DoS attacks disrupting communications, and (iii) privacy leaks enabling user identification and long-term tracking. Bai et al. [12] demonstrated impersonation and spoofing attacks on certain Zeroconf protocols (e.g. AirDrop), which even allows attackers to steal clients' SMS messages, documents, email notifications and photos [13].

Heinrich et al. [16] discovered a series of flaws in AirDrop that allow attackers to learn phone numbers and email addresses of both sender and receiver devices. As stated in the work [16], users of PrivateDrop can be tracked via UUIDs in the TLS certificates used for establishing the protocol communication channels. WTBS [5] dissatisfies bilateral flexible policy control: service providers have the ability to specify the type of clients they intend to communicate with, but clients do not have the option to choose the service providers they want to communicate with. Furthermore, WTBS [5] is susceptible to user tracking attack due to the lack of multi-show unlinkability.

AirDrop and PrivateDrop offer mutual authentication, while PrivateDrop provides an additional feature of bilateral anonymity. The fundamental building block of PrivateDrop is a Diffie-Hellman-based Private Set Intersection (PSI) scheme, which exclusively entails exponentiation computations. Moreover, WTBS [5] further enhances privacy by encrypting broadcast messages, achieving private broadcasting in addition to these features. WTSB has the advantage of high efficiency due to the usage of efficient identity-based prefix encryption scheme. Conversely, PriSrv utilizes both exponentiation and bilinear pairing operations within ACME, and the time consumption increases with the complexity of access policy. Therefore, PriSrv achieves improved privacy but incurs a higher computational overhead as a trade-off.

After conducting a comprehensive comparison, it becomes evident that PriSrv stands out as the only SD protocol that successfully meets all the privacy enhancement requirements. As for high usability, PriSrv satisfies no pre-registerd pairing and no third-party dependence during service discovery. PriSrv does require in-advance identity issuance, but it does not affect the service discovery process.

III. Preliminaries

A. Notation and Bilinear Pairing

Let $\vec{x}$ denote the full attribute set, $\vec{x}^{(in)}$ the private attributes for an inner layer and $\vec{x}^{(out)}$ the public attributes for an outer layer, where $\vec{x}^{(in)}, \vec{x}^{(out)} \subseteq \vec{x}$. Let $f : \{0,1\}^n \to \{0,1\}$ denote the policy; $f(\vec{x}) = 1$ denote $\vec{x}$ satisfying f, and $f(\vec{x}) = 0$ denote $\vec{x}$ not satisfying f.

Let $s \overset{\$}{\leftarrow} S$ denote s sampled uniformly at random from a set S; $\mathbb{N}$ denote the natural number; $\lambda \in \mathbb{N}$ denote the security parameter; $[n_1, n_2]$ denote $\{n_1, \cdots, n_2\}$; PPT denote probabilistic polynomial time; $\mathbb{Z}_p$ represent the group of integers modulo p, and $\mathbb{Z}_p^* = \mathbb{Z}_p \backslash \{0\}$. We use lower case boldface to denote (column) vectors and upper case boldface to denote matrices. Denote a bilinear group with Type-3 pairings as $\mathcal{BG} = (G_1, G_2, G_T, e, p)$, where there is no efficiently computable isomophism between G_1 and G_2. Let $g_1 \in G_1$, $g_2 \in G_2$ and $g_T = e(g_1, g_2) \in G_T$ be the respective generators. For a matrix $\mathbf{A}$ over $\mathbb{Z}_p$, define $[\mathbf{A}]_1 := g_1^{\mathbf{A}}$, $[\mathbf{A}]_2 := g_2^{\mathbf{A}}$, $[\mathbf{A}]_T := g_T^{\mathbf{A}}$, where exponentiation is carried out component-wise.

B. Assumptions

Definition 3.1. (Discrete Logarithm (DL) Assumption). Let g be a generator of a cyclic group G. DL assumption holds if for all PPT adversary $\mathcal{A}$, the advantage function $\mathsf{Adv}_{\mathcal{A}}^{\mathrm{DL}}(\lambda) := Pr[\mathcal{A}(g, g^a) = a]$ is negligible, where $a \xleftarrow{\$} \mathbb{Z}_p^*$.

Definition 3.2. (Decisional Diffie-Hellman (DDH) Assumption). Let g be a generator of G and $\mathcal{T} = (g, g^a, g^b) \in G^3$, where $a, b \xleftarrow{\$} \mathbb{Z}_p^*$. DDH assumption holds if for all PPT adversary $\mathcal{A}$, the advantage $|Pr[\mathcal{A}(\mathcal{T}, g^{ab}) = 1] - Pr[\mathcal{A}(\mathcal{T}, g^c) = 1]|$ is negligible, where $c \xleftarrow{\$} \mathbb{Z}_p^*$.

Definition 3.3. (Matrix DDH (MDDH$_k$) Assumption) [26]. Let $\ell > k \geq 1$, $d \geq 1$. MDDH$_k$ assumption holds if for all PPT adversary $\mathcal{A}$, the advantage $\mathsf{Adv}_{\mathcal{A}}^{\mathrm{MDDH}_k}(\lambda) := |Pr[\mathcal{A}([\mathbf{M}]_1, [\mathbf{MS}]_1) = 1] - Pr[\mathcal{A}([\mathbf{M}]_1, [\mathbf{U}]_1) = 1]|$ is negligible, where $\mathbf{M} \xleftarrow{\$} \mathbb{Z}_p^{\ell \times k}$, $\mathbf{S} \xleftarrow{\$} \mathbb{Z}_p^{k \times d}$ and $\mathbf{U} \xleftarrow{\$} \mathbb{Z}_p^{\ell \times d}$.

C. Linear Secret Sharing for Monotone Boolean Formulae

The information-theoretic linear secret sharing for monotone Boolean formulae [26], [27] is described below.

$\underline{\mathsf{share}(f, \mu)}$. Input: A formula $f : \{0,1\}^n \to \{0,1\}$ of size m (i.e., the number of edges in f is m), and a secret $\mu \in \mathbb{Z}_p$. 1) For each non-output wire $j = 1, \cdots, m-1$, select $\hat{\mu}_j \xleftarrow{\$} \mathbb{Z}_p$. For the output wire, set $\hat{\mu}_m := \mu$. 2) For each outgoing wire j from input node i, add $\mu_j := \hat{\mu}_j$ to the output set of shares and set $\rho(j) := i$. 3) For each AND gate g with input wires a, b and output wire c, add $\mu_{c_a} := \hat{\mu}_c + \hat{\mu}_a + \hat{\mu}_b \in \mathbb{Z}_p$ to the output set of shares and set $\rho(c) := 0$. 4) For each OR gate g with input wires a, b and output wire c, add $\mu_{c_a} := \hat{\mu}_c + \hat{\mu}_a \in \mathbb{Z}_p$ and $\mu_{c_b} := \hat{\mu}_c + \hat{\mu}_b \in \mathbb{Z}_p$ to the output set of shares and set $\rho(c_a) := 0$ and $\rho(c_b) := 0$. 5) Output $(\{\mu_j\}_{j \in [\hat{m}]}, \rho)$.

$\underline{\mathsf{reconstruct}(f, x, \{\mu_j\}_{\rho(j)=0 \vee x_{\rho(j)}=1})}$. Input: A formula f, $\vec{x} \in \{0,1\}^n$, and $\{\mu_j\}_{\rho(j)=0 \vee x_{\rho(j)}=1}$. From the leaves of the formula to the root, calculate the output wire value $\hat{\mu}_c$ at each node. 1) Given $\hat{\mu}_a, \hat{\mu}_b$ associated with the input wires a and b of an AND gate, compute $\hat{\mu}_c = \mu_c - \hat{\mu}_a - \hat{\mu}_b$. 2) Given $\hat{\mu}_a$ (or $\hat{\mu}_b$) associated with the input wires a (or b) of an OR gate, compute $\hat{\mu}_c = \mu_{c_a} - \hat{\mu}_a$ (or $\hat{\mu}_c = \mu_{c_b} - \hat{\mu}_b$). 3) Output $\mu = \hat{\mu}_m$.

D. NC^1 Circuit and Monotone Formulae

We define NC^1 circuit and monotone Boolean formulae following Kowalczyk's [26] and Katsumata's [27] works. A monotone Boolean formula $f : \{0,1\}^n \to \{0,1\}$ is specified by a directed acyclic graph (DAG) with three kinds of nodes: input gate nodes, gate nodes and a single output node. Input nodes have in-degree 0 and out-degree 1, AND/OR nodes have in-degree (fan-in) 2 and out-degree (fan-out) 1, and the output node has in-degree 1 and out-degree 0. We number the edges (wires) $1, 2, \cdots, m$, and each gate node is defined by a tuple (g, a_g, b_g, c_g), where $g : \{0,1\}^2 \to \{0,1\}$ is either AND or OR, a_g, b_g are the incoming wires, c_g is the outgoing wire and $a_g, b_g < c_g$. The size m of a formula is the number of edges in the underlying DAG and the depth d of a formula is the length of the longest path from the output node. Lemma 2.1 in Katsumata's work [27] states the well-known equivalence between the monotone Boolean formulae and NC^1 circuits.

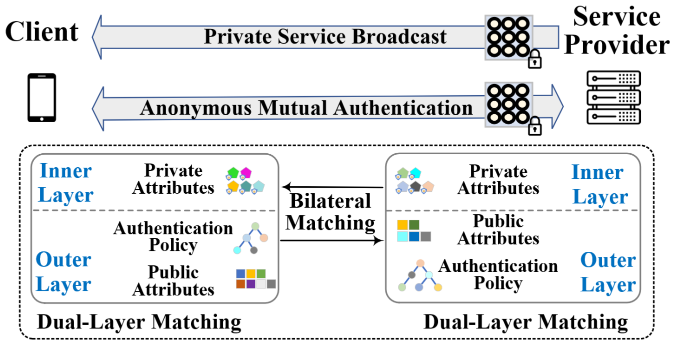


Fig. 1: Overview of PriSrv

IV. PriSrv's Overview

First, we present a technical overview of PriSrv. Next, we provide an example to illustrate how PriSrv is used. Finally, we highlight how PriSrv meets all privacy-enhancement and high usability requirements.

Technical overview. At a high level, PriSrv is a private service discovery protocol that ensures services are only discoverable by an authorized set of clients. PriSrv consists of a private service broadcast phase and an anonymous mutual authentication phase as shown in Fig. 1.

PriSrv's design incorporates a novel crypto-enforced construction that enables both service providers and clients to express flexible access control policies and disclose partial attributes. To meet the privacy enhancement and high usability requirements outlinted in §I-A, we design a dual-layer matching mechanism: an outer layer defines bilateral public authorization policies for filtering unauthorized service providers and clients based on their public attributes; an inner layer performs mutual authentication based on the selectively disclosed private attributes. We design a new cryptographic primitive, named anonymous credential-based matchmaking encryption (ACME), to realize such a dual-layer design in PriSrv.

Anonymous credential (AC) realizes attribute based anonymous authentication with selective attribute disclosure, making it a potential tool for ACME construction. Existing AC schemes suffer from either large credential sizes or cumbersome show and verification mechanisms [28], [29], [30], rendering them unsuitable for privacy-enhanced and highly usable service discovery in wireless networks. We design a new AC scheme, named fast anonymous credential (FAC) as a building block of ACME.

To realize bilateral policy control in ACME, one promising technology is the Matchmaking Encryption (ME) proposed by Ateniese et al. in CRYPTO'19 [15]. In ME, sender (snd) and receiver (rcv) possess a set of attributes $\vec{x}_{\mathsf{snd}}$ and $\vec{x}_{\mathsf{rcv}}$, respectively. The sender is able to specify an authorization policy f_{snd} for the receiver's attributes $\vec{x}_{\mathsf{rcv}}$ to satisfy, and vice versa. ME enables both participants to specify fine-grained policies for encrypted data, which satisfies our need for bilateral policy control. Nonetheless, the ME in [15] has three limitations: (1) the conception of ME to support expressive policies relies on heavy cryptographic tools, including Functional Encryption (FE) and general Zero-Knowledge Proof (ZKP), whose known instantiations are still far from practical; (2) ME does not support selective attribute disclosure; (3) concrete instantiations of ME [15], [31] only support identity-based equality matching. It remains an open problem to develop an efficient ME that supports fine-grained policy based fuzzy matching [15]. We develop ACME to solve this open problem and overcome the above limitations. We further develop PriSrv based on ACME to meet both privacy enhancement and high usability requirements.

Example. We provide a smart office example to exemplify the use of bilateral policy control and selective attribute disclosure in PriSrv. Consider a screen mirroring service provided by a smart TV, which only allows authorized devices to connect to it. On the other hand, a client device should only project its screen to an authorized screen mirroring service device to prevent any leakage of private information. The *service type* in this scenario is the screen mirroring service, and the *service parameters* include resolution, refresh rate, etc. The smart TV is associated with a set of attributes: $\vec{x}_s$=(device type, vendor, model, OS, domain name, device name, location, IP address, security domain), where the first five are public attributes and the rest are private. The mirroring service provider may select a set of public attributes $\vec{x}_s^{(out)}$=(device type, vendor, domain name) to be used in the outer layer, and a set of private attributes $\vec{x}_s^{(in)}$=(IP address) to be used in the inner layer. The client device is associated with another set of attributes: $\vec{x}_c$=(device type, model, OS, department, device name, classified device, IP address, security domain), where the first four are public attributes and the rest are private. The client selects a set of public attributes $\vec{x}_c^{(out)}$=(device type, OS, department) for outer layer matching, and a set of private attributes $\vec{x}_c^{(in)}$=(classified device, security domain) for inner layer authentication.

To realize bilateral control, the service provider (i.e., the smart TV) sets a service policy as

$$\begin{aligned} f_s = \big(&\quad \text{Device Type = "Smart phone} \vee \text{Laptop"} \\ \bigwedge &\quad \text{OS = "Android} \vee \text{iOS} \vee \text{Windows"} \\ \bigwedge &\quad \text{Department = "A} \vee \text{B"}\big). \end{aligned}$$

The client device specifies a connection policy as

$$\begin{aligned} f_c = \big(&\quad \text{Device Type = "TV"} \bigwedge \text{Vendor = "C} \vee \text{D"} \\ \bigwedge &\quad \text{Domain Name = "*.XYZ.COM"}\big). \end{aligned}$$

The screen mirroring service can be discovered by the client if and only if $f_s(\vec{x}_c^{(out)}) = 1 \wedge f_c(\vec{x}_s^{(out)}) = 1$, which indicates that the public attributes of the service provider (and the client, respectively) satisfy the policy of its peer. The private attributes selected by smart TV and client device are used for mutual authentication.

How PriSrv Meets Requirements. PriSrv meets both privacy enhancement and high usability requirements as outlined in §I-A.

- *Private Service Broadcast* & *Mutual Authentication*. The messages broadcasted by service providers are encrypted using ACME such that only intended clients can obtain the decrypted information. Both service providers and clients authenticate each other's private attributes before establishing a secure communication channel.

- *Bilateral Anonymity* & *Bilateral Flexible Policy Control*. Both service providers and clients maintain their anonymity during the discovery process. Bilateral flexible policy control is achieved via ACME, as decryption fails if any protocol participant's policy is not satisfied by its peers' attributes.

- *Selective Attribute Disclosure* & *Multi-Show Unlinkability*. According to the minimum privacy leakage principle, any participant in PriSrv only reveals a subset of its attributes to its peer. Both service provider and client select a subset of their attributes, including public attributes and private attributes to generate their authentication tokens. Multi-show unlinkability of PriSrv is inherited from that of FAC, which ensures the unlinkability of multiple instances of authentication tokens generated by the same entity across multiple protocol sessions (even using the same subset of non-unique attributes).

- *No Pre-registered Pairing* & *No Third-party Dependency for Service Discovery*. PriSrv protocol execution does not require any service provider to know its clients, or any client to subscribe to its service providers in advance. PriSrv operates without relying on any external services during protocol execution.

Threat and Attacker Model. The credential issuer is considered trustworthy to issue and revoke anonymous credentials. Both service providers and clients in the protocol are considered untrustworthy, as they have the potential to launch any passive or active attacks. Specifically, a service provider may attempt to impersonate other providers by broadcasting deceptive messages or to track clients' activities. Likewise, a client may impersonate other clients to obtain unauthorized network access.

Following the Canetti-Krawczyk model for authenticated key-exchange (AKE) in [32], [33] and the service discovery model in [5], the attackers against PriSrv include malicious service providers, clients, and external adversaries. We aim to comprehensively model the attackers' capabilities in the real world to gain full control over public network communication. This control encompasses actions such as revealing certain protocol secrets, intercepting or tampering with channel messages, replaying, delaying, injecting or dropping data packets, and interleaving messages from different sessions, etc. They are capable to launch various types of attacks, including eavesdropping attacks, spoofing attacks, impersonation attacks, man-in-the-middle attacks, etc. The attackers' goals include:

(1) breaking authenticated key-exchange security; and (2) revealing sensitive information pertaining to clients or service providers, enabling attackers to track their activities.

Formal Security Definition and Analysis. The formal security models of private service discovery include service discovery security and bilateral anonymity, which is followed by formal security proofs. The formal security models and proofs are shown in Appendix C.

V. Fast Anonymous Credential

We propose a fast anonymous credential scheme (FAC) as a component of ACME to enable fast anonymous authentication while maintaining a constant and small credential size. FAC supports re-randomization of credentials to support multi-show unlinkability, and selective attribute disclosure. We provide the syntax for anonymous credentials and proceed to construct a concrete FAC scheme for mobile devices.

A. Syntax of Anonymous Credential

Anonymous credential (AC) is formally defined by the following PPT algorithms [35], [36].

• $\mathsf{Setup}(1^\lambda, 1^n) \rightarrow \mathsf{pp}$: On input a security parameter λ and a function parameter 1^n, it outputs public parameter pp, which is an implicit input to all the other algorithms.

• $\mathsf{CredKeyGen}(\mathsf{pp}) \rightarrow (\mathsf{pk}, \mathsf{sk})$: On input pp, this algorithm creates credential issuer's public/secret keys pk/sk, where pk is an implicit input to the algorithms below.

• $\mathsf{UserKeyGen}(\mathsf{pp}) \rightarrow (\mathsf{upk}, \mathsf{usk})$: On input pp, the algorithm generates user's public key upk and secret key usk.

• $\langle \mathsf{Issue.I}(\mathsf{sk}, \mathsf{upk}) \rightleftarrows \mathsf{Issue.U}(uid, \vec{x}, \mathsf{usk}) \rangle \rightarrow \mathsf{cred}$. This is an interactive protocol for AC issuance executed between the issuer and a user over a secure channel. The user executes the protocol by inputting a user's identity uid, an attribute set $\vec{x}$ and a secret key usk. The credential issuer runs the protocol by inputting sk and upk. The issuer hands over a credential cred to user via secure channel.

• $\mathsf{Show}(uid, \{x_i\}_{i\in\mathcal{I}}, \mathsf{cred}, \mathsf{usk}, m) \rightarrow \mathsf{tok}$: On input uid, an attribute subset $\{x_i\}_{i\in\mathcal{I}} \subseteq \vec{x}$ ($\mathcal{I} \subseteq [1, n]$), cred, usk and a message m, it outputs an authentication token tok.

• $\mathsf{Verify}(\mathsf{tok}, m) \rightarrow b \in \{0, 1\}$. On input tok and m, it outputs $b = 1$ if tok is valid; otherwise, it outputs $b = 0$.

Following the security definitions in [35], [36], the *correctness*, *unforgeability*, *anonymity* and *unlinkability* of AC are defined, which are shown in Appendix A.

B. Construction of FAC

Our construction of FAC is given below.

• $\mathsf{Setup}(1^\lambda, 1^n) \rightarrow \mathsf{pp}$: Let λ be the security parameter, and n the attribute number in the system. Run $\mathbb{G} = (p, G_1, G_2, G_T, e) \xleftarrow{\$} \mathsf{GGen}(1^\lambda)$, and output $\mathsf{pp} = (g, h, n)$, where g, h are the generators of G_1, G_2, respectively.

• $\mathsf{CredKeyGen}(\mathsf{pp}) \rightarrow (\mathsf{pk}, \mathsf{sk})$: The issuer samples $\tau, y_i \xleftarrow{\$} \mathbb{Z}_p^*$, computes $W \leftarrow g^\tau$, $X_i \leftarrow h^{y_i}$, $Y_i \leftarrow g^{y_i}$ for $i \in [0, n+1]$, and $Z_{i,j} = g^{y_i \cdot y_j}$ for $0 \leq i \neq j \leq n+1$. Then, it outputs secret key $\mathsf{sk} = (\tau, \{y_i\}_{i\in[0,n+1]})$ and public key $\mathsf{pk} \leftarrow (W, \{X_i, Y_i\}_{i\in[0,n+1]}, \{Z_{i,j}\}_{0\leq i\neq j\leq n+1})$.

• $\mathsf{UserKeyGen}(\mathsf{pp}) \rightarrow (\mathsf{upk}, \mathsf{usk})$: The user with uid samples $\mathsf{usk} \xleftarrow{\$} \mathbb{Z}_p^*$, computes $\mathsf{upk} \leftarrow h^{\mathsf{usk}}$, and creates a signature proof of knowledge π_1 as $\mathsf{SPK}\{\mathsf{usk} : \mathsf{upk} = h^{\mathsf{usk}}\}$. The issuer registers upk if $\mathsf{Verify}_{\mathsf{SPK}}(\mathsf{upk}, \pi_1) = 1$ holds.

• $\langle \mathsf{Issue.I}(\mathsf{sk}, \mathsf{upk}) \rightleftarrows \mathsf{Issue.U}(uid, \vec{x}, \mathsf{usk}) \rangle \rightarrow \mathsf{cred}$. The secure channel between issuer and user can be established by standard protocols, such as TLS.

(1) User sends uid and attributes $\vec{x} = \{x_i\}_{i\in[1,n]}$ to issuer.

(2) The issuer samples $r \xleftarrow{\$} \mathbb{Z}_p^*$ to calculate $\mathsf{cred} \leftarrow (\sigma_1, \sigma_2)$, where

$$\sigma_1 \leftarrow h^r, \sigma_2 \leftarrow \mathsf{upk}^{r\cdot y_0} \cdot h^{r(\tau + \sum_{i=1}^n y_i x_i + y_{n+1}\cdot uid)}.$$

(3) The user accepts the credential cred if the following equation holds

$$e(W \cdot Y_0^{\mathsf{usk}} \cdot Y_{n+1}^{uid} \prod\nolimits_{i=1}^n Y_i^{x_i}, \sigma_1) = e(g, \sigma_2).$$

• $\mathsf{Show}(uid, \{x_i\}_{i\in\mathcal{I}}, \mathsf{cred}, \mathsf{usk}, m) \rightarrow \mathsf{tok}$: The user generates a token on selected attribute subset $\{x_i\}_{i\in\mathcal{I}}$, $\mathcal{I} \subseteq [1, n]$. Select $t_1, t_2 \xleftarrow{\$} \mathbb{Z}_p^*$ to compute

$$T_1 = g^{t_1} \prod_{j\in[1,n]\setminus\mathcal{I}} Y_j^{x_j}, \quad T_2 = (\prod_{i\in\mathcal{I}'} Y_i)^{t_1} \prod_{i\in\mathcal{I}', j\in[1,n]\setminus\mathcal{I}} Z_{i,j}^{x_j},$$

$\bar{\sigma}_1 = \sigma_1^{t_2}$, $\bar{\sigma}_2 = \sigma_2^{t_2}\bar{\sigma}_1^{t_1}$, and create π_2 as

$$\mathsf{SPK}\left\{ (\mathsf{usk}, uid) : \begin{array}{c} \bar{\sigma}_1 = \sigma_1^{t_2}, \bar{\sigma}_2 = \sigma_2^{t_2}\bar{\sigma}_1^{t_1}, \sigma_1 = h^r, \\ \sigma_2 = (h^{\mathsf{usk}})^{r\cdot y_0} h^{r(\tau + \sum\limits_{i=1}^{n} y_i x_i + y_{n+1}\cdot uid)} \end{array} \right\}(m),$$

where $\mathcal{I}' = \mathcal{I} \cup \{0, n+1\}$. The token is

$$\mathsf{tok} \leftarrow (\{x_i\}_{i\in\mathcal{I}}, T_1, T_2, \bar{\sigma}_1, \bar{\sigma}_2, \pi_2).$$

• $\mathsf{Verify}(\mathsf{tok}, m) \rightarrow b \in \{0, 1\}$. The algorithm outputs $b = 1$ if $\mathsf{Verify}_{\mathsf{SPK}}(\mathsf{tok}, m) = 1$. Otherwise, it returns $b = 0$.

Instantiation of SPK. Following the standard Fiat-Shamir paradigm, SPKs in FAC are instantiated as follows.

The SPK π_1:

Prove: Prover selects $\widetilde{\mathsf{usk}} \xleftarrow{\$} \mathbb{Z}_p^*$ and computes $\gamma \leftarrow h^{\widetilde{\mathsf{usk}}}$, $c \leftarrow H(\mathsf{upk}, \gamma)$, $\overline{\mathsf{usk}} = \widetilde{\mathsf{usk}} - c \cdot \mathsf{usk} \mod p$. Return $\pi_1 \leftarrow (c, \gamma, \overline{\mathsf{usk}})$.

Verify: Given upk and SPK π_1, the verifier checks $c \overset{?}{=} H(\mathsf{upk}, \gamma)$, $\gamma \overset{?}{=} h^{\overline{\mathsf{usk}}}\mathsf{upk}^c$. It outputs 1 if these equations hold, and 0 otherwise.

The SPK π_2:

Prove: Prover selects $\widetilde{uid}, \widetilde{\mathsf{usk}} \xleftarrow{\$} \mathbb{Z}_p^*$ and computes $\Lambda \leftarrow e(Y_0^{\widetilde{\mathsf{usk}}} Y_{n+1}^{\widetilde{uid}}, \bar{\sigma}_1)$, $c \leftarrow H(m, \{x_i\}_{i\in\mathcal{I}}, \Lambda, T_1, T_2, \bar{\sigma}_1, \bar{\sigma}_2)$, $\overline{uid} \leftarrow \widetilde{uid} - c \cdot uid \mod p$, $\overline{\mathsf{usk}} \leftarrow \widetilde{\mathsf{usk}} - c \cdot \mathsf{usk} \mod p$. Set $\pi_2 \leftarrow (c, \overline{uid}, \overline{\mathsf{usk}}, \Lambda)$, and return $\mathsf{tok} \leftarrow (\{x_i\}_{i\in\mathcal{I}}, T_1, T_2, \bar{\sigma}_1, \bar{\sigma}_2, \pi_2)$.

Verify: Given tok, the verifier checks $c \overset{?}{=} H(m, \{x_i\}_{i\in\mathcal{I}}, \Lambda, T_1, T_2, \bar{\sigma}_1, \bar{\sigma}_2)$, $e(Y_0^{\overline{\mathsf{usk}}} Y_{n+1}^{\overline{uid}}, \bar{\sigma}_1)^{-1} \cdot \Lambda \overset{?}{=} [e(g, \bar{\sigma}_2) \cdot \Gamma]^c$, $e(T_1, \prod_{i\in\mathcal{I}'} X_i) \overset{?}{=} e(T_2, h)$, where $\Gamma = e(W \cdot T_1 \cdot \prod_{i\in\mathcal{I}} Y_i^{x_i}, \bar{\sigma}_1)^{-1}$. It outputs 1 if these equations hold, and 0 otherwise.

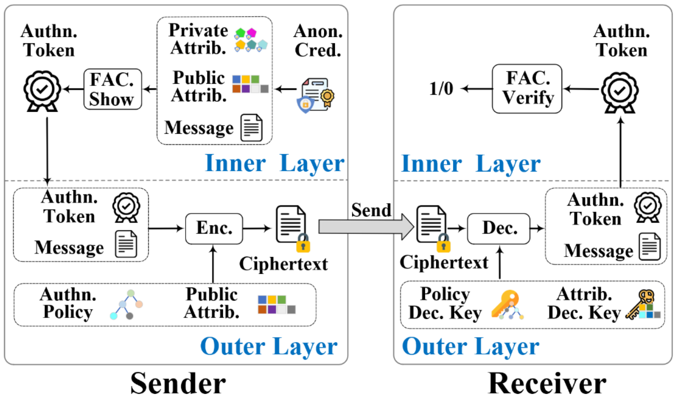


Fig. 2: Architecture of ACME

Our fast anonymous credential (FAC) scheme has the following advantages: 1) FAC offers a non-interactive $\mathsf{Show} \leftrightarrows \mathsf{Verify}$ process, ensuring fast anonymous authentication. 2) FAC generates anonymous credentials of a constant and small size. 3) An authentication token generated in FAC consists of only two group elements. The construction is based on the unlinkable redactable signature (URS) scheme [36], which is one of the initial frameworks for generating constant-size redactable signatures on attributes $\vec{x} = (x_1, \cdots, x_n)$. FAC generates an anonymous credential cred based on the URS scheme [36]. When a request is made to verify the authenticity of a subset of attributes $\{x_i\}_{i\in\mathcal{I}} \subseteq \vec{x}$, the Show algorithm in FAC performs the following steps: it derives an authentication token tok from the anonymous credential cred, and then produces a signature proof of knowledge (SPK) for the authentication token. The Verify algorithm in FAC is responsible for checking the validity of tok. The correctness proof of FAC is shown in Appendix A.

Theorem 5.1. The FAC scheme is secure (i.e., achieves unforgeability, anonymity and unlinkability) under the DL and DDH assumptions.

The proof of Theorem 5.1 is shown in Appendix A.

VI. Anonymous credential-based matchmaking encryption (ACME)

We introduce a new cryptographic primitive named ACME to support several core features in PriSrv protocol, including bilateral policy control, anonymous authentication and selective attribute disclosure. ACME is a variant of ME where the sender and the receiver can use anonymous credentials to prove their attributes without revealing their identities. This is useful because it allows for stronger privacy guarantees and flexible policy enforcement in scenarios such as secure online dating, e-voting, and anonymous whistleblowing, where the parties do not trust each other or third parties. ACME is of independent interests for advancing research on Matchmaking Encryption.

A. Design Intuition

Matchmaking Encryption (ME) is a natural starting point to construct ACME. However, the conception of ME [15] that can simultaneously support expressive policy (e.g., monotone Boolean formulae) and policy hiding is of theoretical interest only since no concrete instantiation has been proposed. Although identity-based ME schemes supporting equality policies were introduced in [15], [31], they do not fit for highly-usable service discovery since in general participants of service discovery are unaware of their peers' identities and thus cannot define identity-based equality policies. Meanwhile, we notice that the original ME schemes [15], [31] support hidden policies, but they are not ideal for service discovery because such schemes require clients to blindly decrypt every service advertisement, bringing high costs when multiple services are in presence.

To balance fast service discovery and privacy protection, ACME adopts a dual-layer matching design for a sender (snd) to encrypt any message M and send the ciphertext to a receiver (rcv) with bilateral policy control. Sender snd receives an anonymous credential $\mathsf{cred}_{\mathsf{snd}}$ from a credential issuer for all its attributes $\vec{x}_{\mathsf{snd}}$. As shown in Fig. 2, ACME consists of an inner layer and an outer layer. In the inner layer, sender snd generates an authentication token using $\mathcal{FAC}.\mathsf{Show}$ from a message M and selected attributes (including public and private attributes) based on the received credential $\mathsf{cred}_{\mathsf{snd}}$. In the outer layer, sender snd encrypts the authentication token and the message M using an authentication policy f_{snd} (specified by snd for rcv) and the sender's selected public attributes. Then, sender snd transmits the ciphertext to receiver rcv.

On the receiver side, the ciphertext is decrypted in the outer layer using receiver's policy decryption key and attribute decryption key to recover the authentication token and message M. In the inner layer, the authentication token is verified using $\mathcal{FAC}.\mathsf{Verify}$ to authenticate the sender's selected private attributes, public attributes and the message M.

Impersonation Resistance. ACME is the core component of PriSrv to prevent impersonation attacks. As shown in Fig. 2, both the public and private attributes are used as inputs for authentication token generation in the inner layer. This design has been purposefully engineered to provide robust protection against impersonation attacks. Although the public attributes used in the service provider's outer layer are public, a malicous service provider (without all the authorized public attributes) is not able to impersonate any legal provider since the forged authentication token cannot pass the verification by the receiver (using $\mathcal{FAC}.\mathsf{Verify}$). On the other hand, if the public attributes used in the outer layer are not unique, PriSrv relies on the inner layer to authenticate both public and private attributes, which rules out any impersonation attack. Meanwhile, an attacker impersonating a legitimate receiver cannot be successful in decryption without a valid secret key.

B. Syntax of ACME

Anonymous credential-based matchmaking encryption (ACME) is formally defined below, and the correctness of ACME is defined in Appendix B.

- $\mathsf{Setup}(1^\lambda, 1^n)$: On input a security parameter 1^λ and a function parameter 1^n, this algorithm outputs the master public/secret keys $\mathsf{mpk}/\mathsf{msk}$. Note that mpk is implicit input in all the following algorithms.
- $\mathsf{CredKeyGen}(\mathsf{mpk}) \to (\mathsf{pk}, \mathsf{sk})$: On input mpk, this algorithm creates credential issuer's public key pk and secret key sk. pk is an implicit input to the following algorithms.
- $\mathsf{UserKeyGen}(\mathsf{mpk}) \to (\mathsf{upk}, \mathsf{usk})$: On input mpk, the algorithm generates user's public key upk and secret key usk.
- $\langle \mathsf{Issue.I}(\mathsf{sk}, \mathsf{upk}) \rightleftarrows \mathsf{Issue.U}(uid, \vec{x}, \mathsf{usk})\rangle \to \mathsf{cred}$. The issuer inputs sk, upk and the user inputs uid, usk, full attributes $\vec{x}$. The issuer interacts with user to generate a credential cred for the user.
- $\mathsf{DKGen}(\mathsf{msk}, \vec{x}_{\mathsf{rcv}})$: On input msk and attributes $\vec{x}_{\mathsf{rcv}}$, this algorithm outputs an attribute decryption key $\mathsf{DK}_{\vec{x}_{\mathsf{rcv}}}$.
- $\mathsf{PolGen}(\mathsf{msk}, f_{\mathsf{rcv}})$: On input msk and policy f_{rcv}, this algorithm outputs a policy decryption key $\mathsf{DK}_{f_{\mathsf{rcv}}}$.
- $\mathsf{Enc}(\mathsf{cred}_{\mathsf{snd}}, \vec{x}_{\mathsf{snd}}, f_{\mathsf{snd}}, M)$: On input $\mathsf{cred}_{\mathsf{snd}}$, full attributes $\vec{x}_{\mathsf{snd}}$, policy f_{snd} and message M as input, the sender selects a set of private attributes $\vec{x}^{(in)}_{\mathsf{snd}}$ for an inner layer and a set of public attributes $\vec{x}^{(out)}_{\mathsf{snd}}$ for an outer layer from $\vec{x}_{\mathsf{snd}}$. It firstly generates a token $\mathsf{tok}_{\mathsf{snd}}$ for $\vec{x}^{(in)}_{\mathsf{snd}}$, $\vec{x}^{(out)}_{\mathsf{snd}}$ and message M. Then, it encrypts $(M, \mathsf{tok}_{\mathsf{snd}})$ using the public attributes $\vec{x}^{(out)}_{\mathsf{snd}}$ and policy f_{snd}, and outputs a ciphertext $\mathsf{CT}_{\vec{x}_{\mathsf{snd}}, f_{\mathsf{snd}}}$.
- $\mathsf{Dec}(\mathsf{DK}_{\vec{x}_{\mathsf{rcv}}}, \mathsf{DK}_{f_{\mathsf{rcv}}}, \mathsf{CT}_{\vec{x}_{\mathsf{snd}}, f_{\mathsf{snd}}})$: On input $\mathsf{DK}_{\vec{x}_{\mathsf{rcv}}}$, $\mathsf{DK}_{f_{\mathsf{rcv}}}$ and $\mathsf{CT}_{\vec{x}_{\mathsf{snd}}, f_{\mathsf{snd}}}$, the receiver recovers $(M, \mathsf{tok}_{\mathsf{snd}})$ iff $f_{\mathsf{snd}}(\vec{x}^{(out)}_{\mathsf{rcv}}) = 1$ and $f_{\mathsf{rcv}}(\vec{x}^{(out)}_{\mathsf{snd}}) = 1$; otherwise, it outputs $\perp$. If the above step succeeds, the receiver verifies $\mathsf{tok}_{\mathsf{snd}}$ for $\vec{x}^{(in)}_{\mathsf{snd}} \cup \vec{x}^{(out)}_{\mathsf{snd}}$ and M. It outputs the message M if the token is valid; otherwise, it outputs $\perp$.

Remark. In encryption algorithm, the authentication token $\mathsf{tok}_{\mathsf{snd}}$ is generated for selected public and private attributes rather than just private attributes. The purpose is to authenticate sender's selective attributes in both layers to prevent spoofing attacks. The token also authenticates M to prevent message forgery.

Definition 6.1. An ACME scheme is secure if it satisfies *privacy*, *authenticity*, *anonymity* and *unlinkability*.

The formal definitions of these security properties are provided in Appendix B.

C. Construction of ACME

FAC in §V is leveraged in the inner layer of ACME for authentication. For outer-layer encryption and bilateral policy control, we resort to attribute-based encryption (ABE) that supports expressive access policies. However, ABE only supports unilateral policy control. To enable bilateral control, a potential solution is to integrate key policy (KP-)ABE and ciphertext policy (CP-)ABE so that the secret key of CP-ABE (resp. KP-ABE) functions as attribute decryption key (resp. policy decryption key) produced by the DKGen (resp. PolGen) algorithm. Although the idea seems straightforward, there are a few subtleties to be addressed. Firstly, compact ABE schemes are preferred for compact ciphertext size and key size. The compact KP-ABE and CP-ABE schemes proposed by Kowalczyk et al. [26] in Eurocrypt'19 are natural candidates because they are in the dual form with common parameters and support Boolean formulae (equivalent to NC^1 circuits[1]). Nonetheless, the decryption process of the dual ABE schemes in [26] involves a large number of time-consuming pairing operations (depending on the complexity of NC^1). If we construct ACME based on the dual schemes given in [26], such ACME would incur high computational costs for wireless devices. Meanwhile, we notice that for KP-ABE, Katsumata et al. proposed an improved scheme in [27] with faster decryption, which requires only a constant number of pairing operations. We apply the technique in [27] to improve CP-ABE scheme in [26] to achieve fast decryption with a constant number of pairing operations. By integrating the improved CP-ABE with Katsumata's KP-ABE [27], which are also in a dual form, we can achieve both fine-grained bilateral policy control and fast decryption.

Concrete Construction. Our ACME scheme for general policies is built from the above fast anonymous credential scheme $\mathcal{FAC}$, a symmetric encryption scheme $\mathcal{SE} = (\mathsf{SGen}, \mathsf{SEnc}, \mathsf{SDec})$ with key space $\mathcal{K}$, and a hash function $H : \{0,1\}^* \to \mathcal{K}$.

$\mathsf{Setup}(1^\lambda, 1^n)$: Run $\mathbb{G} = (p, G_1, G_2, G_T, e) \xleftarrow{\$} \mathsf{GGen}(1^\lambda)$. Let g, h be the generators of G_1, G_2, respectively. Run $\mathcal{FAC}.\mathsf{Setup}(1^\lambda, 1^n)$ to get pp. Sample $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_p^{k\times 2k}$, $\mathbf{B} \xleftarrow{\$} \mathbb{Z}_p^{k\times k}$, $\mathbf{U}_0, \mathbf{W}_i \xleftarrow{\$} \mathbb{Z}_p^{2k\times k}$ for $i \in [n]$, $\mathbf{v} \xleftarrow{\$} \mathbb{Z}_p^{2k}$, output

$$\mathsf{msk} = (\mathbf{v}, \mathbf{B}, \mathbf{U}_0, \mathbf{W}_1, \cdots, \mathbf{W}_n),$$
$$\mathsf{mpk} = (\mathsf{pp}, [\mathbf{A}]_1, [\mathbf{A}\mathbf{U}_0]_1, [\mathbf{A}\mathbf{W}_1]_1, \cdots, [\mathbf{A}\mathbf{W}_n]_1, e([\mathbf{A}]_1, [\mathbf{v}]_2)).$$

$\mathsf{CredKeyGen}(\mathsf{mpk}) \to (\mathsf{pk}, \mathsf{sk})$: This algorithm executes $\mathcal{FAC}.\mathsf{CredKeyGen}$ to generate issuer's pk and sk.

$\mathsf{UserKeyGen}(\mathsf{mpk}) \to (\mathsf{upk}, \mathsf{usk})$: This algorithm executes $\mathcal{FAC}.\mathsf{UserKeyGen}$ to generate user's upk and usk.

$\langle \mathsf{Issue.I}(\mathsf{sk}, \mathsf{upk}) \rightleftarrows \mathsf{Issue.U}(uid, \vec{x}, \mathsf{usk})\rangle \to \mathsf{cred}$. This algorithm executes $\mathcal{FAC}.\mathsf{Issue}$ to create user's credential cred.

$\mathsf{DKGen}(\mathsf{msk}, \vec{x}_{\mathsf{rcv}})$: To generate an attribute decryption key for receiver's attributes $\vec{x}_{\mathsf{rcv}}$, it samples $\mathbf{r} \xleftarrow{\$} \mathbb{Z}_p^k$ and outputs $\mathsf{DK}_{\vec{x}_{\mathsf{rcv}}} = (\mathsf{dk}_1, \mathsf{dk}_2, \mathsf{dk}_3)$:

$$\mathsf{dk}_1 = [\mathbf{v}+\mathbf{U}_0\mathbf{B}\mathbf{r}]_2, \mathsf{dk}_2 = [\mathbf{B}\mathbf{r}]_2, \mathsf{dk}_3 = [\textstyle\sum_{i:x^{(out)}_{r,i}=1} \mathbf{W}_i\mathbf{B}\mathbf{r}]_2.$$

$\mathsf{PolGen}(\mathsf{msk}, f_{\mathsf{rcv}})$: To generate a policy decryption key for receiver's policy f_{rcv}, this algorithm samples $(\{\mathbf{v}_j\}_{j\in[\hat{m}_r]}, \rho_r) \xleftarrow{\$} \mathsf{share}(f_{\mathsf{rcv}}, \mathbf{v})$, $\mathbf{r}_j \xleftarrow{\$} \mathbb{Z}_p^k$ and outputs $\mathsf{DK}_{f_{\mathsf{rcv}}} = (\{\mathsf{dk}_j, \mathsf{dk}_{\rho_r(j),j}, \{\mathsf{dk}_{i,j}\}_{i\in[n]\setminus\{\rho_r(j)\}}\}_{j\in[\hat{m}_r]})$:

$$\mathsf{dk}_j = [\mathbf{r}_j]_2, \mathsf{dk}_{\rho_r(j),j} = [\mathbf{v}_j + \mathbf{W}_{\rho_r(j)}\mathbf{r}_j]_2, \mathsf{dk}_{i,j} = [\mathbf{W}_i\mathbf{r}_j]_2,$$

where $\mathbf{W}_0 = \mathbf{0}$, $\hat{m}_r$ is the number of shares for receiver's policy, and ρ_r is a mapping from the indices of the shares to

[1]In computational complexity theory, NC^i is the class of decision problems decidable by uniform boolean circuits with a polynomial number of gates of at most two inputs and depth $O(\log^i n)$, or the class of decision problems solvable in time $O(\log^i n)$ on a parallel computer with a polynomial number of processors, where NC is short for "Nick Pippenger's Class".

the indices of receiver's public attributes[2]. For $\rho_r(j)=0$, we have $[n]\backslash\{\rho_r(j)\}=[n]$.

$\mathsf{Enc}(\mathsf{cred}_{\mathsf{snd}}, \vec{x}_{\mathsf{snd}}, f_{\mathsf{snd}}, M)$: The sender selects the private attributes $\vec{x}^{(in)}_{\mathsf{snd}}$ for inner layer and public attributes $\vec{x}^{(out)}_{\mathsf{snd}}$ for outer layer from $\vec{x}_{\mathsf{snd}}$. Then, it runs $\mathcal{FAC}.\mathsf{Show}$ to obtain $\mathsf{tok}_{\mathsf{snd}}$ for $\vec{x}^{(in)}_{\mathsf{snd}}$, $\vec{x}^{(out)}_{\mathsf{snd}}$ and $M\in\{0,1\}^*$. Next, it encrypts $(M,\mathsf{tok}_{\mathsf{snd}})$ using the public attributes $\vec{x}^{(out)}_{\mathsf{snd}}$ and policy f_{snd} as follows.

The sender samples $\widetilde{\mathbf{s}}, \mathbf{s}, \mathbf{s}_j \xleftarrow{\$} \mathbb{Z}_p^k$, $(\{\mathbf{u}_j^\top\}_{j\in[\hat{m}_s]}, \rho_s) \xleftarrow{\$} \mathsf{share}(f_{\mathsf{snd}}, \mathbf{s}^\top\mathbf{A}\mathbf{U}_0)$, $K\in G_T$, and compute $\mathsf{CT}_{\vec{x}_{\mathsf{snd}}, f_{\mathsf{snd}}} = (\mathsf{ct}_M, \mathsf{ct}_0, \mathsf{ct}'_1, \mathsf{ct}'_2, \mathsf{ct}_1, \{\widetilde{\mathsf{ct}}_j, \mathsf{ct}_{\rho_s(j),j}, \{\mathsf{ct}_{i,j}\}_{i\in[n]\backslash\{\rho_s(j)\}}\}_{j\in[\hat{m}_s]})$:

$$\begin{aligned}
&\mathsf{ct}_M = \mathcal{SE}.\mathsf{SEnc}(H(K), (M, \mathsf{tok}_{\mathsf{snd}})),\\
&\mathsf{ct}_0 = e([\widetilde{\mathbf{s}}^\top\mathbf{A} + \mathbf{s}^\top\mathbf{A}]_1, [\mathbf{v}]_2)\cdot K,\\
&\mathsf{ct}'_1 = [\widetilde{\mathbf{s}}^\top\mathbf{A}]_1, \quad \mathsf{ct}'_2 = \left[\widetilde{\mathbf{s}}^\top \textstyle\sum_{i:x^{(out)}_{s,i}=1}\mathbf{A}\mathbf{W}_i\right]_1,\\
&\mathsf{ct}_1 = [\mathbf{s}^\top\mathbf{A}]_1, \quad \widetilde{\mathsf{ct}}_j = [\mathbf{s}_j^\top\mathbf{A}]_1,\\
&\mathsf{ct}_{\rho_s(j),j} = \left[\mathbf{u}_j^\top + \mathbf{s}_j^\top\mathbf{A}\mathbf{W}_{\rho_s(j)}\right]_1, \quad \mathsf{ct}_{i,j} = \left[\mathbf{s}_j^\top\mathbf{A}\mathbf{W}_i\right]_1,
\end{aligned}$$

where $\mathbf{W}_0=\mathbf{0}$, $x^{(out)}_{s,i}$ is sender's i-th public attribute for outer layer, $\hat{m}_s$ is the number of shares for sender's policy, and ρ_s is a mapping from the indices of the shares to the indices of sender's public attributes.

$\mathsf{Dec}(\mathsf{DK}_{\vec{x}_{\mathsf{rcv}}}, \mathsf{DK}_{f_{\mathsf{rcv}}}, \mathsf{CT}_{\vec{x}_{\mathsf{snd}}, f_{\mathsf{snd}}})$: The receiver recovers $(M, \mathsf{tok}_{\mathsf{snd}})$ using $(\mathsf{DK}_{\vec{x}_{\mathsf{rcv}}}, \mathsf{DK}_{f_{\mathsf{rcv}}})$ as follows. It compute ω_j, μ_j such that $\mathbf{v} = \sum_{j\in\mathcal{S}_r}\omega_j\mathbf{v}_j$, $\mathbf{s}^\top\mathbf{A}\mathbf{U}_0 = \sum_{j\in\mathcal{S}_s}\mu_j\mathbf{u}_j^\top$, and calculates

$$\begin{aligned}
K &= \mathsf{ct}_0\cdot\frac{e\big(\mathsf{ct}'_2, \prod_{j\in\mathcal{S}_r}\mathsf{dk}_j^{\omega_j}\big)\cdot}{e\big(\mathsf{ct}'_1, \prod_{j\in\mathcal{S}_r}\big(\prod_{i:x^{(out)}_{s,i}=1}\mathsf{dk}_{i,j}\big)^{\omega_j}\big)}\\
&\quad\cdot\frac{e(\prod_{j\in\mathcal{S}_s}(\prod_{i:x^{(out)}_{r,i}=1}\mathsf{ct}_{i,j})^{\mu_j}, \mathsf{dk}_2)}{e(\mathsf{ct}_1,\mathsf{dk}_1)e(\prod_{j\in\mathcal{S}_s}\widetilde{\mathsf{ct}}_j^{\mu_j}, \mathsf{dk}_3)},
\end{aligned}$$

where $\mathcal{S}_r = \{j:\rho_r(j)=0\vee x^{(out)}_{s,\rho_r(j)}=1\}$, $\mathcal{S}_s=\{j:\rho_s(j)=0\vee x^{(out)}_{r,\rho_s(j)}=1\}$ and $x^{(out)}_{r,i}$ is receiver's i-th public attribute for outer layer.

If $f_{\mathsf{snd}}(\vec{x}^{(out)}_{\mathsf{rcv}})=0\vee f_{\mathsf{rcv}}(\vec{x}^{(out)}_{\mathsf{snd}})=0$, it outputs $\perp$; otherwise, it recovers $(M,\mathsf{tok}_{\mathsf{snd}})\leftarrow\mathcal{SE}.\mathsf{SDec}(H(K),\mathsf{ct}_M)$. Then, the receiver runs $\mathcal{FAC}.\mathsf{Verify}(\mathsf{tok}_{\mathsf{snd}}, M)$ to verify $\mathsf{tok}_{\mathsf{snd}}$ for $\vec{x}^{(in)}_{\mathsf{snd}}\cup\vec{x}^{(out)}_{\mathsf{snd}}$ and M. It outputs the message M if the token is valid; otherwise, it outputs $\perp$.

The correctness of ACME scheme is analyzed below.

Denote $\mathcal{S}_s=\{j:\rho_s(j)=0\vee x^{(out)}_{r,\rho_s(j)}=1\}$ and $\mathcal{S}_r=\{j:\rho_r(j)=0\vee x^{(out)}_{s,\rho_r(j)}=1\}$. The correctness of ACME relies on the fact that $\prod_{j\in\mathcal{S}_r}\mathsf{dk}_j^{\omega_j}=\prod_{j\in\mathcal{S}_r}[\mathbf{r}_j]_2^{\omega_j}=[\hat{\mathbf{r}}]_2$,

$$\prod_{j\in\mathcal{S}_r}\Big(\prod_{i:x^{(out)}_{s,i}=1}\mathsf{dk}_{i,j}\Big)^{\omega_j} = [\mathbf{v}+\sum_{i:x^{(out)}_{s,i}=1}\mathbf{W}_i\hat{\mathbf{r}}]_2,$$

where $\hat{\mathbf{r}}=\sum_{j\in\mathcal{S}_r}\omega_j\mathbf{r}_j$. Also we have,

$$\begin{aligned}
&e(\mathsf{ct}_1,\mathsf{dk}_1) = [\mathbf{s}^\top\mathbf{A}\mathbf{v}+\mathbf{s}^\top\mathbf{A}\mathbf{U}_0\mathbf{B}\mathbf{r}]_T,\\
&\textstyle\prod_{j\in\mathcal{S}_s}\widetilde{\mathsf{ct}}_j^{\mu_j} = \prod_{j\in\mathcal{S}_s}[\mathbf{s}_j^\top\mathbf{A}]_1^{\mu_j} = [\hat{\mathbf{s}}^\top\mathbf{A}]_1,
\end{aligned}$$

[2]Please refer to the details of linear secret sharing for NC^1 in §5.1 of [26].

$$\prod_{j\in\mathcal{S}_s}\Big(\prod_{i:x^{(out)}_{r,i}=1}\mathsf{ct}_{i,j}\Big)^{\mu_j} = [\mathbf{s}^\top\mathbf{A}\mathbf{U}_0+\hat{\mathbf{s}}^\top\sum_{i:x^{(out)}_{r,i}=1}\mathbf{A}\mathbf{W}_i]_1,$$

where $\hat{\mathbf{s}}^\top=\sum_{j\in\mathcal{S}_s}\mu_j\mathbf{s}_j^\top$.

Therefore, for all $f_{\mathsf{rcv}}, \vec{x}_{\mathsf{snd}}$ such that $f_{\mathsf{rcv}}(\vec{x}^{(out)}_{\mathsf{snd}})=1$, we have:

$$\begin{aligned}
&\frac{e\big(\mathsf{ct}'_2, \prod_{j\in\mathcal{S}_r}\mathsf{dk}_j^{\omega_j}\big)\cdot}{e\big(\mathsf{ct}'_1, \prod_{j\in\mathcal{S}_r}\big(\prod_{i:x^{(out)}_{s,i}=1}\mathsf{dk}_{i,j}\big)^{\omega_j}\big)}\\
=\;&\frac{e\big([\widetilde{\mathbf{s}}^\top\sum_{i:x^{(out)}_{s,i}=1}\mathbf{A}\mathbf{W}_i]_1, [\hat{\mathbf{r}}]_2\big)}{e\big([\widetilde{\mathbf{s}}^\top\mathbf{A}]_1, [\mathbf{v}+\sum_{i:x^{(out)}_{s,i}=1}\mathbf{W}_i\hat{\mathbf{r}}]_2\big)}\\
=\;&\frac{[\widetilde{\mathbf{s}}^\top\mathbf{A}\hat{\mathbf{r}}\sum_{i:x^{(out)}_{s,i}=1}\mathbf{W}_i]_T}{[\widetilde{\mathbf{s}}^\top\mathbf{A}\mathbf{v}+\widetilde{\mathbf{s}}^\top\mathbf{A}\hat{\mathbf{r}}\sum_{i:x^{(out)}_{s,i}=1}\mathbf{W}_i]_T} = ([\widetilde{\mathbf{s}}^\top\mathbf{A}\mathbf{v}]_T)^{-1}.
\end{aligned}$$

where $\hat{\mathbf{r}}=\sum_{j\in\mathcal{S}_r}\omega_j\mathbf{r}_j$.

For all $f_{\mathsf{snd}}, \vec{x}_{\mathsf{rcv}}$ such that $f_{\mathsf{snd}}(\vec{x}^{(out)}_{\mathsf{rcv}})=1$, we have:

$$\begin{aligned}
&\frac{e(\prod_{j\in\mathcal{S}_s}(\prod_{i:x^{(out)}_{r,i}=1}\mathsf{ct}_{i,j})^{\mu_j}, \mathsf{dk}_2)}{e(\mathsf{ct}_1,\mathsf{dk}_1)\cdot e(\prod_{j\in\mathcal{S}_s}\widetilde{\mathsf{ct}}_j^{\mu_j}, \mathsf{dk}_3)}\\
=\;&\frac{e([\mathbf{s}^\top\mathbf{A}\mathbf{U}_0+\hat{\mathbf{s}}^\top\sum_{i:x^{(out)}_{r,i}=1}\mathbf{A}\mathbf{W}_i]_1, [\mathbf{B}\mathbf{r}]_2)}{[\mathbf{s}^\top\mathbf{A}\mathbf{v}+\mathbf{s}^\top\mathbf{A}\mathbf{U}_0\mathbf{B}\mathbf{r}]_T\cdot e([\hat{\mathbf{s}}^\top\mathbf{A}]_1, [\sum_{i:x^{(out)}_{r,i}=1}\mathbf{W}_i\mathbf{B}\mathbf{r}]_2)}\\
=\;&\frac{[\mathbf{s}^\top\mathbf{A}\mathbf{U}_0\mathbf{B}\mathbf{r}]_T}{[\mathbf{s}^\top\mathbf{A}\mathbf{v}+\mathbf{s}^\top\mathbf{A}\mathbf{U}_0\mathbf{B}\mathbf{r}]_T} = ([\mathbf{s}^\top\mathbf{A}\mathbf{v}]_T)^{-1}.
\end{aligned}$$

Theorem 6.2. The ACME scheme achieves privacy, authenticity, anonymity and unlinkability if the $MDDH_k$ assumption holds and the underlying FAC is secure.

The proof of Theorem 6.2 is shown in Appendix B.

D. Comparison of ME Schemes

Mathmaking encryption (ME) protects data confidentiality with bilateral control for both senders and receivers in communications. The existing instantiations of ME include an identity-based scheme (IBME) [15] proposed in CRYPTO'19 and a security enhanced version [31] in Asiacrypt'22, but they do not support fine-grained access control. Table II compares our ACME with IBME [15], [31]. Since IBME simply sets $\vec{x}_{\mathsf{snd}}=\mathsf{snd}$, $f_{\mathsf{snd}}=\mathsf{rcv}$ and $\vec{x}_{\mathsf{rcv}}=\mathsf{rcv}$, $f_{\mathsf{rcv}}=\mathsf{snd}$, it requires pre-registration pairing between service providers and clients. On the contrary, ACME relies on bilateral policy matching for service discovery and thus it does not need pre-registration pairing. Furthermore, ACME supports expressive policy (i.e., Boolean formulae equivalent to NC^1 circuit), while IBME is constrained to equality policy. On the other hand, the expressive policy in ACME is public to enable fast service discovery, while the equality policy is hidden from the public in IBME.

Service Broadcast Phase
Service Provider S's Broadcast: $bid, \mathsf{CT}_B \leftarrow \mathcal{ACME}.\mathsf{Enc}(\mathsf{cred}_s, \vec{x}_s, f_s, MSG_B)$
where $MSG_B = \{bid \| Z \| Service_{Type} \| Service_{Par} \| K_c\}$, $z \xleftarrow{\$} \mathbb{Z}_p^*$, $Z \leftarrow h^z \in G_2$, $K_c \leftarrow \mathcal{MAC}.\mathsf{KeyGen}(1^\lambda)$

Anonymous Mutual Authentication Phase		
Client (C)		**Service Provider** (S)
$(\mathsf{cred}_c, \mathsf{DK}_{\vec{x}_c}, \mathsf{DK}_{f_c})$		$(\mathsf{cred}_s, \mathsf{DK}_{\vec{x}_s}, \mathsf{DK}_{f_s})$
$MSG_B \leftarrow \mathcal{ACME}.\mathsf{Dec}(\mathsf{DK}_{\vec{x}_c}, \mathsf{DK}_{f_c}, \mathsf{CT}_B)$		
$x_1, x_2 \xleftarrow{\$} \mathbb{Z}_p^*$, $X_1 \leftarrow g^{x_1} \in G_1$, $X_2 \leftarrow h^{x_2} \in G_2$		
$\sigma_c \leftarrow \mathcal{MAC}.\mathsf{MAC}(K_c, M_c)$		
where $M_c = (\text{"}C \rightarrow S\text{"}, bid, sid, X_1, X_2, Z)$		
$K_s \leftarrow \mathcal{MAC}.\mathsf{KeyGen}(1^\lambda)$	$\xrightarrow{bid, sid, \sigma_c, \mathsf{CT}_c}$	$MSG_c \leftarrow \mathcal{ACME}.\mathsf{Dec}(\mathsf{DK}_{\vec{x}_s}, \mathsf{DK}_{f_s}, \mathsf{CT}_c)$
$\mathsf{CT}_c \leftarrow \mathcal{ACME}.\mathsf{Enc}(\mathsf{cred}_c, \vec{x}_c, f_c, MSG_c)$		$b_c \leftarrow \mathcal{MAC}.\mathsf{Verify}(K_c, M_c, \sigma_c)$
where $MSG_c = (K_s, M_c)$		If $b_c = 0$, abort; otherwise,
		$y \xleftarrow{\$} \mathbb{Z}_p^*$, $Y \leftarrow g^y \in G_1$
$b_s \leftarrow \mathcal{MAC}.\mathsf{Verify}(K_s, M_s, \sigma_s)$		$\sigma_s \leftarrow \mathcal{MAC}.\mathsf{MAC}(K_s, M_s)$
If $b_s = 0$, abort; otherwise,	$\xleftarrow{M_s, \sigma_s}$	where $M_s = (\text{"}S \rightarrow C\text{"}, bid, sid, X_1, X_2, Y, Z)$
$SSK_{c,s} \leftarrow H(Y^{x_1}, Z^{x_2})$		$SSK_{c,s} \leftarrow H(X_1^y, X_2^z)$

Fig. 3: PriSrv Protocol

Scheme	ME	Anon.	Complex Policy	Selective Disclosure	No Pre-reg. Pairing	Hidden Policy
IBME [15], [31]	√	√	×	×	×	√
ACME	√	√	√	√	√	×

TABLE II: Comparison of ME Schemes

VII. PriSrv: Privacy-enhanced Fast Service Discovery

A. *PriSrv Protocol and Security*

Fig. 3 shows PriSrv, which consists of a service broadcast phase and an anonymous mutual authentication phase. A unique broadcast identifier bid is assigned to each broadcast cycle; and a unique session identifier sid is assigned to each session. A lifetime should be set for each broadcast cycle (e.g., 30 seconds) by including a timestamp (which can be part of bid) and a client verifies the timestamp upon successful decryption to ensure the freshness. Let $\mathcal{MAC} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{MAC}, \mathsf{Verify})$ be a message authentication code (MAC) scheme [35], [37], [38], and $H : \{0,1\}^* \rightarrow \mathcal{K}$ be a hash function, where $\mathcal{K}$ is the secret session key space. We assume that the generation and dissemination of anonymous credential, attribute and policy decryption keys to both service provider (S) and client (C) are performed according to ACME.

Service Broadcast Phase. To initiate a broadcast session with identifier bid, S defines a policy f_s to be satisfied by C. S selects an ephemeral Diffie-Hellman (DH) exponent $z \xleftarrow{\$} \mathbb{Z}_p^*$ and calculates $Z \leftarrow h^z$. S also runs $K_c \leftarrow \mathcal{MAC}.\mathsf{KeyGen}(1^\lambda)$ to generate an MAC key. S generates the broadcast message $MSG_B = \{bid \| Z \| Service_{Type} \| Service_{Par} \| K_c\}$ including the broadcast identifier, service type and parameters as well as a MAC key for the client. Next, S encrypts MSG_B to a broadcast ciphertext $\mathsf{CT}_B = \mathsf{CT}_{\vec{x}_s, f_s} \leftarrow \mathcal{ACME}.\mathsf{Enc}(\mathsf{cred}_s, \vec{x}_s, f_s, MSG_B)$. Then, the broadcast identifier bid and service ciphertext CT_B are announced over the public network.

Anonymous Mutual Authentication Phase. To establish a secure session between C and S, the anonymous mutual authentication is executed to establish a session key $SSK_{c,s}$.

(1) To discover the private service, C firstly checks whether $\vec{x}_c^{(out)}$ satisfies with the anonounced access policy f_s of S, i.e. $f_s(\vec{x}_c^{(out)}) \overset{?}{=} 1$. C quickly filters out mismatched services without decryption when $f_s(\vec{x}_c^{(out)}) = 0$. Otherwise, C attempts to decrypt CT_B using its attribute and policy decryption keys $(\mathsf{DK}_{\vec{x}_c}, \mathsf{DK}_{f_c})$. If the decryption fails which means $f_s(\vec{x}_c^{(out)}) = 0 \vee f_c(\vec{x}_s^{(out)}) = 0$, then C aborts. Otherwise, C responds to the broadcast message by executing $\mathcal{ACME}.\mathsf{Dec}$ to recover MSG_B. Next, C selects ephemeral DH exponents $x_1, x_2 \xleftarrow{\$} \mathbb{Z}_p^*$ and calculates $X_1 = g^{x_1}$, $X_2 = h^{x_2}$. C computes a MAC key K_s and an authentication tag σ_c of $M_c = (\text{"}C \rightarrow S\text{"}, bid, sid, X_1, X_2, Z)$ using K_c from MSG_B, where "$C \rightarrow S$" denotes the message direction. Then, C defines a policy f_c to be satisfied by S, and selects a set of public attributes and a set of private attributes to be disclosed to S. C runs $\mathcal{ACME}.\mathsf{Enc}$ to compute $\mathsf{CT}_c = \mathsf{CT}_{\vec{x}_c, f_c}$ and sends it to S.

(2) S authenticates C's service access request and computes a secret session key. S executes $\mathcal{ACME}.\mathsf{Dec}$ to recover MSG_c. S aborts the protocol if decryption fails. Next, S verifies σ_c and selects DH exponent $y \xleftarrow{\$} \mathbb{Z}_p^*$ to calculate $Y \leftarrow g^y$. S sets $M_s = (\text{"}S \rightarrow C\text{"}, bid, sid, X_1, X_2, Y, Z)$ and generates a tag $\sigma_s \leftarrow \mathcal{MAC}.\mathsf{MAC}(K_s, M_s)$ using K_s from MSG_c.

Then, S computes a secret session key $SSK_{c,s} \leftarrow H(X_1^y, X_2^z)$ and sends (M_s, σ_s) to C.

(3) Receiving the message, C checks the validity of σ_s. If it is valid, C computes $SSK_{c,s} \leftarrow H(Y^{x_1}, Z^{x_2})$ using the secret values (x_1, x_2). Therefore, C and S derive the same session key $SSK_{c,s}$ since $X_1^y = Y^{x_1} = g^{x_1 y} \in G_1$ and $X_2^z = Z^{x_2} = h^{x_2 z} \in G_2$.

The following theorem shows the security of PriSrv.

Theorem 7.1. Suppose that the DDH assumption holds, $\mathcal{ACME}$ is secure, $\mathcal{MAC}$ is unforgeable, and H is a random oracle, then PriSrv is a secure service discovery protocol and satisfies bilateral anonymity.

The proof of Theorem 7.1 is shown in Appendix C.

B. PriSrv Credential Management

Now we discuss credential management, including credential issuance, credential interoperability, and credential revocation.

Credential Issuance. PriSrv leverages FAC to implement a digital identity system for service providers and clients, offering the advantages of unforgeability, anonymous authentication, unlinkability, and selective attribute disclosure. W3C published Decentralized Identifiers (DIDs) [39] and Verifiable Credentials (VC) [40] specifications to regulate verifiable and decentralized digital identities. Decentralized Identity Foundation (DIF) [41] developed a set of standards to support a decentralized identity ecosystem [42]. Technology giants, such as IBM [43] and Microsoft [44], also provide flexible identity governance and administration services for credentials. CanDID proposed in [45] allows user's attributes to be verified by issuers or imported from existing authority systems. PriSrv may follow any of these existing DID frameworks to issue credentials.

Credential Interoperability. Credentials complying with standard specifications are interoperable across different platforms. DID [39] and VC [40] have regulated the process for inteoperable usage of credentials, which is also supported by DIF [41]. Backed by Microsoft, Google, Yahoo, IBM, VeriSign, PayPal, and Facebook, the OpenID Foundation[3] promotes identity management, federation and interoperation, in compliance with the specifications of W3C. PriSrv may follow these specifications to ensure credential interoperability when deployed in various service discovery settings.

Credential Revocation. Another consideration of PriSrv is to manage revocation of user's credentials whenever it is necessary. Credential revocation has been intensively studied in the last decade: various types of dynamic accumulators (such as RSA-based and bilinear map based) with ZKP are adopted for credential revocation [46], [47], [48]. It can also be achieved by the combination of ElGamal encryption and Schnorr proofs [49], or n-times unlinkable proofs [50]. PriSrv may incorporate the above techniques to realize credential revocation.

[3]OpenID Foundation: https://openid.net/foundation.

C. Interoperability of PriSrv with Existing Protocols

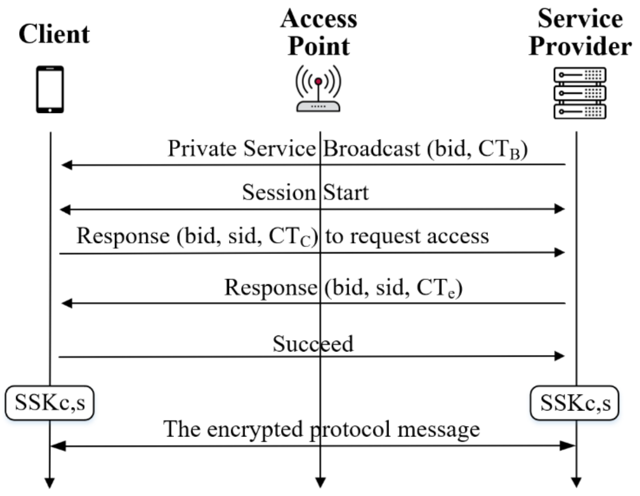
There are two approaches to make PriSrv work on top of/with different layers of different wireless protocols. The first approach is to position PriSrv at the application layer providing application payload to lower layers. If the payload of PriSrv is oversized in lower layers, the lower layer protocols need to perform segmentation on the sender side and assembling on the receiver side without changing the protocol logics. The second approach is to substitute target protocols at lower layers with PriSrv. However, the second approach requests for specific adaptations of the concret protocols. In the following, we give two examples for each approach, including mDNS and BLE for the first approach, and EAP, AirDrop for the second approach.

1) Privacy Enhanced mDNS and BLE: PriSrv can be integrated in the Vanadium[4] framework for developing privacy enhanced mDNS and BLE. Vanadium provides service discovery APIs to broadcast and scan services over widely deployed protocols, such as mDNS [51], [52] and BLE [3]. mDNS can work in conjunction with DNS Service Discovery (DNS-SD), a companion zero-configuration networking technique specified separately in RFC 6763[5]. DNS-SD extends the functionality of mDNS by adding additional attributes to the service discovery process. Specifically, the TXT (Text) resource record can be used to carry the attributes in the payload, where the maximum size for a single TXT record in DNS is 65535 bytes. The service broadcast of PriSrv is in the form (bid, CT_B), which takes 531996 bytes in communications on BN256 elliptic curve (100-bit security) [53]. Therefore, privacy enhanced mDNS may use 9 TXT records in DNS-SD to transmit the broadcast ciphertext of PriSrv.

On the other hand, the payload of BLE broadcast is constrained to 31 bytes, which is too small for carrying a broadcast ciphertext in PriSrv. To enable privacy enhanced BLE using PriSrv, the BLE Attribute Protocol (ATT) and Attribute Protocol Data Unit (PDU) Segmentation techniques can be leveraged to extend the payload size. If the payload exceeds the standard packet size in BLE, the ATT protocol (which is used for exchanging data between devices) can segment the payload data into multiple Attribute Protocol Data Units (PDUs) and transmit them sequentially. These PDUs can be reassembled on the receiver side to recover the original payload for the ciphertext in PriSrv.

2) Privacy Enhanced EAP: Figure 4 presents the architecture of privacy enhanced EAP using PriSrv, which extends RFC 3748 on Extensible Authentication Protocol (EAP) [54] to support private service discovery. An access point (AP) is involved in the interactions between client and service provider, which acts as a pass-through agent for a backend authentication server [54]. The anonymous authentication exchange in privacy enhanced EAP proceeds as follows. (1) At the beginning, the service provider announces private service broadcast information via AP, which contains the broadcast

[4]Vanadium. https://vanadium.github.io/.
[5]https://tools.ietf.org/html/rfc6763.



Client
Access Point
Service Provider
Private Service Broadcast (bid, CT_B)
Session Start
Response (bid, sid, CT_C) to request access
Response (bid, sid, CT_c)
Succeed
SSKc,s
SSKc,s
The encrypted protocol message

Fig. 4: Architecture of Privacy Enahnced EAP

identifier bid and the broadcast ciphertext $\mathsf{CT}_B = \mathsf{CT}_{\vec{x}_s, f_s} = \mathcal{ACME}.\mathsf{Enc}(\cdots, MSG_B)$. This step corresponds to the statement "the authenticator sends a request to authenticate the peer" in EAP Standard. (2) If the client can decrypt MSG_B from CT_B, he/she sends a response packet $(bid, sid, \sigma_c, \mathsf{CT}_c)$ as reply to the service provider, where $\mathsf{CT}_c = \mathsf{CT}_{\vec{x}_c, f_c} = \mathcal{ACME}.\mathsf{Enc}(\cdots, MSG_C)$ and sid is a session identifier. (3) Receiving the response, the service provider proceeds to recover MSG_c from CT_c. If it succeeds, the service provider sends (M_s, σ_s) to client, where M_s contains the DH shares for computing a session key and σ_s is the corresponding MAC value. After the client verifies σ_s, it calculates a secret session key $SSK_{c,s}$, and responds with a message "succeed". (4) Finally, the service provider also computes $SSK_{c,s}$ so that a secure session is established between service provider and client. All subsequent protocol messages are encapsulated in EAPOL frames and re-encapsulated as RADIUS packets on the back-haul. Following [9], the privacy enhanced EAP can be adopted to enhance the privacy of Wi-Fi connections.

3) Privacy Enhanced Apple AirDrop: AirDrop applies BLE to advertise the hashed identity of a service provider to look for potential clients in their proximity. If a match is confirmed, a TLS handshake is performed to exchange their certificates in cleartext. Both hashed identities and certificates are disclosed to the public, which is subject to identification and tracking attacks. Following the PrivateDrop mechanism in [16], we can improve the privacy of AirDrop by avoiding transmitting private information (such as identifier) of service provider during the advertising phase using BLE, and then encrypt the certificates of both parties using ACME at the beginning of TLS handshake. Apple may take the role of credential issuer in this case to generate necessary secret keys and credentials in addition to their existing iCloud certificates.

D. Limitations of PriSrv

One limitation of PriSrv lies in its large message size when compared to existing protocols. This large size of the outer discovery broadcast poses a scalability challenge, particularly on slower networks like BLE, resulting in high transmission overhead and reception delays. Moreover, on networks such as Wi-Fi, broadcasts must always be transmitted at the lowest feasible speed, further exacerbating airtime congestion.

The issue of large message sizes also compounds another challenge in wireless networks: packet loss, especially when using opportunistic transmission protocols like mDNS, which relies on UDP. Although UDP packets can theoretically reach sizes of up to 64K, they are fragmented to align with the Maximum Transmission Unit (MTU) of the physical network. Any loss of a single fragment results in the entire packet being discarded. While Wi-Fi incorporates a rudimentary acknowledgment and retry mechanism, this only applies to unicast traffic and can only recover from brief RF disruptions. Consequently, clients must wait for the broadcast ciphertext in the subsequent round to receive full packets, causing additional delays in reception. How to design efficient privacy-preserving discovery protocols remain an open problem for future research.

PriSrv protects its own payloads for achieving unlinkability at its positioned layer. As for achieving unlinkability at lower layers, the lower layer headers must be protected using specific anti-tracking mechanisms designed at lower layers. For example, PriSrv can work with MAC randomization mechanism at data link layer. Smartphone manufacturers (e.g., Apple iOS) incorporate MAC randomization for Wi-Fi and AWDL connections to provide unlinkability at the link layer, but devices can still be tracked at the layer where PriSrv resides. PriSrv complements the MAC randomization mechanism to realize unlinkability in different layers. Nevertheless, the current MAC address randomization approach (e.g., as implemented in Android and iOS) only performs randomization once when connecting to a new network and not with each subsequent connection. To achieve more robust unlinkability, a more effective MAC address randomization strategy should be devised to ensure unlinkability for each individual connection. Achieving unlinkability across multiple layers remains a persistent challenge.

VIII. Implementation and Comparison

We benchmark the performance of PriSrv on various hardware platforms, including desktop, laptop, smartphone, and Raspberry Pi as shown in Table III. Three asymmetric elliptic curves are selected from the MIRACL library [55] for evaluation, including MNT159 (80-bit security), MNT201 (90-bit security), and BN256 (100-bit security) [53]. We use AES-CTR with 100-bit keys to instantiate the $\mathsf{SEnc}/\mathsf{SDec}$ algorithms in PriSrv, using SHA-256 as the hash function, and use $\mathsf{MAC}_{\mathsf{GGM}}$ [35] as suf-cma secure MAC. The source code of our experiments is written in C/C++ and publicly available on GitHub[6]. For each test case, we report the average over 20 executions.

[6]Source Code: https://github.com/prisrv.

No.	Type	Hardware Platforms
1	Desktop	Intel® Core™ i9-7920X CPU @ 2.9GHz×12, 16GB
2	Laptop	Intel® Core™ i5-10210U CPU @ 1.6GHz×4, 8GB
3	Phone	ARM Cortex @2.84GHz+3×2.4GHz, 4GB
4	Raspberry Pi	ARM Cortex @1.5GHz×4, 2GB

TABLE III: Hardware Platforms for Experiments

A. Evaluation of FAC

In Table IV, we compare FAC with typical anonymous credential schemes. FAC constructs a constant-size anonymous credential. With FAC, a verifier only needs to conduct k operations to check the proof of k attributes, which is an up-to-date optimal solution. The $O(1)$ communication complexity in [29] for its Show algorithm (i.e., $|\mathsf{Show}|$) is composed of about 100 group elements. Since the scheme in [29] is the only one to achieve UC security in Table IV, these overheads can be seen as a tradoff between efficiency and security. Compared with [30], our credential only consists of 2 elements in G_2, which is approximately $2\times$ more efficient than that of [30] (i.e., $3|G_1| + |G_2| + 2|\mathbb{Z}_p|$). To show a credential in FAC, a user transmits 2 elements in G_1, 2 in G_2, 1 in G_T and three scalar elements, which is smaller than 8 elements in G_1, 1 in G_2 and two scalar elements for [30].

Ref.	Issue	|cred|	|Show|	Show	Verify
[56]	$O(1)$	$2\|QR_N\| + \|\ell_N\|$	$O(k)$	$O(k)$	$O(k)$
[57]	$O(1)$	$\|G_1\| + 2\|Z_q\|$	$O(n)$	$O(n)$	$O(n)$
[28]	$O(n)$	$(2n+4)(\|G_1\| + \|Z_q\|)$	$O(n)$	$O(n)$	$O(n)$
[29]	$O(1)$	$6\|G_1\| + 2\|G_2\| + \|Z_p\|$	$O(1)$	$O(n-k)$	$O(k)$
[30]	$O(1)$	$3\|G_1\| + \|G_2\| + 2\|Z_p\|$	$O(1)$	$O(n-k)$	$O(k)$
FAC	$O(1)$	$2\|G_2\|$	$O(1)$	$O(n-k)$	$O(k)$

TABLE IV: Comparison of Anonymous Credential Schemes
$|\mathsf{Show}|$ indicates the communication cost for showing k attributes. Show and Verify represent the computational costs. QR_N represents the group of quadratic residues modulo a composite N, and ℓ_N is an RSA moduli defined in [56].

Ref.	|Cred|	Issue	Show	Verify
Idemix [56]	0.671	76.437	283.245	210.783
UProve [57]	0.768	37.422	12.264	33.231
[29]	1.352	389.513	657.024	253.453
[30]	0.736	371.126	87.625	284.719
FAC	0.544	39.387	28.302	65.819

TABLE V: Performance of AC (ms/KB) (BN256)

Table V compares the performance of FAC with Idemix, UProve and the schemes in [29], [30] on desktop. The parameters for FAC are $n = 10$ and $|\mathcal{I}| = 4$. UProve incurs a low cost without providing multi-show unlinkability, while the other schemes support this privacy property. FAC has the smallest credential size (0.544 KB) in this comparison and its overheads for Issue, Show, Verify are the lowest or the second lowest among those supporting multi-show unlinkability.

B. Evaluation of ACME and PriSrv

Table VI presents the computation cost (comp.) and communication cost (comm.) of ACME for different algorithms on desktop following the example in §IV, where the parameters are $n = 10$, $k = 2$, $\hat{m} = 9$ and $|\mathcal{S}| = 9$. The system setup time, performed on various curves, ranges from 20.526 ms to 33.344 ms. The sizes of master public key ($|\mathsf{mpk}|$) and master secret key ($|\mathsf{msk}|$) for BN256 are 4.128 KB and 1.6 KB, respectively. The credential key generation ($\mathsf{CredKeyGen}$) and user key generation ($\mathsf{UserKeyGen}$) cost no more than 118.622 ms and 9.102 ms, respectively. The credential issue (Issue) algorithm is efficient (39.383 ms) and the size of generated anonymous credential ($|\mathsf{cred}|$) is merely 0.544 KB on BN256 curve, which is consistant with the theoretical analysis of FAC in §VIII-A. The size of attribute decryption key ($\mathsf{DK}_{\vec{x}}$) and the size of policy decryption key (DK_f) are no more than 2.72 KB and 44.064 KB, respectively. The computation costs for encryption and decryption are less than 188 ms and 232 ms, respectively, on BN256 curve. While the computation costs on MNT159 and MNT201 are significantly lower than those on BN256.

Comp. (ms)	Curve and Security Level		
	MNT159 (80-bit Security)	MNT201 (90-bit Security)	BN256 (100-bit Security)
Setup	20.526	26.882	33.344
CredKeyGen	98.261	105.883	118.622
UserKeyGen	6.153	7.582	9.102
Issue	29.298	33.783	39.383
DKGen	21.63	18.64	15.75
PolGen	359.807	327.796	237.675
Enc	146.931	167.337	187.822
Dec	123.772	188.346	231.214
Comm. (KB)	MNT159	MNT201	BN256
$\|\mathsf{mpk}\|/\|\mathsf{msk}\|$	1.044 / 1.2	1.332 / 1.36	4.128 / 1.6
$\|\mathsf{pk}\|/\|\mathsf{sk}\|$	0.91 / 0.18	1.158 / 0.204	3.408 / 0.24
$\|\mathsf{upk}\|/\|\mathsf{usk}\|$	0.116 / 0.03	0.148 / 0.034	0.4 / 0.04
$\|\mathsf{DK}_{\vec{x}}\|/\|\mathsf{DK}_f\|$	0.86 / 13.932	1.1 / 17.82	2.72 / 44.064
$\|\mathsf{cred}\|/\|\mathsf{CT}\|$	0.172 / 164.34	0.220 / 212.964	0.544 / 537.984

TABLE VI: Performance of ACME

Using the same example and parameter settings, Table VII provides a comprehensive evaluation of PriSrv on multiple hardware platforms with various elliptic curves and security levels. The communication overheads of the broadcast and mutual authentication phases are similar, as both of them are primarily determined by the size of the ACME ciphertext. The communication costs remain the same for different platforms, and the computation costs gradually increase from desktop to Raspberry Pi. The desktop, laptop and smartphone take less than 0.483 s for private service broadcast, and less than 0.973 s for anonymous mutual authentication. Raspberry Pi is relatively resource-limited, which takes 1.189 s and 2.712 s for private broadcast and authentication, respectively. The experimental results show that the broadcast and anonymous mutual authentication delays on the first three devices stay well

Device	Private Service Broadcast					
	MNT159 (80-bit Security)		MNT201 (90-bit Security)		BN256 (100-bit Security)	
	Comp.	Comm.	Comp.	Comm.	Comp.	Comm.
1	158.931	164.34	180.337	212.96	202.822	537.98
2	216.493	164.34	261.059	212.96	287.287	537.98
3	385.553	164.34	443.686	212.96	482.725	537.98
4	638.259	164.34	880.868	212.96	1188.392	537.98

Device	Anonymous Mutual Authentication					
	MNT159 (80-bit Security)		MNT201 (90-bit Security)		BN256 (100-bit Security)	
	Comp.	Comm.	Comp.	Comm.	Comp.	Comm.
1	429.282	164.45	517.512	213.09	673.039	538.83
2	576.161	164.45	686.054	213.09	854.177	538.83
3	727.572	164.45	892.712	213.09	972.163	538.83
4	1224.365	164.45	1832.187	213.09	2711.013	538.83

TABLE VII: Performance of PriSrv (ms/KB)

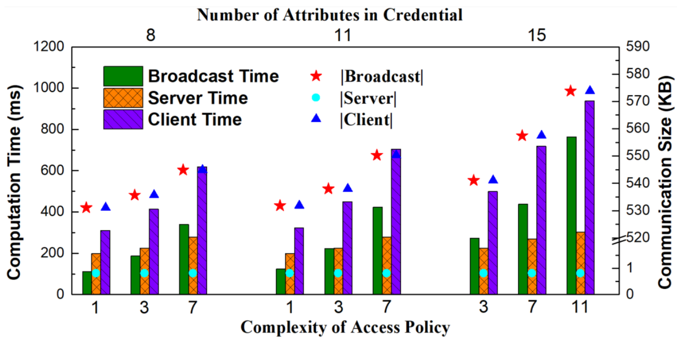


Fig. 5: Computation/Communication cost of PriSrv

below 1 s, which humans perceive the delays as an "immediate response" [16], [17], while the delays on Raspberry Pi are longer but not too significant.

We further implement PriSrv in wireless environment by adapting an open-source project of Wi-Fi Alliance [1], which implements IEEE 802.1X and enables the deployment of clients (running *wpa_supplicant* program of the project) and service providers (running *hostapd* program). Experiments of PriSrv in wireless communication use two laptops running Ubuntu 20.04. We deploy one laptop as the service provider and the other as the client. Fig. 5-6 present the broadcast time (T_B), server's computation time (T_S) and client's computation time (T_C) during the anonymous mutual authentication phase, where the total mutual authentication time is $T_{MA} = T_S + T_C$. The left y-axis shows the computation time, and right y-axis indicates the communication overhead in the broadcast phase (|Broadcast|) and the communication overhead of service provider/client in the authentication phase (|Server|/|Client|). The performance of PriSrv varies with the attribute number n (top x-axis) and the wire number $\hat{m}$ of NC^1 (i.e. number of shares for policy, bottom x-axis), where the matrix size is fixed to be $k = 2$.

In Fig. 5, we set $n = 8, 11, 15$ and vary the complexity of access policy $\hat{m}$ among $\{1, 3, 7, 11\}$ for practicality test. For $n = 15$, $\hat{m} = 11$, we have $T_B = 763.892$ ms, $T_S = 302.973$ ms and $T_C = 938.395$ ms, |Client|=573.852 KB and |Server| $= 0.82$ KB. Fig. 6 sets $n = 25, 50$ and varies $\hat{m}$ among $\{3, 7, 11, 13, 17, 21, 27\}$ for testing complex policies involving large number of attributes. The computation time increases with the number of attributes and complexity of access policies. For $n = 50$ and $\hat{m} = 27$, the computation costs are $T_B = 5.711$s, $T_S = 0.549$s, $T_C = 6.262$s. The communication cost in the broadcast phase grows from 0.549 MB to 0.881 MB. The transmission overhead of the server in the authentication phase remains relatively low (no more than 0.82 KB), while that of the client is mainly influenced by the ACME ciphertext, ranging from 0.549 MB to 0.881 MB. The comprehensive evaluations demonstrate the efficiency of PriSrv in wireless communications.

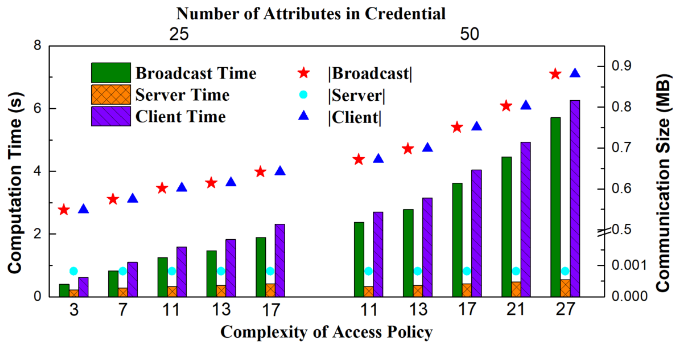


Fig. 6: Performance of PriSrv with Complex Policies

IX. Conclusion

This paper presented PriSrv, a privacy-enhanced service discovery protocol with high usability, for wireless communications. PriSrv enforces bilateral flexible policy control for anonymous mutual authentication, making it an ideal solution for enhancing privacy protection in popular wireless communication protocols such as EAP, mDNS, BLE, and AirDrop. PriSrv is built upon a novel primitive called anonymous credential-based matchmaking encryption (ACME), which extends the concept of ME proposed in CRYPTO'19 by offering selective attribute disclosure and eliminating the need for heavy cryptographic tools. ACME relies on a newly designed Fast Anonymous Credential (FAC) scheme to generate and verify authentication tokens that are unlinkable across multiple protocol sessions. Comprehensive experimental evaluations and comparisons demonstrated that ACME outperforms existing ME instantiations in terms of functionality and efficiency, which makes it a contribution of independent interests. Formal security models are provided to prove that PriSrv, ACME and FAC have desired security and privacy properties. Benchmarks on multiple hardware platforms demonstrated that PriSrv is suitable for interoperating with a wide range of service discovery protocols with enhanced privacy protection and high usability.

Acknowledgment

The authors would like to thank the shepherd and anonymous reviewers for their valuable comments and insightful suggestions. Yang Yang is supported by Lee Kong Chian Professor Fund, National Natural Science Foundation of China under Grant No. 62372110, and Fujian Provincial Natural Science of Foundation under Grant 2023J02008. Robert Deng is supported by AXA Research Fund. Guomin Yang is supported by Lee Kong Chian Fellowship awarded by Singapore Management University. HweeHwa Pang is supported by Lee Kong Chian Chair Professor Fund. Jian Weng is supported by National Key Research and Development Plan of China under Grant No. 2020YFB1005600, National Natural Science Foundation of China under Grant Nos. 61825203, 62332007 and U22B2028, Science and Technology Major Project of Tibetan Autonomous Region of China under Grant No. XZ202201ZD0006G, 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.

References

[1] WiFi [Online]. Available: https://w1.fi.

[2] How to use AirDrop on your iPhone or iPad. [Online]. Available: https://support.apple.com/en-us/HT204144.

[3] Bluetooth specification version 4.2. [Online]. Available: Bluetooth.com.

[4] B. Könings, C. Bachmaier, F. Schaub, M. Weber. Device names in the wild: Investigating privacy risks of zero configuration networking. In *MDM*, 2013.

[5] D. J. Wu, A. Taly, A. Shankar, D. Boneh. Privacy, discovery, and authentication for the internet of things. In *ESORICS*, 2016.

[6] W. Zhou, Y. Jia, Y. Yao, L. Zhu, L. Guan, Y. Mao, P. Liu, Y. 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.

[7] M. Stute, S. Narain, A. Mariotto, A. Heinrich, D. Kreitschmann, G. Noubir, M. 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.

[8] M. Stute, A. Heinrich, J. Lorenz, M. 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.

[9] A. Cassola, E. O. Blass, G. Noubir. Authenticating privately over public Wi-Fi hotspots. In *CCS*, 2015.

[10] K. Fawaz, K. H. Kim, K. G. Shin. Protecting privacy of BLE device users. In *USENIX Security*, 2016.

[11] X. Na, X. Guo, Y. He, R. Xi. Wi-attack: cross-technology impersonation attack against iBeacon services. In *SECON*, 2021.

[12] X. Bai, L. Xing, N. Zhang, X. Wang, X. Liao, T. Li, S. M. Hu. Staying secure and unprepared: understanding and mitigating the security risks of apple zeroconf. In *S&P*, 2016.

[13] X. Bai, L. Xing, N. Zhang, X. Wang, X. Liao, T. Li, S. M. Hu. Apple ZeroConf holes: how hackers can steal iPhone photos. *IEEE Security & Privacy Magazine*, 2017, 15(2): 42-49.

[14] J. Xu, Y. Liu, H. Shi. A survey on privacy-preserving wireless network protocols: techniques and challenges. *IEEE Communications Surveys & Tutorials*, 2020: 22(1), 572-598.

[15] G. Ateniese, D. Francati, D. Nuñez, D. Venturi. Match me if you can: matchmaking encryption and its applications. *Journal of Cryptology*, 2021, 34: 1-50.

[16] A. Heinrich, M. Hollick, T. Schneider, M. Stute, C. Weinert. PrivateDrop: practical privacy-preserving authentication for Apple airDrop. In *USENIX Security*, 2021.

[17] S. K. Stuart, G. G. George, J. D. Jock. The Information Visualizer, an Information Workspace. In *CHI*, 1991.

[18] S. Cheshire, M. Krochmal. RFC 6763: DNS-based service discovery. 2013.

[19] S. Cheshire, M. Krochmal. RFC 6762: Multicast DNS. 2013.

[20] Y. Y. Goland, T. Cai, P. Leach, Y. Gu. Simple service discovery protocol/1.0 operating without on arbiter. IETF INTERNET-DRAFT draft-cai-ssdp-v1-03. 1999.

[21] M. Boucadair, R. Penno, D. Wing. Universal Plug and Play (UPnP) Internet Gateway Device - Port Control Protocol Interworking Function (IGD-PCP IWF). RFC 6970: Multicast DNS. 2013.

[22] X. Wang, Y. Sun, S. Nanda, X. Wang. Looking from the Mirror: Evaluating IoT Device Security through Mobile Companion Apps. In *USENIX Security*, 2019.

[23] L. Yu, B. Luo, J. Ma, Z. Zhou, Q. Liu. You Are What You Broadcast: Identification of Mobile and IoT Devices from (Public) WiFi. In *USENIX Security*, 2020.

[24] J. Wu, Y. Nan, V. Kumar, D. Tian, A. Bianchi, M. Payer, D. Xu. BLESA: Spoofing Attacks against Reconnections in Bluetooth Low Energy. In *WOOT@USENIX Security*, 2020.

[25] R. H. Venkatnarayan, M. Shahzad, S. Yun, C. Vlachou, K. H. Kim. Leveraging polarization of WiFi signals to simultaneously track multiple people. In *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 2020, 4(2): 1-24.

[26] L. Kowalczyk, H. Wee. Compact adaptively secure ABE for NC1 from k-Lin. *Journal of Cryptology*, 2020, 33(3): 954-1002.

[27] S. Katsumata, R. Nishimaki, S. Yamada, T. Yamakawa. Compact NIZKs from standard assumptions on bilinear maps. In *EUROCRYPT*, 2020.

[28] S. Ringers, E. Verheul, J. H. Hoepman. An efficient self-blindable attribute-based credential scheme. In *FC*, 2017.

[29] J. Camenisch, M. Dubovitskaya, K. Haralambiev, M. Kohlweiss. Composable and modular anonymous credentials: definitions and practical constructions. In *ASIACRYPT*, 2015.

[30] G. Fuchsbauer, C. Hanser, D. Slamanig. Structure-preserving signatures on equivalence classes and constant-size anonymous credentials. *Journal of Cryptology*, 2019.

[31] J. Chen, Y. Li, J. Wen, J. Weng. Identity-based matchmaking encryption from standard assumptions. In *ASIACRYPT*, 2022.

[32] R. Canetti, H. Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In *EUROCRYPT*, 2001.

[33] R. Canetti, H. Krawczyk. Security analysis of IKE's signature-based key-exchange protocol. In *CRYPTO*, 2002.

[34] J. Camenisch, M. Drijvers, M. Dubovitskaya. Practical UC-secure delegatable credentials with attributes and their application to blockchain. In *CCS*, 2017.

[35] M. Chase, S. Meiklejohn, G. Zaverucha. Algebraic MACs and keyed-verification anonymous credentials. In *CCS*, 2014.

[36] O. Sanders. Efficient redactable signature and application to anonymous credentials. In *PKC*, 2020.

[37] M. Chase, T. Perrin, G. Zaverucha. The signal private group system and anonymous credentials supporting efficient verifiable encryption. In *CCS*, 2020.

[38] Z. Zhang, K. Yang, X. Hu, Y. Wang. Practical anonymous password authentication and TLS with anonymous client authentication. In *CCS*, 2016.

[39] D. Reed, M. Sporny, D. Longley, C. Allen, R. Grant, M. Sabadello, J. Holt. Decentralized identifiers (DIDs) v1.0: Core architecture, data model, and representations. W3C Working Draft, 2020.

[40] M. Sporny, D. Longley, D. Chadwick. Verifiable credentials data model v1.1. Available: https://www.w3.org/TR/vc-data-model, 2022.

[41] DIF. Decentralized Identity Foundation. [Online]. Available: https://identity.foundation/.

[42] L. Lesavre, P. Varin, P. Mell, M. Davidson, J. Shook. A taxonomic approach to understanding emerging blockchain identity management systems. NIST Cybersecurity White Paper, 2019.

[43] IBM. Blockchain for Digital Identity and Credentials. [Online]. Available: https://www.ibm.com.

[44] Microsoft. Decentralized Identity, Blockchain, and Privacy. [Online]. Available: https://www.microsoft.com.

[45] D. Maram, H. Malvai, F. Zhang, N. Jean-Louis, A. Frolov, T. Kell, T. Lobban, C. Moy, A. Juels, A. Miller. Candid: Can-do decentralized identity with legacy compatibility, sybil-resistance, and accountability. In *S&P*, 2021.

[46] D. Boneh, B. Bünz, B. Fisch. Batching techniques for accumulators with applications to IOPs and stateless blockchains. In *CRYPTO*, 2019.
[47] J. Camenisch, M. Kohlweiss, C. Soriente. An accumulator based on bilinear maps and efficient revocation for anonymous credentials. In *PKC*, 2009.
[48] F. Baldimtsi, J. Camenisch, M. Dubovitskaya, A. Lysyanskaya, L. Reyzin, K. Samelin, S. Yakoubov. Accumulators with applications to anonymity-preserving revocation. In *EuroS&P*, 2017.
[49] D. Bogatov, A. De Caro, K. Elkhiyaoui, B. Tackmann. Anonymous transactions with revocation and auditing in hyperledger fabric. In *CANS*, 2021.
[50] J. Camenisch, M. Drijvers, J. Hajny. Scalable revocation scheme for anonymous credentials based on n-times unlinkable proofs. In *WPES*, 2016.
[51] S. Cheshire, M. Krochmal. DNS-based service discovery. Technical Report. RFC 6763, 2013.
[52] S. Cheshire, M. Krochmal. Multicast DNS. Technical Report. RFC 6762, 2013.
[53] Pairing-Friendly Curves. [Online]. Available: https://www.ietf.org/archive/id/draft-irtf-cfrg-pairing-friendly-curves-02.html.
[54] B. Aboba, L. Blunk, J. Vollbrecht, J. Carlson, H. Levkowetz. Extensible authentication protocol (EAP). 2004.
[55] MIRACL: Multiprecision integer and rational arithmetic c/c++ library. [Online]. Available: https://github.com/miracl/MIRACL.
[56] J. Camenisch, A. Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *EUROCRYPT*, 2001.
[57] C. Paquin, G. Zaverucha. U-prove cryptographic specification v1.1. Technical Report, Microsoft Corporation, 2011.
[58] M. Bellare, R. Canetti, H. Krawczyk. A modular approach to the design and analysis of authentication and key exchange protocols. In *STOC*, 1998.
[59] E. Fujisaki, T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In *CRYPTO*, 1999.
[60] H. Krawczyk, H. Wee. The OPTLS protocol and TLS 1.3. In *EuroS&P*, 2016.

Appendix

A. FAC: Security Model and Proof

(1) Security Model

Following the definitions in [35], [36], we define *correctness*, *unforgeability*, *anonymity* and *unlinkability* for anonymous credential scheme. The lists in the security models are given in Table VIII.

Definition A.1 Let $\mathcal{D}$ be the universe of user identity, and Ω be the universe of attribute sets. Then an anonymous credential scheme $\mathcal{AC}$ is *correct* for $\mathcal{D},\Omega$ if all $uid\in\mathcal{D}$, all $\vec{x}\subseteq\Omega$, for all security parameter λ,

$$\Pr\left[\begin{array}{l} \mathsf{pp}\xleftarrow{\$}\mathsf{Setup}(1^\lambda,1^n);(\mathsf{pk},\mathsf{sk})\xleftarrow{\$}\mathsf{CredKeyGen}(\mathsf{pp});\\ (\mathsf{upk},\mathsf{usk})\xleftarrow{\$}\mathsf{UserKeyGen}(\mathsf{pp});\\ \mathsf{cred}\xleftarrow{\$}\langle\mathsf{Issue.I}(\mathsf{sk},\mathsf{upk})\rightleftarrows\mathsf{Issue.U}(uid,\vec{x},\mathsf{usk})\rangle;\\ \mathsf{tok}\xleftarrow{\$}\mathsf{Show}(uid,\{x_i\}_{i\in\mathcal{I}},\mathsf{cred},\mathsf{usk},m):\\ \mathsf{Verify}(\mathsf{tok},m)=0\end{array}\right]$$

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

Definition A.2 An AC scheme satisfies *unforgeability* if for any PPT adversary $\mathcal{A}$, there exists a negligible function ν such that $\mathsf{Adv}^{\mathsf{unforge}}_{\mathcal{AC}}(\lambda)\overset{\mathsf{def}}{=}$

$$\Pr\left[b=1\left|\begin{array}{l} \mathsf{pp}\leftarrow\mathsf{Setup}(1^\lambda,1^n),\\ (\mathsf{pk,sk})\leftarrow\mathsf{CredKeyGen}(\mathsf{pp})\\ (uid^*,\vec{x}^*,m^*,\mathsf{cred}^*,\mathsf{tok}^*)\leftarrow\mathcal{A}^{\mathcal{O}(\mathsf{sk},\cdot)}(\mathsf{pp},\mathsf{pk})\\ b\leftarrow\mathsf{Verify}(\mathsf{tok}^*,m^*)\\ \text{return } b \text{ if }\ (m^*,\vec{x}^*,\mathsf{cred}^*,\mathsf{tok}^*)\notin\mathcal{L}_{\mathsf{show}}\\ \qquad\wedge\ \mathsf{cred}^*\notin\mathcal{L}_{\mathsf{issue}}\wedge\ (uid^*,\vec{x}^*)\notin\mathcal{L}_{\mathsf{corrupt}}\\ \text{else abort}\end{array}\right.\right]$$

$\le\nu(\lambda)$, where the oracle set $\mathcal{O}=\{\mathsf{UserKeyGen},\mathsf{Issue},\mathsf{Show},\mathsf{Corrupt}\}$ is implemented by $\mathsf{UserKeyGen}(\mathsf{pp},\cdot)$, $\mathsf{Issue}(\mathsf{sk},\mathsf{st},\cdot)$, $\mathsf{Show}(\mathsf{pk},\cdot)$ and $\mathsf{Corrupt}(\cdot)$.

Definition A.3 An AC scheme $\mathcal{AC}$ satisfies *anonymity* if for any PPT adversary $\mathcal{A}$, there exists a negligible function ν such that $\mathsf{Adv}^{\mathsf{anon}}_{\mathcal{AC}}(\lambda)\overset{\mathsf{def}}{=}$

$$\Pr\left[b'=b\left|\begin{array}{l} \mathsf{pp}\leftarrow\mathsf{Setup}(1^\lambda,1^n),\\ (\mathsf{pk,sk})\leftarrow\mathsf{CredKeyGen}(\mathsf{pp})\\ (uid_0^*,uid_1^*,\vec{x}^*,m^*)\leftarrow\mathcal{A}^{\mathcal{O}(\mathsf{sk},\cdot)}(\mathsf{pp},\mathsf{pk})\\ \text{abort if }\exists d\xleftarrow{\$}\{0,1\}:\\ (uid_d^*,\vec{x}^*,\mathsf{cred}_d)\notin\mathcal{L}_{\mathsf{honest}};\ b\xleftarrow{\$}\{0,1\}\\ \mathsf{tok}^*\leftarrow\mathsf{Show}(uid_b^*,\mathsf{cred}_b,\mathsf{usk}_b,\vec{x}^*,m^*)\\ b'\leftarrow\mathcal{A}^{\mathcal{O}(\mathsf{sk},\cdot)}(\mathsf{pp},\mathsf{tok}^*)\\ \text{return } b' \text{ if }\ \ \mathsf{cred}^*\notin\mathcal{L}_{\mathsf{issue}}\wedge\\ (m^*,\vec{x}^*,\mathsf{tok}^*)\notin\mathcal{L}_{\mathsf{show}}\ \wedge\ (uid_{0/1}^*,\vec{x}^*)\notin\mathcal{L}_{\mathsf{corrupt}}\\ \text{else abort}\end{array}\right.\right]$$

$\le\nu(\lambda)$, where the oracle set $\mathcal{O}=\{\mathsf{UserKeyGen},\mathsf{Issue},\mathsf{Show},\mathsf{Corrupt}\}$ is implemented by $\mathsf{UserKeyGen}(\mathsf{pp},\cdot)$, $\mathsf{Issue}(\mathsf{sk},\mathsf{st},\cdot)$, $\mathsf{Show}(\mathsf{pk},\cdot)$ and $\mathsf{Corrupt}(\cdot)$.

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

$$\Pr\left[b'=b\left|\begin{array}{l} \mathsf{pp}\leftarrow\mathsf{Setup}(1^\lambda,1^n),\\ (\mathsf{pk,sk})\leftarrow\mathsf{CredKeyGen}(\mathsf{pp})\\ (uid_0^*,uid_1^*,\mathcal{I}^*,\vec{x}_0^*,\vec{x}_1^*,m^*)\leftarrow\mathcal{A}^{\mathcal{O}(\mathsf{sk},\cdot)}(\mathsf{pp},\mathsf{pk})\\ \text{where }\mathcal{I}^*\subseteq[1,n],\vec{x}_d^*=\{x_i^{(d)}\}_{i\in[1,n]},d\in\{0,1\}\\ \text{abort if }\exists j\in\mathcal{I}^*:x_j^{(0)}\neq x_j^{(1)}\\ \text{or }\exists d\xleftarrow{\$}\{0,1\}:(uid_d^*,\vec{x}_d^*,\mathsf{cred}_d)\notin\mathcal{L}_{\mathsf{honest}},\\ b\xleftarrow{\$}\{0,1\}\\ \mathsf{tok}_b^*\leftarrow\mathsf{Show}(uid_b^*,\mathsf{cred}_b^*,\mathsf{usk}_b^*,\{x_i^{(b)}\}_{i\in\mathcal{I}^*},m^*)\\ b'\leftarrow\mathcal{A}^{\mathcal{O}(\mathsf{sk},\cdot)}(\mathsf{pp},\mathsf{tok}_b^*)\\ \text{return } b' \text{ if }\ \forall d'\in\{0,1\},\quad \mathsf{cred}_{d'}^*\notin\mathcal{L}_{\mathsf{issue}}\wedge\\ (m^*,\vec{x}_{d'}^*,\mathsf{tok}_{d'}^*)\notin\mathcal{L}_{\mathsf{show}}\ \wedge\ (uid_{d'}^*,\vec{x}_{d'}^*)\notin\mathcal{L}_{\mathsf{corrupt}}\\ \text{else abort}\end{array}\right.\right]$$

$\le\nu(\lambda)$, where the oracle set $\mathcal{O}=\{\mathsf{UserKeyGen},\mathsf{Issue},\mathsf{Show},\mathsf{Corrupt}\}$ is implemented by $\mathsf{UserKeyGen}(\mathsf{pp},\cdot)$, $\mathsf{Issue}(\mathsf{sk},\mathsf{st},\cdot)$, $\mathsf{Show}(\mathsf{pk},\cdot)$ and $\mathsf{Corrupt}(\cdot)$.

It is easy to see that the security definition of *unlinkability* implicitly implies that of *anonymity*.

List	**Description**
$\mathcal{L}_{\mathsf{registerd}}$	registered users
$\mathcal{L}_{\mathsf{issue}}$	credentials that have been issued
$\mathcal{L}_{\mathsf{show}}$	credentials that have been shown
$\mathcal{L}_{\mathsf{honest}}$	registered users that are honest
$\mathcal{L}_{\mathsf{corrupt}}$	registered users that are corrupted
$\mathcal{L}_{\mathsf{auth}}$	authorized users in service discovery session
$\mathcal{L}_{\mathsf{exec}}$	contains all messages that user or adversary exchanged during executions of protocol

TABLE VIII: Lists in Security Experiments

(2) Correctness Proof

Theorem A.1 The FAC scheme satisfies *correctness*.

Proof. For the correctness proof, we need to demonstrate the following properties: 1) the instantiation of SPK π_1 is correct in the UserKeyGen algorithm; 2) the credential created by the issuer is verified true by the user in the Issue algorithm; 3) the instantiation of SPK π_2 is correct in Show and Verify algorithms.

For the *first property*, The correctness of SPK π_1 is verified as $\gamma=h^{\widetilde{\mathsf{usk}}}=h^{\overline{\mathsf{usk}}}(h^{\mathsf{usk}})^c=h^{\overline{\mathsf{usk}}}\mathsf{upk}^c$, where $\overline{\mathsf{usk}}=\widetilde{\mathsf{usk}}-c\cdot\mathsf{usk}\mod p$ and $\mathsf{upk}=h^{\mathsf{usk}}$.

For the *second property*, a received credential is $\mathsf{cred}=(\sigma_1,\sigma_2)$, $\sigma_1=h^r$, $\sigma_2=\mathsf{upk}^{r\cdot y_0}\cdot h^{r(\tau+\sum_{i=1}^n y_ix_i+y_{n+1}\cdot uid)}$, and the user verifies it as $e(W\cdot Y_0^{\mathsf{usk}}\cdot Y_{n+1}^{uid}\prod_{i=1}^n Y_i^{x_i},\sigma_1)\overset{?}{=}e(g,\sigma_2)$. This equation holds since

$$\begin{aligned} & e(W\cdot Y_0^{\mathsf{usk}}\cdot Y_{n+1}^{uid}\prod_{i=1}^n Y_i^{x_i},\sigma_1)\\ = \ & e(g^\tau\cdot(g^{y_0})^{\mathsf{usk}}\cdot(g^{y_{n+1}})^{uid}\prod_{i=1}^n(g^{y_i})^{x_i},h^r)\\ = \ & e(g,(h^{\mathsf{usk}})^{r\cdot y_0}h^{r(\tau+\sum_{i=1}^n y_ix_i+y_{n+1}\cdot uid)})\end{aligned}$$

$$= \quad e(g, \mathsf{upk}^{r\cdot y_0} \cdot h^{r(\tau+\sum_{i=1}^{n} y_i x_i + y_{n+1}\cdot uid)}) = e(g, \sigma_2).$$

For the *third property*, assume that cred is a valid anonymous credential on uid, $\vec{x}$ and usk, and we have

$$e(W \cdot Y_0^{\mathsf{usk}} \cdot Y_{n+1}^{uid} \prod_{i=1}^{n} Y_i^{x_i}, \sigma_1) = e(g, \sigma_2),$$

$$\Rightarrow e(Y_0^{\mathsf{usk}} \cdot Y_{n+1}^{uid}, \sigma_1) = e(g, \sigma_2) e(W \cdot \prod_{i=1}^{n} Y_i^{x_i}, \sigma_1)^{-1}.$$

The correctness of SPK π_2 can be derived from

$$\begin{aligned}
& e(Y_0^{\overline{\mathsf{usk}}} Y_{n+1}^{\overline{uid}}, \bar{\sigma}_1)^{-1} \cdot \Lambda \\
= \quad & e(Y_0^{\overline{\mathsf{usk}}} Y_{n+1}^{\overline{uid}}, \bar{\sigma}_1)^{-1} \cdot e(Y_0^{\widetilde{\mathsf{usk}}} Y_{n+1}^{\widetilde{uid}}, \bar{\sigma}_1) \\
= \quad & e(Y_0^{\mathsf{usk}} Y_{n+1}^{uid}, \sigma_1)^{c\cdot t_2} \\
= \quad & [e(g, \sigma_2) e(W \cdot \prod_{i\in[1,n]} Y_i^{x_i}, \sigma_1)^{-1}]^{c\cdot t_2} \\
= \quad & [e(g, \bar{\sigma}_2) e(g, \bar{\sigma}_1)^{-t_1} e(W \cdot \prod_{i\in[1,n]} Y_i^{x_i}, \bar{\sigma}_1)^{-1}]^{c} \\
= \quad & [e(g, \bar{\sigma}_2) e(W \cdot g^{t_1} \cdot \prod_{i\in[1,n]\setminus\mathcal{I}} Y_i^{x_i} \prod_{i\in\mathcal{I}} Y_i^{x_i}, \bar{\sigma}_1)^{-1}]^{c} \\
= \quad & [e(g, \bar{\sigma}_2) e(W \cdot T_1 \cdot \prod_{i\in\mathcal{I}} Y_i^{x_i}, \bar{\sigma}_1)^{-1}]^{c} = [e(g, \bar{\sigma}_2) \cdot \Gamma]^{c}.
\end{aligned}$$

On the other hand,

$$\begin{aligned}
& e(T_1, \prod_{i\in\mathcal{I}'} X_i) \\
= \quad & e(g^{t_1}, \prod_{i\in\mathcal{I}'} X_i) e(\prod_{j\in[1,n]\setminus\mathcal{I}} Y_j^{x_j}, \prod_{i\in\mathcal{I}'} X_i) \\
= \quad & e(\prod_{i\in\mathcal{I}'} Y_i, h)^{t_1} e(\prod_{i\in\mathcal{I}', j\in[1,n]\setminus\mathcal{I}} Z_{i,j}^{x_j}, h) \\
= \quad & e((\prod_{i\in\mathcal{I}'} Y_i)^{t_1} \prod_{i\in\mathcal{I}', j\in[1,n]\setminus\mathcal{I}} Z_{i,j}^{x_j}, h) = e(T_2, h).
\end{aligned}$$

Therefore, a valid credential cred and its SPK π_2 will be verified true. □

(3) Security Proof

The security of FAC in Theorem 5.1 is proved in aspects of *unforgeability*, *anonymity* and *unlinkability*.

3.1) Unforgeability Proof of FAC

Lemma A.1 The FAC scheme is *unforgeable* if the underlying unforgeable redactable signature (URS) [36] is unforgeable and the discrete logarithm (DL) assumption holds.

Proof. The proof reduces the unforgeability of FAC to the existential unforgeability of the unlinkable redactable signature (URS) scheme (in §4 of [36]) and discrete logarithm (DL) assumption. Let $\mathcal{A}$ be a PPT adversary that wins the unforgeability game with probability ϵ.

During the challenge phase, $\mathcal{A}$ returns a challenge user identifier uid^*, attribute set $\vec{x}^*$ and proves possession of a valid authentication token tok^* for credential cred^* on $(uid^*, \vec{x}^*)$. Obviously, it should be constrained that $(m^*, \vec{x}^*, \mathsf{cred}^*, \mathsf{tok}^*) \notin \mathcal{L}_{\mathsf{show}}$, $\mathsf{cred}^* \notin \mathcal{L}_{\mathsf{issue}}$, $(uid^*, \vec{x}^*) \notin \mathcal{L}_{\mathsf{corrupt}}$. Let usk^* be the secret key whose knowledge should be proved by $\mathcal{A}$ when it generates a challenge credential cred^* on $(uid^*, \vec{x}^*)$. Denote $\mathcal{L}_{\mathsf{honest}}$ as a set of registered users that are honest and $\mathcal{L}_{\mathsf{corrupt}}$ as a set of registered users that are corrupted. Denote q as the number of honest user. We define two types of adversaries $(\mathcal{A}_1, \mathcal{A}_2)$ that possess different resources for the attack: type-1 adversary $\mathcal{A}_1$: $\exists uid_{\beta^*} \in \mathcal{L}_{\mathsf{honest}}$, s.t., $\mathsf{usk}_{\beta^*} = \mathsf{usk}^*$; type-2 adversary $\mathcal{A}_2$: $\forall uid_i \in \mathcal{L}_{\mathsf{honest}}$, s.t., $\mathsf{usk}_i \neq \mathsf{usk}^*$. In the following, we prove the unforgeability of FAC with two propositions for the two types of adversaries.

Proposition A.1 Suppose type-1 adversary $\mathcal{A}_1$ is able to break the unforgeability of FAC with advantage ϵ_1. Then, we can utilize $\mathcal{A}_1$ to solve the DL problem with advantage ϵ_1/q, where q is the number of honest users.

Proof. The proof reduces the unforgeability of FAC to the security of DL assumption. Let $\mathcal{A}_1$ be a PPT adversary that wins the security game with probability ϵ_1. Consider a simulator $\mathcal{S}$ which runs $\mathcal{A}_1$ as a subroutine and interacts with a challenger $\mathcal{C}$ for the DL-assumption. According to the definition of type-1 adversary $\mathcal{A}_1$, there exists an index $\beta^* \in [1, q]$ for $\mathcal{A}_1$ to impersonate the β^*-th honest user in $\mathcal{L}_{\mathsf{honest}}$. Then, the challenger $\mathcal{C}$ is requested to make a guess on β^* from the q honest users. If $\mathcal{A}_1$ is able to break the unforgeability of FAC, $\mathcal{C}$ could makes use of the advantage of $\mathcal{A}_1$ to solve the DL problem. Let g, h be the generators of groups G_1 and G_2, respectively. Let (h, h^a) be challenge tuple of DL assumption on group G_2.

Setup. $\mathcal{S}$ creates the public key $\mathsf{pk} = (W, \{X_i, Y_i\}_{i\in[0,n+1]}, \{Z_{i,j}\}_{0\le i\neq j\le n+1})$ following the Setup and CredKeyGen algorithms in FAC and forwards it to $\mathcal{A}_1$.

Query. The adversary $\mathcal{A}_1$ adaptively makes the following queries.

– According to the definition of type-1 adversary $\mathcal{A}_1$, there exists an index $\beta^* \in [1, q]$ for $\mathcal{A}_1$ to impersonate the β^*-th honest user in $\mathcal{L}_{\mathsf{honest}}$. For the UserKeyGen queries on $(uid_{\beta^*}, \vec{x}_{\beta^*})$, $\mathcal{S}$ implicitly sets $usk_{\beta^*} = a$, and sends $\mathsf{upk}_{\beta^*} = h^a$ to $\mathcal{A}_1$. For the UserKeyGen queries on $(uid_j, \vec{x}_j)$ with $j \neq i$, $\mathcal{S}$ generates user's secret key $\mathsf{usk}_j \xleftarrow{\$} \mathbb{Z}_p^*$ and public key $\mathsf{upk}_j = h^{\mathsf{usk}_j}$, which are returned to $\mathcal{A}_1$. $\mathcal{S}$ adds $(uid, \vec{x}, \mathsf{upk})$ to a list $\mathcal{L}_{\mathsf{honest}}$. If $(uid, \vec{x})$ already exists in $\mathcal{L}_{\mathsf{honest}}$, $\mathcal{S}$ just replies with the same answer.

– For the Issue queries on $(uid_j, \vec{x}_j)$ with restriction that $j \neq \beta^*$, $\mathcal{S}$ runs UserKeyGen to generate $(\mathsf{upk}_j, \mathsf{usk}_j)$ if $(uid_j, \vec{x}_j)$ has not been queried beforehand. Otherwise, $\mathcal{S}$ extracts user's keys $(\mathsf{upk}_j, \mathsf{usk}_j)$ from $\mathcal{L}_{\mathsf{honest}}$. Then, $\mathcal{S}$ queries $\mathcal{C}$ on $(uid_j, \vec{x}_j)$ and obtains a URS signature $\sigma_j = (\widetilde{\sigma}_1, \widetilde{\sigma}_2)$, where $\widetilde{\sigma}_1 \xleftarrow{\$} G_2$ and $\widetilde{\sigma}_2 \leftarrow \widetilde{\sigma}_1^{\tau+\sum_{i=1}^{n} y_i\cdot x_i}$. $\mathcal{S}$ implicitly sets $\sigma_1 = \widetilde{\sigma}_1 = h^r$, computes

$$\begin{aligned}
\sigma_2 \quad &= \quad (\widetilde{\sigma}_1)^{\mathsf{usk}_j\cdot y_0} (\widetilde{\sigma}_1)^{y_{n+1}\cdot uid} \widetilde{\sigma}_2 \\
&= \quad (h_j^{\mathsf{usk}})^{r\cdot y_0} h^{r(\tau+\sum_{i=1}^{n} y_i x_i + y_{n+1}\cdot uid)} \\
&= \quad \mathsf{upk}^{r\cdot y_0} \cdot h^{r(\tau+\sum_{i=1}^{n} y_i x_i + y_{n+1}\cdot uid)},
\end{aligned}$$

and returns $\mathsf{cred}_j \leftarrow (\sigma_1, \sigma_2)$ to $\mathcal{A}_1$. $\mathcal{S}$ inserts $(uid_j, \vec{x}_j, \mathsf{upk}_j, \mathsf{cred}_j)$ into a list $\mathcal{L}_{\mathsf{issue}}$.

– To answer the Show query on $(uid_j, \{x_i\}_{i\in\mathcal{I}}, \mathsf{cred}_j, m_j)$ with restriction that $j \neq \beta^*$, $\mathcal{S}$ runs UserKeyGen to generate $(\mathsf{upk}_j, \mathsf{usk}_j)$ if uid_j has not been queried beforehand. Otherwise, $\mathcal{S}$ extracts user's keys $(\mathsf{upk}_j, \mathsf{usk}_j)$ from $\mathcal{L}_{\mathsf{honest}}$. Then,

$\mathcal{S}$ answers the query by executing the Show algorithm and returns the token tok to $\mathcal{A}_1$. $\mathcal{S}$ inserts $(uid_j, \{x_i\}_{i\in\mathcal{I}}, \mathsf{cred}_j, m_j, \mathsf{tok}_j)$ into a list $\mathcal{L}_{\mathsf{show}}$.

– To answer the Corrupt request on $(uid_j, \vec{x}_j) \in \mathcal{L}_{\mathsf{honest}}$ with restriction that $j \neq \beta^*$, simulator $\mathcal{S}$ returns the corresponding user secret key usk_j, credential cred_j and the token tok_j to $\mathcal{A}_1$, which are recorded in $\mathcal{L}_{\mathsf{honest}}$, $\mathcal{L}_{\mathsf{issue}}$ and $\mathcal{L}_{\mathsf{show}}$, respectively. $\mathcal{S}$ inserts $(uid_j, \vec{x}_j)$ into $\mathcal{L}_{\mathsf{corrupt}}$. If $(uid_j, \vec{x}_j)$ does exist in these lists, $\mathcal{S}$ returns $\perp$.

Challenge. Adversary $\mathcal{A}_1$ outputs a challenge tuple $(\vec{x}^*, m^*, \mathsf{cred}^*, \mathsf{tok}^*)$ with attributes $\vec{x}^*$, a message m^* and an authentication token tok^* for the β^*-th honest user with uid_{β^*}. The restriction is that $(m^*, \vec{x}^*, \mathsf{cred}^*, \mathsf{tok}^*) \notin \mathcal{L}_{\mathsf{show}}$, $\mathsf{cred}^* \notin \mathcal{L}_{\mathsf{issue}}$, $(uid^*, \vec{x}^*) \notin \mathcal{L}_{\mathsf{corrupt}}$. We say that $\mathcal{A}_1$ wins the game if $\mathcal{FAC}.\mathsf{Verify}$ $(\mathsf{tok}^*, m^*) = 1$.

One can note that this game is perfectly simulated if the guess on $\beta^* \in [1, q]$ is correct, which occurs with probability $1/q$. In this case, adversary $\mathcal{A}_1$ is succeed with advantage ϵ_1 to prove knowledge of $\mathsf{usk}_{\beta^*} = a$ when it shows a valid credential. $\mathcal{S}$ sends the challenge tuple to $\mathcal{C}$. Then, $\mathcal{C}$ runs the extractor of the proof of knowledge to recover a, which is a solution to the DL problem. Therefore, the probability for $\mathcal{C}$ to break the DL assumption is ϵ_1/q. □

Proposition A.2 Suppose type-2 adversary $\mathcal{A}_2$ is able to break the unforgeability of FAC with advantage ϵ_2. Then, we can utilize $\mathcal{A}_2$ to break the existential unforgeability of URS in [36] with advantage ϵ_2.

Proof. The proof reduces the unforgeability of FAC to the existential unforgeability of the unlinkable redactable signature (URS) scheme in Section 4 of [36]. As the unforgeability of URS relies on the DL assumption, this proposition follows. Let type-2 adversary $\mathcal{A}_2$ be a PPT adversary that wins the unforgeability game with probability ϵ_2. Consider a simulator $\mathcal{S}$ which runs $\mathcal{A}_2$ as a subroutine and interacts with a unforgeability game challenger $\mathcal{C}$ for the URS scheme in [36].

Setup. $\mathcal{S}$ generates the public parameter $\mathsf{pp} = (g, h, n)$ and sends it to $\mathcal{C}$, where g, h are generators of G_1, G_2, respectively, and n is the attribute number. $\mathcal{C}$ generates the public key $\widetilde{\mathsf{pk}} = (W, \{X_i, Y_i\}_{i\in[1,n]}, \{Z_{i,j}\}_{1\le i\neq j\le n})$ of URS, and transmits it to simulator $\mathcal{S}$, where $W = g^{\tau}$, $X_i = h^{y_i}$, $Y_i = g^{y_i}$ for $i \in [1, n]$, and $Z_{i,j} = g^{y_i\cdot y_j}$ for $1 \le i \neq j \le n$. Note that the secret key $\widetilde{\mathsf{sk}} = (\tau, \{y_i\}_{i\in[1,n]})$ of URS is unknown to $\mathcal{S}$. $\mathcal{S}$ selects random elements $y_0, y_{n+1} \xleftarrow{\$} \mathbb{Z}_p^*$, and implicitly sets the secret key of FAC as $\mathsf{sk} = (\tau, \{y_i\}_{i\in[0,n+1]})$. $\mathcal{S}$ calculates $X_i = h^{y_i}$, $Y_i = g^{y_i}$ for $i = \{0, n+1\}$, computes $Z_{i,n+1} = Y_i^{y_{n+1}} = g^{y_i\cdot y_{n+1}}$ for $1 \le i \le n$, $Z_{0,j} = Y_j^{y_0} = g^{y_0\cdot y_j}$ for $1 \le j \le n$, and $Z_{0,n+1} = g^{y_0\cdot y_{n+1}}$. $\mathcal{S}$ sets the public key of FAC as $\mathsf{pk} = (W, \{X_i, Y_i\}_{i\in[0,n+1]}, \{Z_{i,j}\}_{0\le i\neq j\le n+1})$ and forwards it to $\mathcal{A}_2$.

Query. The adversary $\mathcal{A}_2$ adaptively makes the following queries.

– For the UserKeyGen queries on $(uid, \vec{x})$, $\mathcal{S}$ generates user's secret key $\mathsf{usk} \xleftarrow{\$} \mathbb{Z}_p^*$ and public key $\mathsf{upk} = h^{\mathsf{usk}}$, which are returned to $\mathcal{A}_2$. $\mathcal{S}$ adds $(uid, \vec{x}, \mathsf{upk}, \mathsf{usk})$ to a list $\mathcal{L}_{\mathsf{honest}}$. If $(uid, \vec{x})$ already exists in $\mathcal{L}_{\mathsf{honest}}$, $\mathcal{S}$ just replies with the same answer.

– For the Issue queries on $(uid, \vec{x})$, $\mathcal{S}$ runs UserKeyGen to generate $(\mathsf{upk}, \mathsf{usk})$ if $(uid, \vec{x})$ has not been queried beforehand. Otherwise, $\mathcal{S}$ extracts user's keys $(\mathsf{upk}, \mathsf{usk})$ from $\mathcal{L}_{\mathsf{honest}}$. Then, $\mathcal{S}$ queries $\mathcal{C}$ on $(uid, \vec{x})$ and obtains a URS signature $\sigma = (\widetilde{\sigma}_1, \widetilde{\sigma}_2)$, where $\widetilde{\sigma}_1 \xleftarrow{\$} G_2$ and $\widetilde{\sigma}_2 \leftarrow \widetilde{\sigma}_1^{\tau+\sum_{i=1}^{n} y_i\cdot x_i}$. $\mathcal{S}$ implicitly sets $\sigma_1 = \widetilde{\sigma}_1 = h^r$, computes

$$\begin{aligned}\sigma_2 &= (\widetilde{\sigma}_1)^{\mathsf{usk}\cdot y_0}(\widetilde{\sigma}_1)^{y_{n+1}\cdot uid}\widetilde{\sigma}_2\\ &= (h^{\mathsf{usk}})^{r\cdot y_0} h^{r(\tau+\sum_{i=1}^{n} y_i x_i + y_{n+1}\cdot uid)}\\ &= \mathsf{upk}^{r\cdot y_0}\cdot h^{r(\tau+\sum_{i=1}^{n} y_i x_i + y_{n+1}\cdot uid)},\end{aligned}$$

and returns $\mathsf{cred} \leftarrow (\sigma_1, \sigma_2)$ to $\mathcal{A}_2$. $\mathcal{S}$ inserts $(uid, \vec{x}, \mathsf{upk}, \mathsf{cred})$ into a list $\mathcal{L}_{\mathsf{issue}}$.

– The inputs of Show query are $(uid, \{x_i\}_{i\in\mathcal{I}}, \mathsf{cred}, m)$. A Show query can only be made for a credential that has been created in the Issue query since the latter uses the $\mathcal{O}_{Sign^*}$ oracle of the unforgeability game of URS scheme [36] as subroutine. Then, $\mathcal{S}$ answers the query by executing the Show algorithm and returns the token tok to $\mathcal{A}_2$. $\mathcal{S}$ inserts $(uid, \{x_i\}_{i\in\mathcal{I}}, \mathsf{cred}, m, \mathsf{tok})$ into a list $\mathcal{L}_{\mathsf{show}}$.

– To answer Corrupt on $(uid, \vec{x})$, simulator $\mathcal{S}$ returns the corresponding user secret key usk, credential cred and the token tok to $\mathcal{A}_2$, which are recorded in $\mathcal{L}_{\mathsf{honest}}$, $\mathcal{L}_{\mathsf{issue}}$ and $\mathcal{L}_{\mathsf{show}}$, respectively. $\mathcal{S}$ inserts $(uid, \vec{x})$ into $\mathcal{L}_{\mathsf{corrupt}}$. If $(uid, \vec{x})$ does exist in these lists, $\mathcal{S}$ returns $\perp$.

Challenge. Adversary $\mathcal{A}_2$ outputs a challenge tuple $(uid^*, \vec{x}^*, m^*, \mathsf{cred}^*, \mathsf{tok}^*)$, which associates with challenge secret key usk^*. Since we are simulating a type-2 adversary $\mathcal{A}_2$, it is requested that usk^* should be different from usk_i for any honest user i. The constraints also include that $(m^*, \vec{x}^*, \mathsf{cred}^*, \mathsf{tok}^*) \notin \mathcal{L}_{\mathsf{show}}$, $\mathsf{cred}^* \notin \mathcal{L}_{\mathsf{issue}}$, $(uid^*, \vec{x}^*) \notin \mathcal{L}_{\mathsf{corrupt}}$. We say that $\mathcal{A}_2$ wins the game if $\mathcal{FAC}.\mathsf{Verify}$ $(\mathsf{tok}^*, m^*) = 1$.

If $\mathcal{A}_2$ is succeed with advantage ϵ_2 to prove knowledge of usk^* when it shows a valid credential. $\mathcal{S}$ sends the challenge tuple to $\mathcal{C}$. Then, $\mathcal{C}$ runs the extractor of proof of knowledge to recover usk^*. $\mathcal{C}$ parses $\mathsf{cred}^* \leftarrow (\sigma_1^*, \sigma_2^*)$ and calculates $\widetilde{\sigma}_1^* = \sigma_1^* = h^r$, computes $\widetilde{\sigma}_2^* = \sigma_2^*(\sigma_1^*)^{-\mathsf{usk}^*\cdot y_0}(\sigma_1^*)^{y_{n+1}\cdot uid^*} = (\widetilde{\sigma}_1^*)^{\tau+\sum_{i=1}^{n} y_i\cdot x_i^*}$. Therefore, $\mathcal{C}$ obtains a valid forgery $\sigma^* = (\widetilde{\sigma}_1^*, \widetilde{\sigma}_2^*)$ for the URS scheme in [36] with advantage ϵ_2. □

The proofs for two propositions against $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ conclude the proof for unforgeability of FAC. □

3.2) Anonymity and Unlinkability of FAC

Lemma A.2 The FAC scheme satisfies *anonymity* and *unlinkability* under the decisional Diffie–Hellman (DDH) assumption.

Proof. Suppose a PPT adversary $\mathcal{A}$ is able to break the anonymity of FAC with advantage ϵ. Then, we can utilize $\mathcal{A}$ to solve the DDH problem with advantage ϵ. Consider a simulator $\mathcal{S}$ which runs $\mathcal{A}$ as a subroutine and interacts with a challenger $\mathcal{C}$ for the DDH-assumption. Let g, h be the generators of groups G_1 and G_2, respectively. Let (h, h^a, h^b, h^c) be a challenge tuple of DDH assumption on group G_2. It is required to decide whether $c = a \cdot b$ or $c \xleftarrow{\$} \mathbb{Z}_p^*$.

Setup. $\mathcal{S}$ creates the public key $\mathsf{pk} = (W, \{X_i, Y_i\}_{i \in [0,n+1]}, \{Z_{i,j}\}_{0 \le i \ne j \le n+1})$ following the Setup and CredKeyGen algorithms in FAC and forwards it to $\mathcal{A}$.

Query. $\mathcal{A}$ adaptively makes the following queries.

– For the UserKeyGen queries on $(uid, \vec{x})$, $\mathcal{S}$ generates user's secret key $\mathsf{usk} \xleftarrow{\$} \mathbb{Z}_p^*$ and public key $\mathsf{upk} = h^{\mathsf{usk}}$, which are returned to $\mathcal{A}$. $\mathcal{S}$ adds $(uid, \vec{x}, \mathsf{upk}, \mathsf{usk})$ to a list $\mathcal{L}_{\mathsf{honest}}$. If $(uid, \vec{x})$ already exists in $\mathcal{L}_{\mathsf{honest}}$, $\mathcal{S}$ just replies with the same answer.

– For the Issue queries on $(uid, \vec{x})$, $\mathcal{S}$ runs UserKeyGen to generate $(\mathsf{upk}, \mathsf{usk})$ if $(uid, \vec{x})$ has not been queried beforehand. Otherwise, $\mathcal{S}$ extracts user's keys $(\mathsf{upk}, \mathsf{usk})$ from $\mathcal{L}_{\mathsf{honest}}$. Since $\mathcal{S}$ creates the issuer's secret key sk by itself in Setup phase and knows user's secret key usk, $\mathcal{S}$ executes the Issue protocol to obtain $\mathsf{cred} \leftarrow (\sigma_1, \sigma_2)$, which is returned to $\mathcal{A}$. $\mathcal{S}$ inserts $(uid, \vec{x}, \mathsf{upk}, \mathsf{cred})$ into a list $\mathcal{L}_{\mathsf{issue}}$.

– The inputs of Show query are $(uid, \{x_i\}_{i \in \mathcal{I}}, \mathsf{cred}, m)$. A Show query can only be made for a credential that has been created in the Issue query. $\mathcal{S}$ extracts $(uid, \vec{x}, \mathsf{upk}, \mathsf{cred})$ from $\mathcal{L}_{\mathsf{issue}}$. Then, $\mathcal{S}$ answers the query by executing the Show algorithm and returns the token tok to $\mathcal{A}$. $\mathcal{S}$ inserts $(uid, \{x_i\}_{i \in \mathcal{I}}, \mathsf{cred}, m, \mathsf{tok})$ into a list $\mathcal{L}_{\mathsf{show}}$.

– To answer Corrupt on $(uid, \vec{x})$, simulator $\mathcal{S}$ returns the corresponding user secret key usk, credential cred and the token tok to $\mathcal{A}$, which are recorded in $\mathcal{L}_{\mathsf{honest}}$, $\mathcal{L}_{\mathsf{issue}}$ and $\mathcal{L}_{\mathsf{show}}$, respectively. $\mathcal{S}$ inserts $(uid, \vec{x})$ into $\mathcal{L}_{\mathsf{corrupt}}$. If $(uid, \vec{x})$ does exist in these lists, $\mathcal{S}$ returns $\perp$.

Challenge. In this phase, $\mathcal{A}$ outputs two challenge users with attributes $(uid_0^*, \vec{x}_0^*)$, $(uid_1^*, \vec{x}_1^*)$, and message m^*. The simulator $\mathcal{S}$ flips a random coin $\bar{b} \in \{0,1\}$ and generates challenge tok^* for user $uid_{\bar{b}}^*$, where $\mathsf{tok}^* \leftarrow (\{x_i^*\}_{i \in \mathcal{I}^*}, T_1^*, T_2^*, \bar{\sigma}_1^*, \bar{\sigma}_2^*, \pi_2^*)$, $T_1^* = g^\alpha$, $T_2^* = (T_1^*)^{\sum_{i \in \mathcal{I}^{*\prime}} y_i}$, $\bar{\sigma}_1^* = h^b$, $\bar{\sigma}_2^* = (h^c)^{y_0} \cdot (\bar{\sigma}_1^*)^{\alpha + \tau + \sum_{i \in \mathcal{I}^*} y_i x_i^* + y_{n+1} \cdot uid_{\bar{b}}^*})$, π_2^* is a simulated knowledge of a, the disclosed attribute set $\mathcal{I}^* \subseteq [1, n]$, $\alpha \in_R \mathbb{Z}_p^*$. The restriction is that the disclosed attributes in $\vec{x}_0^*$ and $\vec{x}_1^*$ are the same, $\mathsf{cred}_{0/1}^* \notin \mathcal{L}_{\mathsf{issue}}$, $(m^*, \vec{x}_{0/1}^*, \mathsf{tok}_{0/1}^*) \notin \mathcal{L}_{\mathsf{show}}$, $(uid_{0/1}^*, \vec{x}_{0/1}^*) \notin \mathcal{L}_{\mathsf{corrupt}}$.

Guess. The adversary $\mathcal{A}$ makes a guess $\bar{b}' \in \{0,1\}$ on the identity of the user from (uid_0^*, uid_1^*). $\mathcal{A}$ wins the game if $\bar{b}' = \bar{b}$.

$\mathcal{S}$ sends the guess result of $\mathcal{A}$ to $\mathcal{C}$. It is noted that if $c = ab$, by setting $t_1 = \alpha - \sum_{i \in [1,n] \setminus \mathcal{I}^*} y_i x_i^*$, One can see that tok^* is distributed as in the FAC scheme. Else, c is a random number in $\mathbb{Z}_p^*$ and $\bar{\sigma}_2^*$ is a random element in $\mathbb{G}_2$. Since $(T_1^*, T_2^*, \bar{\sigma}_1^*)$ are independent of a and $\{x_i^*\}_{i \in [n] \setminus \mathcal{I}^*}$, $\mathcal{A}$ cannot succeed in this game with non-negligile advantage. If $\mathcal{A}$ is able to win the security game with advantage ϵ, $\mathcal{C}$ can makes use of $\mathcal{A}$ to solve the DDH problem with advantage ϵ. $\square$

B. ACME: Security Model and Proof

(1) Security Model

This section defines *correctness*, *privacy*, *authenticity*, *anonymity* and *unlinkability* for ACME scheme.

Definition B.1. Let $\mathcal{D}$ be the universe of user identity, and Ω be the universe of attributes. An anonymous credential-based matchmaking encryption encryption scheme $\mathcal{ACME}$ is *correct* for $\mathcal{D}, \Omega$ if all $uid \in \mathcal{D}$, all $\vec{x} \subseteq \Omega$ for all security parameter λ,

$$\Pr\left[\begin{array}{l} (\mathsf{mpk}, \mathsf{msk}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda, 1^n); \\ (\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{CredKeyGen}(\mathsf{mpk}); \\ (\mathsf{upk}, \mathsf{usk}) \xleftarrow{\$} \mathsf{UserKeyGen}(\mathsf{mpk}); \\ \mathsf{cred}_{\mathsf{snd}} \xleftarrow{\$} \langle \mathsf{Issue.I}(\mathsf{sk}, \mathsf{upk}) \rightleftarrows \mathsf{Issue.U}(uid, \vec{x}_{\mathsf{snd}}, \mathsf{usk}) \rangle \,; \\ \mathsf{DK}_{\vec{x}_{\mathsf{rcv}}} \xleftarrow{\$} \mathsf{DKGen}(\mathsf{msk}, \vec{x}_{\mathsf{rcv}}); \\ \mathsf{DK}_{f_{\mathsf{rcv}}} \leftarrow \mathsf{PolGen}(\mathsf{msk}, f_{\mathsf{rcv}}); \\ \mathsf{CT}_{\vec{x}_{\mathsf{snd}}, f_{\mathsf{snd}}} \xleftarrow{\$} \mathsf{Enc}(\mathsf{cred}_{\mathsf{snd}}, \vec{x}_{\mathsf{snd}}, f_{\mathsf{snd}}, M) : \\ f_{\mathsf{rcv}}(\vec{x}_{\mathsf{snd}}^{(out)}) = 1 \wedge f_{\mathsf{snd}}(\vec{x}_{\mathsf{rcv}}^{(out)}) = 1 \wedge \\ \mathsf{Dec}(\mathsf{DK}_{\vec{x}_{\mathsf{rcv}}}, \mathsf{DK}_{f_{\mathsf{rcv}}}, \mathsf{CT}_{\vec{x}_{\mathsf{snd}}, f_{\mathsf{snd}}}) = \perp \end{array}\right]$$

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

Definition B.2. An ACME scheme $\mathcal{ACME}$ satisfies *privacy* if for any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a negligible function ν such that $\mathsf{Adv}_{\mathcal{ACME}}^{\mathsf{priv}}(\lambda) \overset{\mathsf{def}}{=}$

$$\Pr\left[b' = b \;\middle|\; \begin{array}{l} (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n), b \xleftarrow{\$} \{0,1\} \\ (M_0^*, M_1^*, \mathsf{cred}_{\mathsf{snd}_0}^*, \mathsf{cred}_{\mathsf{snd}_1}^*, \vec{x}_{\mathsf{snd}_0}^*, \vec{x}_{\mathsf{snd}_1}^*, f_{\mathsf{snd}}^*) \\ \qquad\qquad \leftarrow \mathcal{A}_1^{\mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_3}(\mathsf{mpk}) \\ \mathsf{CT}^* \leftarrow \mathsf{Enc}(\mathsf{cred}_{\mathsf{snd}_b}^*, \vec{x}_{\mathsf{snd}_b}^*, f_{\mathsf{snd}}^*, M_b) \\ b' \leftarrow \mathcal{A}_2^{\mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_3}(\mathsf{mpk}, \mathsf{CT}^*) \end{array}\right]$$

$\le \nu(\lambda)$, where oracles $\mathcal{O}_1$, $\mathcal{O}_2$, $\mathcal{O}_3$ are implemented by Issue $(\mathsf{msk}, \cdot)$, $\mathsf{DKGen}(\mathsf{msk}, \cdot)$, $\mathsf{PolGen}(\mathsf{msk}, \cdot)$, respectively. It is required that $\mathcal{O}_2$ and $\mathcal{O}_3$ are not queried for attributes and policies that can satisfy $(\vec{x}_{\mathsf{snd}_0}^*, f_{\mathsf{snd}}^*)$ or $(\vec{x}_{\mathsf{snd}_1}^*, f_{\mathsf{snd}}^*)$. It is also required that for public attributes $\vec{x}_{\mathsf{snd}_0}^{(out)*}$/ $\vec{x}_{\mathsf{snd}_1}^{(out)*}$ in $\vec{x}_{\mathsf{snd}_0}^*$ / $\vec{x}_{\mathsf{snd}_1}^*$, we have $\vec{x}_{\mathsf{snd}_0}^{(out)*} = \vec{x}_{\mathsf{snd}_1}^{(out)*}$.

This model only captures security under chosen plaintext attacks (CPA). We can extend the above definition by introducing another decryption oracle $\mathcal{O}_4$ which can decrypt ciphertexts except the challenge ciphertext CT^* to capture security under chosen-ciphertext attacks (CCA).

Definition B.3. An ACME scheme $\mathcal{ACME}$ satisfies *authenticity* if for any PPT adversary $\mathcal{A}$, there exists a negligible function ν such that $\mathsf{Adv}_{\mathcal{ACME}}^{\mathsf{auth}}(\lambda) \overset{\mathsf{def}}{=}$

$$\Pr\left[\begin{array}{l} (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n) \\ (\mathsf{CT}_{\vec{x}_{\mathsf{snd}}, f_{\mathsf{snd}}}, \vec{x}_{\mathsf{rcv}}, f_{\mathsf{rcv}}) \leftarrow \mathcal{A}^{\mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_3}(\mathsf{mpk}) \\ \mathsf{DK}_{\vec{x}_{\mathsf{rcv}}} \leftarrow \mathsf{DKGen}(\mathsf{msk}, \vec{x}_{\mathsf{rcv}}) \\ \mathsf{DK}_{f_{\mathsf{rcv}}} \leftarrow \mathsf{PolGen}(\mathsf{msk}, f_{\mathsf{rcv}}) \\ M = \mathsf{Dec}(\mathsf{DK}_{\vec{x}_{\mathsf{rcv}}}, \mathsf{DK}_{f_{\mathsf{rcv}}}, \mathsf{CT}_{\vec{x}_{\mathsf{snd}}, f_{\mathsf{snd}}}) \\ \forall \vec{x} \in \mathcal{Q}_{\mathcal{O}_1, \mathcal{O}_2} : (f_{\mathsf{rcv}}(\vec{x}^{(out)}) = 0) \wedge (M \ne \perp) \end{array}\right]$$

$\le \nu(\lambda)$, where oracles $\mathcal{O}_1$, $\mathcal{O}_2$, $\mathcal{O}_3$ are implemented by Issue $(\mathsf{msk}, \cdot)$, $\mathsf{DKGen}(\mathsf{msk}, \cdot)$, $\mathsf{PolGen}(\mathsf{msk}, \cdot)$.

ACME also satisfy the security properties of anonymous credential, namely *anonymity* (Def. A.3) and *unlinkability* (Def. A.4), against an entity who can decrypt the ciphertext.

(2) Security Proof

Theorem 6.2. The proposed ACME scheme achieves privacy, authenticity, anonymity and unlinkability if the $MDDH_k$ assumption holds and the underlying FAC is secure.

The security proofs of authenticity, anonymity and unlinkability of ACME follows those of unforgeability, anonymity and unlinkability of FAC. Next, we prove that the proposed ACME scheme achieves privacy under the $MDDH_k$ assumption.

Proof Intuition. In our proposed ACME scheme, the message M and the FAC token $\mathsf{tok}_{\mathsf{snd}}$ (corresponding to the private attributes) are encrypted using a symmetric key K which is encapsulated in $\mathsf{ct}_0 = e([\widetilde{\mathbf{s}}^\top \mathbf{A} + \mathbf{s}^\top \mathbf{A}]_1, [\mathbf{v}]_2) \cdot K$. Hence, the privacy of both M and $\mathsf{tok}_{\mathsf{snd}}$ is based on the confidentiality of the symmetric key K.

As shown in the correctness, when $f_{\mathsf{rcv}}(\vec{x}^{(out)}_{\mathsf{snd}}) = 1$ (corresponding to KP-ABE), we have

$$\frac{e(\mathsf{ct}'_2, \prod_{j \in \mathcal{S}_r} \mathsf{dk}_j^{\omega_j})}{e(\mathsf{ct}'_1, \prod_{j \in \mathcal{S}_r} (\prod_{i:x^{(out)}_{s,i}=1} \mathsf{dk}_{i,j})^{\omega_j})} = ([\widetilde{\mathbf{s}}^\top \mathbf{A}\mathbf{v}]_T)^{-1}$$

and when $f_{\mathsf{snd}}(\vec{x}^{(out)}_{\mathsf{rcv}}) = 1$ (corresponding to CP-ABE), we have

$$\frac{e(\prod_{j \in \mathcal{S}_s} (\prod_{i:x^{(out)}_{r,i}=1} \mathsf{ct}_{i,j})^{\mu_j}, \mathsf{dk}_2)}{e(\mathsf{ct}_1, \mathsf{dk}_1) \cdot e(\prod_{j \in \mathcal{S}_s} \widetilde{\mathsf{ct}}_j^{\mu_j}, \mathsf{dk}_3)} = ([\mathbf{s}^\top \mathbf{A}\mathbf{v}]_T)^{-1}.$$

Hence, the proof for the confidentiality of K essentially follows the same proof techniques used in the underlying (dual) CP-ABE and KP-ABE schemes [26], [27]. Specifically, if $f_{\mathsf{rcv}}(\vec{x}^{(out)*}_{\mathsf{snd}}) \neq 1$, then the confidentiality of K is ensured by the security of the KP-ABE scheme; otherwise, since according to the security game, the adversary is not allowed to obtain keys for f_{rcv} and $\vec{x}_{\mathsf{rcv}}$ such that $f_{\mathsf{rcv}}(\vec{x}^{(out)*}_{\mathsf{snd}}) = 1 \wedge f^*_{\mathsf{snd}}(\vec{x}^{(out)}_{\mathsf{rcv}}) = 1$, the security is ensured by that of the CP-ABE scheme.

The dual ABE schemes in [26], [27] both applied a sequence of games and a hybrid argument in the security proofs. The initial game is the same as the original security game whereas in the last game, the encrypted symmetric key K is replaced by a random key. Here we follow the same game sequences defined in [26], [27] by considering two cases: if the adversary would not query a key for f_{rcv} such that $f_{\mathsf{rcv}}(\vec{x}^{(out)*}_{\mathsf{snd}}) = 1$, then we follow the transitions of the keys and ciphertexts in the proof of KP-ABE and use normal keys and ciphertexts for the CP-ABE components; otherwise, we perfom the transitions in the opposite way. Below we outline the crucial steps of the proof.

Proof. Let $\equiv$ denote that two distributions are identically distributed, and $\approx_c$ represent that two distributions are computationally indistinguishable.

The security of ACME scheme is proved by a series of hybrid games, depending on whether the adversary would query a key for f_{rcv} such that $f_{\mathsf{rcv}}(\vec{x}^{(out)*}_{\mathsf{snd}}) = 1$.

Case 1: the adversary queries a key for f_{rcv} such that $f_{\mathsf{rcv}}(\vec{x}^{(out)*}_{\mathsf{snd}}) = 1$. Note that in this case the adversary cannot query a key for $\vec{x}_{\mathsf{rcv}}$ such that $f^*_{\mathsf{snd}}(\vec{x}^{(out)}_{\mathsf{rcv}}) = 1$.

A ciphertext (under access policy f and attributes $\vec{x}$) can be in one of the following forms:

- Normal: A normal ciphertext is generated by Enc.

- SF: An SF ciphertext is the same as Normal ciphertext, except that $\mathbf{s}^\top \mathbf{A}, \mathbf{s}_j^\top \mathbf{A}$ are replaced with $\mathbf{c}^\top$, $\mathbf{c}_j^\top$, where $\mathbf{c}, \mathbf{c}_j \leftarrow \mathbb{Z}_p^{2k}$. Let $\widetilde{\mathbf{c}} := \widetilde{\mathbf{s}}^\top \mathbf{A}$ as in the normal ciphertext, then $\mathsf{CT}_{\vec{x},f} :=$

$$(\mathsf{ct}_0 = e([\widetilde{\mathbf{c}}^\top + \boxed{\mathbf{c}^\top}]_1, [\mathbf{v}]_2) \cdot K,$$

$$\mathsf{ct}'_1 = [\widetilde{\mathbf{c}}^\top]_1, \mathsf{ct}'_2 = [\widetilde{\mathbf{c}}^\top \sum\nolimits_{i:x_i^{(out)}=1} \mathbf{W}_i]_1,$$

$$\mathsf{ct}_1 = [\boxed{\mathbf{c}^\top}]_1, \{\widetilde{\mathsf{ct}}_j = [\boxed{\mathbf{c}_j^\top}]_1, \mathsf{ct}_{\rho(j),j} = [\mathbf{u}_j^\top + \boxed{\mathbf{c}_j^\top} \mathbf{W}_{\rho(j)}]_1,$$

$$\mathsf{ct}_{i,j} = [\boxed{\mathbf{c}_j^\top} \mathbf{W}_i]_1\}).$$

A secret key DK_f (for policy f) follows its normal form in this case. A secret key (for attributes $\vec{x}$) can be in one of the following forms:

- Normal: A normal secret key is generated by DKGen.

- SF: An SF key is sampled as a Normal key, except $\mathbf{v}$ is replaced with $\mathbf{v} + \mathbf{A}^\perp \boldsymbol{\delta}^{(q)}$, where a fresh $\boldsymbol{\delta}^{(q)} \leftarrow \mathbb{Z}_p^k$ is chosen per SF key and $\mathbf{A}^\perp$ is any fixed $\mathbf{A}^\perp \in \mathbb{Z}_p^{2k \times k} \backslash \{\mathbf{0}\}$ such that $\mathbf{A}\mathbf{A}^\perp = \mathbf{0}$. That is $\mathsf{DK}_{\vec{x}} :=$

$$\begin{aligned}(\mathsf{dk}_1 &= [\boxed{\mathbf{v} + \mathbf{A}^\perp \boldsymbol{\delta}^{(q)}} + \mathbf{U}_0 \mathbf{B}\mathbf{r}]_2, \mathsf{dk}_2 = [\mathbf{B}\mathbf{r}]_2,\\ \mathsf{dk}_3 &= [\sum\nolimits_{i:x_i^{(out)}=1} \mathbf{W}_i \mathbf{B}\mathbf{r}]_2).\end{aligned}$$

- P-Normal: A P-Normal key as the same as a Normal key, except $\mathbf{B}\mathbf{r}$ is replaced with $\mathbf{d} \leftarrow \mathbb{Z}_p^k$. That is $\mathsf{DK}_{\vec{x}} :=$

$$(\mathsf{dk}_1 = [\mathbf{v} + \mathbf{U}_0 \boxed{\mathbf{d}}]_2, \mathsf{dk}_2 = [\boxed{\mathbf{d}}]_2, \mathsf{dk}_3 = [\sum\nolimits_{i:x_i^{(out)}=1} \mathbf{W}_i \boxed{\mathbf{d}}]_2).$$

- P-SF: A P-SF key is the same as an SF key, except $\mathbf{B}\mathbf{r}$ is replaced with $\mathbf{d} \leftarrow \mathbb{Z}_p^k$. That is $\mathsf{DK}_{\vec{x}} :=$

$$(\mathsf{dk}_1 = [\boxed{\mathbf{v} + \mathbf{A}^\perp \boldsymbol{\delta}^{(q)}} + \mathbf{U}_0 \boxed{\mathbf{d}}]_2, \mathsf{dk}_2 = [\boxed{\mathbf{d}}]_2,$$

$$\mathsf{dk}_3 = [\sum\nolimits_{i:x_i^{(out)}=1} \mathbf{W}_i \boxed{\mathbf{d}}]_2).$$

Next, we define the hybrid sequence for the proof. Assume the adversary $\mathcal{A}$ makes at most Q_x attribute decryption key queries.

- H_0: This is the real game where all secret keys and ciphertexts are Normal.

- H_1: This game is the same as H_0 except that the challenge ciphertext is SF.

- $\mathsf{H}_{2,\ell,1}$: This game is the same as H_1 except that the ℓ-th attribute decryption key is P-Normal, the first $\ell - 1$ attribute decryption keys are SF and the last $Q_x - \ell$ attribute decryption keys are Normal, where $\ell = 0, \cdots, Q_x$.

- $\mathsf{H}_{2,\ell,2}$: This game is the same as $\mathsf{H}_{2,\ell,1}$ except the ℓ'th attribute decryption key is P-SF, where $\ell = 0, \cdots, Q_x$.

- $\mathsf{H}_{2,\ell,3}$: This game is the same as $\mathsf{H}_{2,\ell,2}$ except the ℓ'th attribute decryption key is SF, where $\ell = 0, \cdots, Q_x$.

- H_3: This game is the same as $\mathsf{H}_{2,Q_x,3}$ except that the message encryption symmetric key K to be encrypted is replaced by a random $\widetilde{K}$.

Let $\mathcal{A}$ be a PPT adversary, and $\mathsf{Adv}_{\mathsf{xxx}}$ be the advantage of $\mathcal{A}$ in game $\mathsf{H}_{\mathsf{xxx}}$. Also, define $\mathsf{H}_1 \equiv \mathsf{H}_{2,0,1}$. To complete the proof for Case 1, we prove lemmas E.2-E.6 in the following.

Lemma B.1. Under the $MDDH_k^{2m+1}$ assumption on G_1, we have

$$|Pr[\langle \mathcal{A}, \mathsf{H}_0\rangle = 1] - Pr[\langle \mathcal{A}, \mathsf{H}_1\rangle = 1]| = \mathsf{negl}(\lambda).$$

Proof. Assume that $\mathcal{A}$ distinguishes H_0 and H_1 with non-negligible advantage. Then, we can construct another adversary $\mathcal{B}$ to break the $MDDH_k^{2m+1}$ assumption. On input a $MDDH_k^{2m+1}$ challenge $([\mathbf{A}]_1, [\mathbf{Z}]_1)$, where either $\mathbf{Z}^\top = \mathbf{S}^\top\mathbf{A}$ or $\mathbf{Z} = \mathbf{C}$, for $\mathbf{S}, \mathbf{C} \leftarrow \mathbb{Z}_p^{(2m+1)\times k}$. $\mathcal{B}$ proceeds as below.

Setup. $\mathcal{B}$ chooses generators $g \leftarrow G_1$, $h \leftarrow G_2$, user's attribute number n, and sets $\mathsf{pp} = (g, h, n)$. $\mathcal{B}$ runs $\mathcal{FAC}.\mathsf{CredKeyGen}$ to create $(\mathsf{pk}, \mathsf{sk})$. $\mathcal{B}$ selects $\mathbf{B} \leftarrow \mathbb{Z}_p^{k\times k}$, $\mathbf{U}_0, \mathbf{W}_i \leftarrow \mathbb{Z}_p^{2k\times k}$, $\mathbf{v} \leftarrow \mathbb{Z}_p^{2k}$ and sets mpk, msk as in the scheme, where the elements $[\mathbf{AW}_i]_1$ in mpk can be derived from $[\mathbf{A}]_1$ and $\mathbf{W}_i$.

Issue Query. $\mathcal{B}$ firstly executes $\mathcal{FAC}.\mathsf{UserKeyGen}$ to create user's public/secret keys $\mathsf{upk}/\mathsf{usk}$. $\mathcal{B}$ can response to any credential issue query since credential issuer's secret key sk is generated by $\mathcal{B}$.

Attribute Decryption Key Query. $\mathcal{B}$ can response to any attribute decryption key query since msk is generated by $\mathcal{B}$.

Policy Decryption Key Query. $\mathcal{B}$ can response to any policy decryption key query since msk is generated by $\mathcal{B}$.

Challenge. After the secret key queries, $\mathcal{A}$ requests for the challenge ciphertext corresponding to symmetric keys (K_0, K_1), attributes $\vec{x}$ and formula f. $\mathcal{B}$ flips a random coin $\mathsf{coin} \leftarrow \{0,1\}$ and constructs the challenge ciphertext for K_{coin}. $\mathcal{B}$ computes $(\{\mathbf{u}_j^\top\}, \rho) \leftarrow \mathsf{share}(f, \mathbf{z}_{2m+1}^\top\mathbf{U}_0)$ and sets the challenge ciphertext as $\mathsf{CT}_{\vec{x},f} :=$

$$\big(\mathsf{ct}_0 = e([\widetilde{\mathbf{z}}^\top + \mathbf{z}_{2m+1}^\top]_1, [\mathbf{v}]_2)\cdot K_{\mathsf{coin}},$$
$$\mathsf{ct}'_1 = [\widetilde{\mathbf{z}}^\top]_1, \mathsf{ct}'_2 = [\widetilde{\mathbf{z}}^\top \textstyle\sum_{i:x_i^{(out)}=1}\mathbf{W}_i]_1,$$
$$\mathsf{ct}_1 = [\mathbf{z}_{2m+1}^\top]_1, \{\widetilde{\mathsf{ct}}_j = [\mathbf{z}_j^\top]_1, \mathsf{ct}_{\rho(j),j} = [\mathbf{u}_j^\top + \mathbf{z}_j^\top\mathbf{W}_{\rho(j)}]_1,$$
$$\mathsf{ct}_{i,j} = [\mathbf{z}_j^\top\mathbf{W}_i]_1\}\big),$$

where $(\widetilde{\mathbf{z}}, \mathsf{ct}'_1, \mathsf{ct}'_2)$ are computed normally and note that $|\{\mathbf{u}_j\}| \le 2m$.

Guess. $\mathcal{A}$ halts the game with a guess $\mathsf{coin}' \leftarrow \{0,1\}$. $\mathcal{B}$ outputs 1 if $\mathsf{coin}' = \mathsf{coin}$, and 0 otherwise. It is straight forward to see that if $\mathbf{Z}^\top = \mathbf{S}^\top\mathbf{A}$, the challenge ciphertext is Normal and $\mathcal{B}$ simulates H_0; if $\mathbf{Z}^\top = \mathbf{C}^\top$, the challenge ciphertext is SF and $\mathcal{B}$ simulates H_1. ☐

Lemma B.2. Under the $MDDH_k$ assumption on G_2, we have

$$|Pr[\langle \mathcal{A}, \mathsf{H}_{2,\ell-1,3}\rangle = 1] - Pr[\langle \mathcal{A}, \mathsf{H}_{2,\ell,1}\rangle = 1]| = \mathsf{negl}(\lambda).$$

Proof. Assume that $\mathcal{A}$ distinguishes $\mathsf{H}_{2,\ell-1,3}$ and $\mathsf{H}_{2,\ell,1}$ with non-negligible advantage. Then, we can construct another adversary $\mathcal{B}$ to break the $MDDH_k$ assumption. On input $MDDH_k$ challenge $([\mathbf{B}]_2, [\mathbf{z}]_2)$, where either $\mathbf{z} = \mathbf{Br}$ for $\mathbf{r} \leftarrow \mathbb{Z}_p^k$, or $\mathbf{z} = \mathbf{d}$ for $\mathbf{d} \leftarrow \mathbb{Z}_p^{k+1}$. $\mathcal{B}$ proceeds as below.

Setup. $\mathcal{B}$ chooses generators $g \leftarrow G_1$, $h \leftarrow G_2$, user's attribute number n, and sets $\mathsf{pp} = (g, h, n)$. Next, $\mathcal{B}$ selects $\mathbf{A} \leftarrow \mathbb{Z}_p^{k\times 2k}$, $\mathbf{U}_0, \mathbf{W}_i \leftarrow \mathbb{Z}_p^{2k\times(k+1)}$, $\mathbf{v} \leftarrow \mathbb{Z}_p^{2k}$, and forms mpk with these parameters as in the scheme. $\mathcal{B}$ computes $\mathbf{A}^\perp \in \mathbb{Z}_p^{2k\times k}$ such that $\mathbf{AA}^\perp = \mathbf{0}$, which is used for responding secret key queries. $\mathcal{B}$ runs $\mathcal{FAC}.\mathsf{CredKeyGen}$ to create $(\mathsf{pk}, \mathsf{sk})$.

Issue Query. $\mathcal{B}$ firstly executes $\mathcal{FAC}.\mathsf{UserKeyGen}$ to create user's public/secret keys $\mathsf{upk}/\mathsf{usk}$. $\mathcal{B}$ can response to any credential issue query since credential issuer's secret key sk is generated by $\mathcal{B}$.

Attribute Decryption Key. $\mathcal{B}$ simulates attribute decryption keys as below.

- For the first $\ell - 1$ attribute decryption key queries, say the q-th request is for $\vec{x}$, $\mathcal{B}$ samples $\boldsymbol{\delta}^{(q)}, \mathbf{r}^{(q)} \in \mathbb{Z}_p^k$, and creates (SF) attribute decryption key $\mathsf{DK}_{\vec{x}} :=$

$$\big([\mathbf{v} + \mathbf{A}^\perp\boldsymbol{\delta}^{(q)} + \mathbf{U}_0\mathbf{Br}^{(q)}]_2, [\mathbf{Br}^{(q)}]_2, [\textstyle\sum_{i:x_i=1}\mathbf{W}_i\mathbf{Br}^{(q)}]_2\big).$$

- For the last $Q_x - \ell$ attribute decryption key queries, $\mathcal{B}$ proceeds as before for the first $\ell - 1$ keys except substituting $\mathbf{v} + \mathbf{A}^\perp\boldsymbol{\delta}^{(q)}$ with $\mathbf{v}$. It is obvious that it becomes a Normal key.

- For the ℓth attribute decryption key request, $\mathcal{A}'$ creates the secret key $\mathsf{DK}_{\vec{x}} := \big([\mathbf{v} + \mathbf{U}_0\mathbf{z}]_2, [\mathbf{z}]_2, [\sum_{i:x_i=1}\mathbf{W}_i\mathbf{z}]_2\big)$.

Policy Decryption Key. $\mathcal{B}$ simulates any policy decryption key normally since the elements for generating policy decryption key are generated by $\mathcal{B}$.

Challenge. After the secret key queries, $\mathcal{A}$ requests for the challenge ciphertext corresponding to symmetric keys (K_0, K_1), attributes $\vec{x}$ and formula f. $\mathcal{B}$ flips a random coin $\mathsf{coin} \leftarrow \{0,1\}$ and constructs the challenge ciphertext for K_{coin}. Sample $\mathbf{c}, \mathbf{c}_j \leftarrow \mathbb{Z}_p^{2k}$ for each j, compute $(\{\mathbf{u}_j^\top\}, \rho) \leftarrow \mathsf{share}(f, \mathbf{c}^\top\mathbf{U}_0)$ and return (SF) challenge ciphertext $\mathsf{CT}_{\vec{x},f} :=$

$$\big(\mathsf{ct}_0 = e([\widetilde{\mathbf{z}}^\top + \mathbf{c}^\top]_1, [\mathbf{v}]_2)\cdot K_{\mathsf{coin}},$$
$$\mathsf{ct}'_1 = [\widetilde{\mathbf{z}}^\top]_1, \mathsf{ct}'_2 = [\widetilde{\mathbf{z}}^\top \textstyle\sum_{i:x_i^{(out)}=1}\mathbf{W}_i]_1,$$
$$\mathsf{ct}_1 = [\mathbf{c}^\top]_1, \widetilde{\mathsf{ct}}_j = [\mathbf{c}_j^\top]_1, \mathsf{ct}_{\rho(j),j} = [\mathbf{u}_j^\top + \mathbf{c}_j^\top\mathbf{W}_{\rho(j)}]_1,$$
$$\mathsf{ct}_{i,j} = [\mathbf{c}_j^\top\mathbf{W}_i]_1\big).$$

Guess. $\mathcal{A}$ halts the game with a guess $\mathsf{coin}' \leftarrow \{0,1\}$. $\mathcal{B}$ outputs 1 if $\mathsf{coin}' = \mathsf{coin}$, and 0 otherwise. It is straight forward to see that if $\mathbf{z} = \mathbf{Br}$, then the ℓth attribute decryption key is Normal and $\mathcal{A}'$ has simulated $\mathsf{H}_{2,\ell-1,3}$; if $\mathbf{z} = \mathbf{d}$, then the ℓth attribute decryption key is $\mathsf{P\text{-}Normal}$ and $\mathcal{A}'$ has simulated $\mathsf{H}_{2,\ell,1}$. ☐

Lemma B.3. Under the $MDDH_k$ assumption, we have

$$|Pr[\langle \mathcal{A}, \mathsf{H}_{2,\ell,1}\rangle = 1] - Pr[\langle \mathcal{A}, \mathsf{H}_{2,\ell,2}\rangle = 1]| = \mathsf{negl}(\lambda).$$

Proof. Assume that $\mathcal{A}$ distinguishes $\mathsf{H}_{2,\ell,1}$ and $\mathsf{H}_{2,\ell,2}$ with non-negligible advantage. Then, we can construct another adversary $\mathcal{B}$ that distinguishes the oracles in $\mathsf{G}_\beta^{1-\mathsf{ABE}}$ of [26], which implies an attacker against the $MDDH_k$ assumption.

Given $\mu^{(0)}$ as an input and equipped with oracles $\mathcal{O}_{\mathsf{F},\beta}$, $\mathcal{O}_X$ and $\mathcal{O}_E$ (defined in $\mathsf{G}_\beta^{1-\mathsf{ABE}}$ of [26]), $\mathcal{B}$ proceeds as below.

Setup. $\mathcal{B}$ chooses generators $g \leftarrow G_1$, $h \leftarrow G_2$, user's attribute number n, and sets $\mathsf{pp} = (g, h, n)$. Next, $\mathcal{B}$ chooses $\mathbf{A} \leftarrow \mathbb{Z}_p^{k\times 2k}$, $\mathbf{B} \leftarrow \mathbb{Z}_p^{k\times k}$, $\widetilde{\mathbf{U}}_0, \widetilde{\mathbf{W}}_i \leftarrow \mathbb{Z}_p^{2k\times k}$ for $i \in [1, n]$, and $\widetilde{\mathbf{v}} \leftarrow \mathbb{Z}_p^{2k}$, computes $\mathbf{A}^\perp \in \mathbb{Z}_p^{2k\times k}\backslash\{\mathbf{0}\}$, $\mathbf{b}^\perp \leftarrow \mathbb{Z}_p^k$ such that $\mathbf{A}\mathbf{A}^\perp = \mathbf{0}$ and $(\mathbf{b}^\perp)^\top\mathbf{B} = \mathbf{0}$ and implicitly defines

$$\mathbf{v} := \widetilde{\mathbf{v}} - \frac{\mu^{(0)}((\mathbf{b}^\perp)^\top\mathbf{d})}{\mathbf{c}^\top\mathbf{A}^\perp\mathbf{u}}\mathbf{A}^\perp\mathbf{u},$$

$$\mathbf{U}_0 := \widetilde{\mathbf{U}}_0 + \frac{\mu^{(\beta)}}{\mathbf{c}^\top\mathbf{A}^\perp\mathbf{u}}\mathbf{A}^\perp\mathbf{u}(\mathbf{b}^\perp)^\top,$$

$$\mathbf{W}_i := \widetilde{\mathbf{W}}_i + \mathbf{A}^\perp\mathbf{w}_i(\mathbf{b}^\perp)^\top,$$

where $\mathbf{w}_i \in \mathbb{Z}_p^k$, $\mu^{(\beta)} \in \mathbb{Z}_p$ are chosen by the $\mathsf{G}_\beta^{\text{1-ABE}}$ game in [26], $\mathbf{c} \leftarrow \mathbb{Z}_p^{2k}$ is selected for generating the challenge ciphertext, $\mathbf{d} \leftarrow \mathbb{Z}_p^{k+1}$ is selected for generating the ℓth secret key, and $\mathbf{u} \leftarrow \mathbb{Z}_p^k$. Note that $\mathcal{B}$ can compute $\mathbf{v}$ since it has $\mu^{(0)}$ from the game and knows all other vectors. Then, $\mathcal{B}$ creates

$$\mathsf{mpk} := (\mathbf{pp}, [\mathbf{A}]_1, [\mathbf{A}\widetilde{\mathbf{U}}_0]_1, [\mathbf{A}\widetilde{\mathbf{W}}_1]_1, \cdots, [\mathbf{A}\widetilde{\mathbf{W}}_n]_1, e([\mathbf{A}]_1, [\widetilde{\mathbf{v}}]_2)).$$

$\mathcal{B}$ runs $\mathcal{FAC}.\mathsf{CredKeyGen}$ to create $(\mathsf{pk}, \mathsf{sk})$.

Issue Query. $\mathcal{B}$ firstly executes $\mathcal{FAC}.\mathsf{UserKeyGen}$ to create user's public/secret keys $\mathsf{upk}/\mathsf{usk}$. $\mathcal{B}$ can response to any credential issue query since credential issuer's secret key sk is generated by $\mathcal{B}$.

Attribute Decryption Key. $\mathcal{B}$ simulates attribute decryption key as below.

- For the first $\ell - 1$ attribute decryption key queries, say the qth request is for $\vec{x}$, $\mathcal{B}$ samples $\boldsymbol{\delta}^{(q)}, \mathbf{r}^{(q)} \leftarrow \mathbb{Z}_p^k$, and creates the (SF) attribute decryption key: $\mathsf{DK}_{\vec{x}} :=$

$$(\mathsf{dk}_1 = [\mathbf{v} + \mathbf{A}^\perp\boldsymbol{\delta}^{(q)} + \underbrace{\widetilde{\mathbf{U}}_0\mathbf{B}\mathbf{r}^{(q)}}_{=\mathbf{U}_0\mathbf{B}\mathbf{r}^{(q)}}]_2,$$

$$\mathsf{dk}_2 = [\mathbf{B}\mathbf{r}^{(q)}]_2, \quad \mathsf{dk}_3 = [\textstyle\sum_{i:x_i^{(out)}=1} \underbrace{\widetilde{\mathbf{W}}_i\mathbf{B}\mathbf{r}^{(q)}}_{=\mathbf{W}_i\mathbf{B}\mathbf{r}^{(q)}}]_2).$$

- For the last $Q_x - \ell$ attribute decryption key queries, $\mathcal{B}$ proceeds as before for the first $\ell - 1$ keys except using just $\mathbf{v}$ instead of $\mathbf{v} + \mathbf{A}^\perp\boldsymbol{\delta}^{(q)}$. It is easy to see that it forms a Normal attribute decryption key.

- For the ℓth attribute decryption key query for $\vec{x}$, queries $\mathcal{O}_{\mathsf{X}}(\vec{x}) \rightarrow (\{\mathbf{w}_i\}_{x_i=1})$ and creates the attribute decryption key: $\mathsf{DK}_{\vec{x}} :=$

$$(\mathsf{dk}_1 = [\underbrace{\widetilde{\mathbf{v}} + \widetilde{\mathbf{U}}_0\mathbf{d}}_{=\mathbf{v} + \frac{(\mu^{(0)} - \mu^{(\beta)})((\mathbf{b}^\perp)^\top\mathbf{d})}{(\mathbf{c}^\top\mathbf{A}^\perp\mathbf{u})}\mathbf{A}^\perp\mathbf{u} + \mathbf{U}_0\mathbf{d}}]_2, \quad \mathsf{dk}_2 = [\mathbf{d}]_2,$$

$$\mathsf{dk}_3 = [\textstyle\sum_{i:x_i^{(out)}=1} \underbrace{(\widetilde{\mathbf{W}}_i + \mathbf{A}^\perp\mathbf{w}_i(\mathbf{b}^\perp)^\top)\mathbf{d}}_{=\mathbf{W}_i\mathbf{d}}]_2).$$

We claim that if $\beta = 0$, then the ℓth key is a $\mathsf{P\text{-}Normal}$ attribute decryption key since $\mathbf{v} + \frac{(\mu^{(0)} - \mu^{(0)})((\mathbf{b}^\perp)^\top\mathbf{d})}{(\mathbf{c}^\top\mathbf{A}^\perp\mathbf{u})}\mathbf{A}^\perp\mathbf{u} = \mathbf{v}$; and if $\beta = 1$, then the ℓth key is a $\mathsf{P\text{-}SF}$ key since $\boldsymbol{\delta}^{(\ell)} = \frac{(\mu^{(0)} - \mu^{(1)})((\mathbf{b}^\perp)^\top\mathbf{d})}{(\mathbf{c}^\top\mathbf{A}^\perp\mathbf{u})}\mathbf{u}$.

Policy Decryption Key. $\mathcal{B}$ can simulate any policy key normally since the elements for generating policy decryption key are generated by $\mathcal{B}$.

Challenge. When $\mathcal{A}$ requests a challenge ciphertext for symmetric keys (K_0, K_1), attributes $\vec{x}$ and formula f, $\mathcal{B}$ flips a random coin $\mathsf{coin} \leftarrow \{0, 1\}$ and constructs the challenge ciphertext for K_{coin}. $\mathcal{B}$ queries $\mathcal{O}_{\mathsf{F}}(f) \rightarrow (\{[\mu_j + \mathbf{r}_j^\top\mathbf{w}_{\rho(j)}]_1, [\mathbf{r}_j]_1\})$ on input f. $\mathcal{B}$ samples $\widetilde{\mathbf{c}}_j \leftarrow \mathbb{Z}_p^k$ for each j, defines $\mathbf{A}_C^\perp := \begin{bmatrix}(\mathbf{A}^\perp)^\top \\ \mathbf{M}\end{bmatrix} \in \mathbb{Z}_p^{2k\times 2k}$ for a choice of $\mathbf{M}$ that makes $\mathbf{A}_C^\perp$ invertible, computes $(\{\widetilde{\mathbf{u}}_j^\top\}, \rho) \leftarrow \mathsf{share}(f, \mathbf{c}^\top\widetilde{\mathbf{U}}_0)$, $[\mathbf{c}_j]_1 := \left[(\mathbf{A}_C^\perp)^{-1}\begin{pmatrix}\mathbf{r}_j \\ \widetilde{\mathbf{c}}_j\end{pmatrix}\right]_1$, and constructs the (SF) challenge ciphertext $\mathsf{CT}_{\vec{x},f} :=$

$$(\mathsf{ct}_0 = e([\widetilde{\mathbf{z}}^\top + \mathbf{c}^\top]_1, [\mathbf{v}]_2) \cdot K_{\mathsf{coin}},$$

$$\mathsf{ct}_1' = [\widetilde{\mathbf{z}}^\top]_1, \mathsf{ct}_2' = [\widetilde{\mathbf{z}}^\top \sum_{i:x_i^{(out)}=1} \underbrace{\widetilde{\mathbf{W}}_i + \mathbf{A}^\perp\mathbf{w}_i(\mathbf{b}^\perp)^\top}_{=\mathbf{W}_i}]_1,$$

$$\mathsf{ct}_1 = [\mathbf{c}^\top]_1, \quad \widetilde{\mathsf{ct}}_j = [\mathbf{c}_j^\top]_1,$$

$$\mathsf{ct}_{\rho(j),j} = [\widetilde{\mathbf{u}}_j^\top + (\mu_j + \mathbf{r}_j^\top\mathbf{w}_{\rho(j)})(\mathbf{b}^\perp)^\top + \mathbf{c}_j^\top\widetilde{\mathbf{W}}_{\rho(j)}]_1),$$

$$\mathsf{ct}_{i,j} = \mathbf{r}_j^\top\mathbf{w}_i(\mathbf{b}^\perp)^\top + \mathbf{c}_j^\top\widetilde{\mathbf{W}}_i]_1).$$

We can deduce that

$$\mathsf{ct}_{\rho(j),j} = [\underbrace{\widetilde{\mathbf{u}}_j^\top + \mu_j(\mathbf{b}^\perp)^\top}_{\equiv\mathsf{share}(f,\mathbf{c}^\top\mathbf{U}_0)} + \underbrace{\mathbf{c}_j^\top\widetilde{\mathbf{W}}_{\rho(j)} + \mathbf{r}_j^\top\mathbf{w}_{\rho(j)}(\mathbf{b}^\perp)^\top}_{=\mathbf{c}_j^\top\mathbf{W}_{\rho(j)}}]_1.$$

It should be noted that $\{\mu_j(\mathbf{b}^\perp)^\top\}$ is distributed like the output of $\mathsf{share}(f, \mu^b(\mathbf{b}^\perp)^\top)$, and therefore due to linearity and the fact that $\mathbf{c}^\top\mathbf{U}_0 = \mathbf{c}^\top\widetilde{\mathbf{U}}_0 + \mu^{(b)}(\mathbf{b}^\perp)^\top$, then $\{\widetilde{\mathbf{u}}_j^\top + \mu_j(\mathbf{b}^\perp)^\top\}$ is distributed like $\mathsf{share}(f, \mathbf{c}^\top\widetilde{\mathbf{U}}_0 + \mu^{(b)}(\mathbf{b}^\perp)^\top) \equiv \mathsf{share}(f, \mathbf{c}^\top\mathbf{U}_0)$. Also, note that $\mathbf{c}_j^\top\mathbf{W}_i = \mathbf{c}_j^\top\widetilde{\mathbf{W}}_i + \mathbf{r}_j\mathbf{w}_{\rho(j)}\mathbf{b}^\perp$ since $\mathbf{c}_j^\top\mathbf{A}^\perp = \mathbf{r}_j$.

Guess. $\mathcal{A}$ halts the game with a guess $\mathsf{coin}' \leftarrow \{0, 1\}$. $\mathcal{B}$ outputs 1 if $\mathsf{coin}' = \mathsf{coin}$, and 0 otherwise.

Putting everything together, we can see that $\mathcal{B}$ simulates $\mathsf{H}_{2,\ell,1}$ when $\beta = 0$; and $\mathsf{H}_{2,\ell,2}$ when $\beta = 1$. $\square$

Lemma B.4. Under the $MDDH_k$ assumption, we have

$$|Pr[\langle\mathcal{A}, \mathsf{H}_{2,\ell,2}\rangle = 1] - Pr[\langle\mathcal{A}, \mathsf{H}_{2,\ell,3}\rangle = 1]| = \mathsf{negl}(\lambda).$$

Proof. Omitted, since this proof is similar to that of Lemma E.3. We need to substitute $\mathbf{v}$ with $\mathbf{v} + \mathbf{A}^\perp\boldsymbol{\delta}^{(\ell)}$ for the ℓth attribute decryption key query, where $\boldsymbol{\delta}^{(\ell)}$ is a random element. $\square$

Lemma B.5. We have

$$|Pr[\langle\mathcal{A}, \mathsf{H}_{2,Q_x,3}\rangle = 1] - Pr[\langle\mathcal{A}, \mathsf{H}_3\rangle = 1]| \leq 1/p$$

unconditionally.

Proof. These two hybrids are identically distributed conditioned on $\mathbf{c}^\top\mathbf{A}^\perp \neq \mathbf{0}$. To see this, consider two ways of choosing $\mathbf{v} : \mathbf{v} = \widetilde{\mathbf{v}} \leftarrow \mathbb{Z}_p^{2k}$ and $\mathbf{v} : \mathbf{v} = \widetilde{\mathbf{v}} + \mathbf{A}^\perp\widetilde{\mathbf{m}}$ for an independently random $\widetilde{\mathbf{m}} \leftarrow \mathbb{Z}_p^k$. Note that both result in $\mathbf{v}$ having a uniform distribution.

Using $\widetilde{\mathbf{v}}$ to simulate hybrid $\mathsf{H}_{2,Q_x,3}$ obviously results in $\mathsf{H}_{2,Q_x,3}$ (where $\mathbf{v} = \widetilde{\mathbf{v}}$). However, using the identically distributed $\mathbf{v} = \widetilde{\mathbf{v}} + \mathbf{A}^{\perp}\widetilde{\mathbf{m}}$ to simulate $\mathsf{H}_{2,Q_x,3}$ results in H_3 with $\widetilde{K} = K_{\mathsf{coin}} \cdot [\mathbf{c}^{\top}\mathbf{A}^{\perp}\widetilde{\mathbf{m}}]_T$. Note that the information of $\widetilde{\mathbf{m}}$ is not leaked to $\mathcal{A}$ from the secret key queries since $\widetilde{\mathbf{m}}$ is blinded by random value $\boldsymbol{\delta}^{(i)}$ for each key. Therefore, $\widetilde{K}$ is distributed uniformly at random over G_T as long as $\mathbf{c}^{\top}\mathbf{A}^{\perp} \neq \mathbf{0}$.

Since $\mathbf{c}$ is chosen at random and independent from $\widetilde{\mathbf{s}}^{\top}\mathbf{A}$ and $\mathbf{A}^{\perp} \neq \mathbf{0}$, so $\mathbf{c}^{\top}\mathbf{A}^{\perp} = \mathbf{0}$ with probability $1/p$, and since we know that $\mathsf{H}_{2,Q_x,3} \equiv \mathsf{H}_3$ conditioned on $\mathbf{c}^{\top}\mathbf{A}^{\perp} \neq \mathbf{0}$, then the lemma follows. ☐

Case 2: the adversary does not query a key for f_{rcv} such that $f_{\mathsf{rcv}}(\vec{x}^{(out)*}_{\mathsf{snd}}) = 1$.

A ciphertext (under access policy f and attributes $\vec{x}$) can be in one of the following forms:

- Normal: A normal ciphertext is generated by Enc.

- SF: An SF ciphertext is the same as Normal ciphertext, except that $\widetilde{\mathbf{s}}^{\top}\mathbf{A}$ is replaced with $\widetilde{\mathbf{c}}^{\top} \leftarrow \mathbb{Z}_p^{2k}$. That is $\mathsf{CT}_{\vec{x},f} :=$

$$(\mathsf{ct}_0 = e([\boxed{\widetilde{\mathbf{c}}^{\top}} + \mathbf{c}^{\top}]_1, [\mathbf{v}]_2) \cdot K,$$

$$\mathsf{ct}'_1 = [\boxed{\widetilde{\mathbf{c}}^{\top}}]_1, \mathsf{ct}'_2 = [\boxed{\widetilde{\mathbf{c}}^{\top}} \textstyle\sum_{i:x_i^{(out)}=1} \mathbf{W}_i]_1,$$

$$\mathsf{ct}_1 = [\mathbf{c}^{\top}]_1, \{\widetilde{\mathsf{ct}}_j = [\mathbf{c}_j^{\top}]_1, \mathsf{ct}_{\rho(j),j} = [\mathbf{u}_j^{\top} + \mathbf{c}_j^{\top}\mathbf{W}_{\rho(j)}]_1,$$

$$\mathsf{ct}_{i,j} = [\mathbf{c}_j^{\top}\mathbf{W}_i]_1\}),$$

where $\mathbf{c}^{\top} = \mathbf{s}^{\top}\mathbf{A}$ and $\mathbf{c}_j^{\top} = \mathbf{s}_j^{\top}\mathbf{A}$ are same as in the original ciphertext.

A secret key (for attributes $\vec{x}$) follows its normal form. A secret key (for a Boolean formula f) can be one of the following forms:

- Normal: A normal secret key is generated by PolGen.

- SF: An SF key is sampled as a Normal key, except $\mathbf{v}$ is replaced with $\mathbf{v} + \delta\mathbf{a}^{\perp}$, where a fresh δ is chosen per SF key and $\mathbf{a}^{\perp}$ is any fixed $\mathbf{a}^{\perp} \in \mathbb{Z}_p^{2k}\backslash\{\mathbf{0}\}$. That is $\mathsf{DK}_f :=$

$$\big(\mathsf{dk}_j = [\mathbf{r}_j]_2, \mathsf{dk}_{\rho(j),j} = [\mathbf{v}_j + \mathbf{W}_{\rho(j)}\mathbf{r}_j]_2, \mathsf{dk}_{i,j} = [\mathbf{W}_i\mathbf{r}_j]_2\big).$$

where $(\{\mathbf{v}_j\}_{j\in[\hat{m}]}, \rho) \xleftarrow{\$} \mathsf{share}(f, \boxed{\mathbf{v} + \delta\mathbf{a}^{\perp}})$, $\mathbf{r}_j \xleftarrow{\$} \mathbb{Z}_p^k$.

Next, we define hybrid sequences for the proof. Assume the adversary $\mathcal{A}$ makes at most Q_x attribute decryption key queries and Q_f policy decryption key queries.

- H_0: This is the real game where all secret keys and ciphertexts are Normal.

- H_1: This game is the same as H_0 except that the challenge ciphertext is SF.

- $\mathsf{H}_{2,\ell}$: This game is the same as H_1 except that the first ℓ policy decryption keys are SF and the remaining $Q_f - \ell$ keys are Normal, where $\ell = 0, \cdots, Q_f$.

- H_3: This game is the same as H_{2,Q_f} except that the message encryption symmetric key K to be encrypted is replaced by a random $\widetilde{K}$.

Lemma B.6. Under the $MDDH_k$ assumption on G_1, we have

$$|Pr[\langle\mathcal{A}, \mathsf{H}_0\rangle = 1] - Pr[\langle\mathcal{A}, \mathsf{H}_1\rangle = 1]| = \mathsf{negl}(\lambda).$$

Proof. If $\mathcal{A}$ can distinguish H_0 from H_1 with non-negligible advantage, then we can construct an algorithm $\mathcal{B}$ that can solve the $MDDH_k$ assumption. On input a $MDDH_k$ challenge $([\mathbf{A}]_1, [\widetilde{\mathbf{z}}]_1)$, where either $\widetilde{\mathbf{z}}^{\top} = \widetilde{\mathbf{s}}^{\top}\mathbf{A}$ or $\widetilde{\mathbf{z}} = \widetilde{\mathbf{c}}$ for $\widetilde{\mathbf{s}} \leftarrow \mathbb{Z}_p^k$ and $\widetilde{\mathbf{c}} \leftarrow \mathbb{Z}_p^{2k}$. $\mathcal{B}$ proceeds as in the proof of Lemma E.2 except that an SF challenge ciphertext is returned to $\mathcal{A}$. $\mathcal{B}$ flips a random coin $\mathsf{coin} \leftarrow \{0,1\}$ and constructs the challenge ciphertext for K_{coin} by computing $(\{\mathbf{u}_j^{\top}\}, \rho) \leftarrow \mathsf{share}(f, \mathbf{z}^{\top}\mathbf{U}_0)$, where $\mathbf{z}^{\top} = \mathbf{s}^{\top}\mathbf{A}$, and $\mathbf{z}_j^{\top} = \mathbf{s}_j^{\top}\mathbf{A}$ as in the normal ACME construction, and sets the challenge ciphertext as $\mathsf{CT}_{\vec{x},f} :=$

$$\big(\mathsf{ct}_0 = e([\widetilde{\mathbf{z}}^{\top} + \mathbf{z}^{\top}]_1, [\mathbf{v}]_2) \cdot K_{\mathsf{coin}},$$

$$\mathsf{ct}'_1 = [\widetilde{\mathbf{z}}^{\top}]_1, \mathsf{ct}'_2 = [\widetilde{\mathbf{z}}^{\top} \textstyle\sum_{i:x_i^{(out)}=1} \mathbf{W}_i]_1,$$

$$\mathsf{ct}_1 = [\mathbf{z}^{\top}]_1, \{\widetilde{\mathsf{ct}}_j = [\mathbf{z}_j^{\top}]_1, \mathsf{ct}_{\rho(j),j} = [\mathbf{u}_j^{\top} + \mathbf{z}_j^{\top}\mathbf{W}_{\rho(j)}]_1.$$

$$\mathsf{ct}_{i,j} = [\mathbf{z}_j^{\top}\mathbf{W}_i]_1\}\big).$$

Guess. $\mathcal{A}$ halts the game with a guess $\mathsf{coin}' \leftarrow \{0,1\}$. $\mathcal{B}$ outputs 1 if $\mathsf{coin}' = \mathsf{coin}$, and 0 otherwise. It is straight forward to see that if $\widetilde{\mathbf{z}}^{\top} = \widetilde{\mathbf{s}}^{\top}\mathbf{A}$, the challenge ciphertext is Normal and $\mathcal{B}$ simulates H_0; if $\widetilde{\mathbf{z}}^{\top} = \widetilde{\mathbf{c}}^{\top}$, the challenge ciphertext is SF and $\mathcal{B}$ simulates H_1. ☐

Lemma B.7. Under the $MDDH_k$ assumption on G_2, we have

$$|Pr[\langle\mathcal{A}, \mathsf{H}_{2,\ell-1}\rangle = 1] - Pr[\langle\mathcal{A}, \mathsf{H}_{2,\ell}\rangle = 1]| = \mathsf{negl}(\lambda).$$

Proof. Assume that $\mathcal{A}$ distinguishes $\mathsf{H}_{2,\ell-1}$ and $\mathsf{H}_{2,\ell}$ with non-negligible advantage. Then, we can construct another adversary $\mathcal{B}$ that distinguishes the oracles in $\mathsf{G}_{\beta}^{1-\mathsf{ABE}}$ of [27], which implies an attacker against the $MDDH_k$ assumption. Given $\mu^{(0)}$ as an input and equipped with oracles $\mathcal{O}_{\mathsf{F},\beta}$, $\mathcal{O}_X$ and $\mathcal{O}_E$ (defined in $\mathsf{G}_{\beta}^{1-\mathsf{ABE}}$ of [27]), $\mathcal{B}$ proceeds as below.

Setup. $\mathcal{B}$ chooses generators $g \leftarrow G_1$, $h \leftarrow G_2$, user's attribute number n, and sets $\mathsf{pp} = (g, h, n)$. Next, $\mathcal{B}$ chooses $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_p^{k\times 2k}$, $\widetilde{\mathbf{U}}_0, \widetilde{\mathbf{W}}_i \xleftarrow{\$} \mathbb{Z}_p^{2k\times k}$ for $i \in [1,n]$, and $\widetilde{\mathbf{v}} \xleftarrow{\$} \mathbb{Z}_p^{2k}$, computes $\mathbf{a}^{\perp} \in \mathbb{Z}_p^{2k}\backslash\{\mathbf{0}\}$ such that $\mathbf{A}\mathbf{a}^{\perp} = \mathbf{0}$. It then sets $\widetilde{\mathbf{W}}_0 := \mathbf{0}$ and implicitly defines $\mathbf{v} := \widetilde{\mathbf{v}} + \mu^{(0)}\mathbf{a}^{\perp}$, $\mathbf{W}_i := \widetilde{\mathbf{W}}_i + \mathbf{a}^{\perp}\mathbf{w}_i^{\top}$, where $\mathbf{w}_i \in \mathbb{Z}_p^k$, $\mu^{(0)} \in \mathbb{Z}_p$ are chosen by the $\mathsf{G}_{\beta}^{1\text{-}\mathsf{ABE}}$ game in [27]. Then, $\mathcal{B}$ creates

$$\mathsf{mpk} := (\mathbf{pp}, [\mathbf{A}]_1, [\mathbf{A}\widetilde{\mathbf{U}}_0]_1, [\mathbf{A}\widetilde{\mathbf{W}}_1]_1, \cdots, [\mathbf{A}\widetilde{\mathbf{W}}_n]_1, e([\mathbf{A}]_1, [\widetilde{\mathbf{v}}]_2)).$$

$\mathcal{B}$ runs $\mathcal{FAC}.\mathsf{CredKeyGen}$ to create $(\mathsf{pk}, \mathsf{sk})$.

Issue Query. $\mathcal{B}$ firstly executes $\mathcal{FAC}.\mathsf{UserKeyGen}$ to create user's public/secret keys $\mathsf{upk}/\mathsf{usk}$. $\mathcal{B}$ can response to any credential issue query since credential issuer's secret key sk is generated by $\mathcal{B}$.

Attribute Decryption Key Query. $\mathcal{B}$ simulates any attribute decryption key normally.

Policy Decryption Key Query. $\mathcal{B}$ responds to $\mathcal{A}$'s policy decryption key queries as below.

- For the first $\ell - 1$ policy decryption key queries, say for formula f of size m, $\mathcal{B}$ computes:

$$(\{\mathbf{v}_j\}_{j\in[\hat{m}]}, \rho) \xleftarrow{\$} \mathsf{share}(f, \underbrace{\widetilde{\mathbf{v}} + \widetilde{\delta}\mathbf{a}^{\perp}}_{=\mathbf{v}+\delta\mathbf{a}^{\perp}}),$$

where $\widetilde{\delta} \xleftarrow{\$} \mathbb{Z}_p$ is drawn independently for each key (here, the per-key $\delta = \widetilde{\delta} - \mu^{(0)}$ implicitly). Next, for each $j \in [\hat{m}]$, it queries $\mathcal{O}_{\mathsf{E}} \to ([\mathbf{r}_j]_2, \{[\mathbf{w}_i^{\top}\mathbf{r}_j]_2\}_{i\in[n]})$ and forms the SF policy decryption key as $\mathsf{DK}_f :=$

$$(\mathsf{dk}_j = [\mathbf{r}_j]_2, \mathsf{dk}_{\rho(j),j} = [\underbrace{\mathbf{v}_j + \widetilde{\mathbf{W}}_{\rho(j)}\mathbf{r}_j + \mathbf{a}^{\perp}\mathbf{w}_{\rho(j)}^{\top}\mathbf{r}_j}_{\mathbf{v}_j+\mathbf{W}_{\rho(j)}\mathbf{r}_j}]_2,$$

$$\mathsf{dk}_{i,j} = [\underbrace{\widetilde{\mathbf{W}}_i\mathbf{r}_j + \mathbf{a}^{\perp}\mathbf{w}_i^{\top}\mathbf{r}_j}_{=\mathbf{W}_i\mathbf{r}_j}]_2).$$

Then, it returns DK_f to $\mathcal{A}$.

- For the last $Q_f - \ell$ policy decryption key queries, say for formula f of size m, $\mathcal{B}$ proceeds as before for the first $\ell - 1$ policy decryption keys except

$$(\{\mathbf{v}_j\}_{j\in[\hat{m}]}, \rho) \xleftarrow{\$} \mathsf{share}(f, \underbrace{\widetilde{\mathbf{v}} + \mu^{(0)}\mathbf{a}^{\perp}}_{=\mathbf{v}}),$$

It is easy to see that it forms a Normal policy decryption key.

- For the ℓ-th policy decryption key query, say for formula f of size m, $\mathcal{B}$ computes $(\{\mathbf{v}_j\}_{j\in[\hat{m}]}, \rho) \xleftarrow{\$} \mathsf{share}(f, \widetilde{\mathbf{v}})$, queries $\mathcal{O}_{\mathsf{F},\beta}(f) \to (\{[\mathbf{r}_j]_2, [\mu_j + \mathbf{w}_{\rho(j)}\mathbf{r}_j]_2, \{[\mathbf{w}_i^{\top}\mathbf{r}_j]_2\}_{i\in[n]\setminus\{\rho(j)\}}\}_{j\in[\hat{m}]})$ and uses these components to return: $\mathsf{DK}_f :=$

$$(\mathsf{dk}_j = [\mathbf{r}_j]_2, \mathsf{dk}_{\rho(j),j} = [\underbrace{\mathbf{v}_j + \widetilde{\mathbf{W}}_{\rho(j)}\mathbf{r}_j + \mathbf{a}^{\perp}(\mu_j + \mathbf{w}_{\rho(j)}^{\top}\mathbf{r}_j)}_{=(\mathbf{v}_j+\mu_j\mathbf{a}^{\perp})+\mathbf{W}_{\rho(j)}\mathbf{r}_j}]_2,$$

$$\mathsf{dk}_{i,j} = [\underbrace{\widetilde{\mathbf{W}}_i\mathbf{r}_j + \mathbf{a}^{\perp}\mathbf{w}_i^{\top}\mathbf{r}_j}_{=\mathbf{W}_i\mathbf{r}_j}]_2).$$

We claim that if $\beta = 0$, then DK_f is a Normal policy decryption key, and if $\beta = 1$, then DK_f is a SF policy key. This follows from the fact that thanks to linearity, the shares $(\{\mathbf{v}_j + \mu_j\mathbf{a}^{\perp}\}_{j\in[\hat{m}]}, \rho)$, where $(\{\mathbf{v}_j\}_{j\in[\hat{m}]}, \rho) \xleftarrow{\$} \mathsf{share}(f, \widetilde{\mathbf{v}})$, $(\{\mu_j\}_{j\in[\hat{m}]}, \rho) \xleftarrow{\$} \mathsf{share}(f, \mu^{(\beta)})$, are identically distributed to $\mathsf{share}(f, \widetilde{\mathbf{v}} + \mu^{(\beta)}\mathbf{a}^{\perp})$. The claim follows the fact that $\mathbf{v} = \widetilde{\mathbf{v}} + \mu^{(0)}\mathbf{a}^{\perp}$, where we set $\delta := \mu^{(1)} - \mu^{(0)}$ is a fresh random value for the key.

Challenge. When $\mathcal{A}$ requests a challenge ciphertext for symmetric keys (K_0, K_1), attributes $\vec{x}$ and formula f, $\mathcal{B}$ flips a random coin $\mathsf{coin} \leftarrow \{0,1\}$ and constructs the challenge ciphertext for K_{coin}. $\mathcal{B}$ queries $\mathcal{O}_{\mathsf{X}}$ on input $\vec{x}$ to obtain $\{\mathbf{w}_i\}_{i:x_i=1}$. $\mathcal{B}$ computes $\mathbf{c}^{\top} = \mathbf{s}^{\top}\mathbf{A}$, $\mathbf{c}_j^{\top} = \mathbf{s}_j^{\top}\mathbf{A}$, $(\{\mathbf{u}_j^{\top}\}, \rho) \leftarrow \mathsf{share}(f, \mathbf{c}^{\top}\mathbf{U}_0)$ normally, samples $\widetilde{\mathbf{c}} \xleftarrow{\$} \mathbb{Z}_p^{2k}$, and constructs the challenge ciphertext $\mathsf{CT}_{\vec{x},f} :=$

$$\big(\mathsf{ct}_0 = e([\widetilde{\mathbf{c}}^{\top} + \mathbf{c}^{\top}]_1, [\widetilde{\mathbf{v}} + \mu^{(0)}\mathbf{a}^{\perp}]_2) \cdot K_{\mathsf{coin}},$$

$$\mathsf{ct}'_1 = [\widetilde{\mathbf{c}}^{\top}]_1, \mathsf{ct}'_2 = [\widetilde{\mathbf{c}}^{\top} \sum_{i:x_i^{(out)}=1} \underbrace{\widetilde{\mathbf{W}}_i + \mathbf{a}^{\perp}\mathbf{w}_i^{\top}}_{=\mathbf{W}_i}]_1,$$

$$\mathsf{ct}_1 = [\mathbf{c}^{\top}]_1, \quad \widetilde{\mathsf{ct}}_j = [\mathbf{c}_j^{\top}]_1,$$

$$\mathsf{ct}_{\rho(j),j} = [\mathbf{u}_j^{\top} + \mathbf{c}_j^{\top} \underbrace{(\widetilde{\mathbf{W}}_{\rho(j)} + \mathbf{a}^{\perp}\mathbf{w}_{\rho(j)}^{\top})}_{=\mathbf{W}_{\rho(j)}}]_1),$$

$$\mathsf{ct}_{i,j} = [\mathbf{c}_j^{\top} \underbrace{(\widetilde{\mathbf{W}}_i + \mathbf{a}^{\perp}\mathbf{w}_i^{\top})}_{=\mathbf{W}_i}]_1).$$

Guess. $\mathcal{A}$ halts the game with a guess $\mathsf{coin}' \leftarrow \{0,1\}$. $\mathcal{B}$ outputs 1 if $\mathsf{coin}' = \mathsf{coin}$, and 0 otherwise.

Putting everything together, we can see that $\mathcal{B}$ simulates $\mathsf{H}_{2,\ell-1}$ when $\beta = 0$; and $\mathsf{H}_{2,\ell}$ when $\beta = 1$. □

Lemma B.8. We have

$$|Pr[\langle \mathcal{A}, \mathsf{H}_{2,Q_f}\rangle = 1] - Pr[\langle \mathcal{A}, \mathsf{H}_3\rangle = 1]| \leq 1/p$$

unconditionally.

Proof. The two hybrids are identically distributed conditioned on $\widetilde{\mathbf{c}}^{\top}\mathbf{a}^{\perp} \neq 0$. To see this, consider two ways to sample $\mathbf{v}$: as $\widetilde{\mathbf{v}} \xleftarrow{\$} \mathbb{Z}_p^{2k}$ and as $\widetilde{\mathbf{v}} + \widetilde{m}\mathbf{a}^{\perp}$ for an independent $\widetilde{m} \xleftarrow{\$} \mathbb{Z}_p$. Both result in $\mathbf{v}$ having a uniform distribution.

Using $\widetilde{\mathbf{v}}$ to simulate hybrid H_{2,Q_f} obviously results in H_{2,Q_f} (where $\mathbf{v} = \widetilde{\mathbf{v}}$). However, using the identically distributed $\mathbf{v} = \widetilde{\mathbf{v}} + \widetilde{m}\mathbf{a}^{\perp}$ to simulate H_{2,Q_f} results in H_3 with $\widetilde{K} = K_{\mathsf{coin}} \cdot [\widetilde{\mathbf{c}}^{\top}\widetilde{m}\mathbf{a}^{\perp}]_T$ and re-defined randomness $\widetilde{\delta}_j = \delta_j + \widetilde{m}$ for all the keys. Note that the information of $\widetilde{m}$ is not leaked to $\mathcal{A}$ from the secret key queries since $\widetilde{m}$ is blinded by random value δ_j for each key. Therefore, $\widetilde{K}$ is distributed uniformly at random over G_T as long as $[\widetilde{\mathbf{c}}^{\top}\mathbf{a}^{\perp}]_T \neq 0$.

Since $\widetilde{\mathbf{c}}$ is chosen at random and independent from $\mathbf{a}^{\perp} \neq \mathbf{0}$, so $[\widetilde{\mathbf{c}}^{\top}\mathbf{a}^{\perp}]_T = 0$ with probability $1/p$, and since we know that $\mathsf{H}_{2,Q_f} \equiv \mathsf{H}_3$ conditioned on $[\widetilde{\mathbf{c}}^{\top}\mathbf{a}^{\perp}]_T \neq 0$, then the lemma follows. □

This completes the proof of Theorem 6.2. □

C. PriSrv: Security Model and Proof

(1) Security Model of PriSrv

We formalize the security model for PriSrv, which includes the service discovery with bilateral control, key secrecy and bilateral anonymity, by following the Canetti-Krawczyk model for authenticated key-exchange (AKE) in [32], [33], [38] and the service discovery model in [5].

1.1) Service Discovery Security

The service discovery *security* captures *service discovery with bilateral control* and *AKE security*. The framework of PriSrv contains two sub-protocols: a private broadcast protocol that announces the service type, server's identifier, as well as other relevant information *in a privacy-preserving manner*; and an anonymous mutual authentication protocol with bilateral policy control. Compared with traditional mutual authentication settings, a remarkable difference in PriSrv is that multiple clients can respond to a service provider's broadcast message if their credentials satisfy the service authorization policy.

Protocol participants. The participants of PriSrv includes a set of clients $\mathsf{C} = \{C_1, \cdots, C_{n_1}\}$ and a set of service providers $\mathsf{S} = \{S_1, \cdots, S_{n_2}\}$.

Long-term Keys. Each $C_i \in \mathsf{C}$ and $S_j \in \mathsf{S}$ hold long-term secret keys for bilateral authentication and message decryption.

Session and Pairing. Denote the ρ-th instance of participant $U \in \mathsf{C} \cup \mathsf{S}$ as U^ρ, which is modeled as a PPT Turing machine. A participant U^ρ can be activated to initiate a *session* with a broadcast identifier bid_U^ρ, a session identifier sid_U^ρ, attributes $\vec{x}_U^\rho$, and a policy f_U^ρ. A client instance C_i^ρ and a service provider instance S_j^δ are said to be *paired* if their session instances $(C_i^\rho, bid_{C_i}^\rho, sid_{C_i}^\rho, \vec{x}_{C_i}^\rho, f_{C_i}^\rho)$ and $(S_j^\delta, bid_{S_j}^\delta, sid_{S_j}^\delta, \vec{x}_{S_j}^\delta, f_{S_j}^\delta)$ satisfy $bid_{C_i}^\rho = bid_{S_j}^\delta$, $sid_{C_i}^\rho = sid_{S_j}^\delta$, $f_{S_j}^\delta(\vec{x}_{C_i}^{\rho(out)}) = 1$, $f_{C_i}^\rho(\vec{x}_{S_j}^{\delta(out)}) = 1$. A completed *session* contains a tuple $(C_i^\rho, bid_{C_i,S_j}^{\rho,\delta}, sid_{C_i,S_j}^{\rho,\delta}, S_j^\delta, SSK_{C_i,S_j}^{\rho,\delta})$, where $bid_{C_i,S_j}^{\rho,\delta} = bid_{C_i}^\rho = bid_{S_j}^\delta$, $sid_{C_i,S_j}^{\rho,\delta} = sid_{C_i}^\rho = sid_{S_j}^\delta$.

Adversary Capability. We capture all of the adversary's attack capabilities in real world to have full control over the public network communication, including revealing some secrets in the protocol, intercepting or tampering with the channel messages, replaying, delaying, injecting or dropping data packets, interleaving messages from different sessions, etc.

Protocol Execution. An adversary $\mathcal{A}$ is modeled as a PPT machine with a distinguished query tape to issue a set of session exposure queries for gaining the ephemeral and long-term secrets possessed by participants.

- $\mathsf{Send}(U^\rho, M)$: transmits a message M to U^ρ, who executes the protocol and returns the operation result to adversary $\mathcal{A}$. If the message in the query causes the protocol to execute or abort, it will be made known to $\mathcal{A}$.
- $\mathsf{Execute}(C_i^\rho, S_j^\delta)$: executes a complete protocol between C_i^ρ and S_j^δ. The adversary captures all messages transmitted over the public network. Hence, the query to $\mathsf{Execute}$ oracle models passive eavesdropping capability of the adversary.
- $\mathsf{RevealBroadcast}(S_j^\delta, bid)$: returns the semi-static state in a service provider S_j^δ that is maintained for the lifetime of its current broadcast with identifier bid, including the attributes $\vec{x}_{S_j^\delta}$ and authorization policy $f_{S_j^\delta}$. The revealed state does not involve the long-term secrets. $\mathcal{A}$ is allowed to make query for any service provider $S_j \in \mathsf{S}$.
- $\mathsf{RevealState}(U^\rho, bid, sid)$: returns the local state associated with the targeted session, which does not contain the long-term secrets.
- $\mathsf{RevealKey}(U^\rho, bid, sid)$: outputs the secret session key associated with a targeted session.
- $\mathsf{Corrupt}(U)$: returns all information (including ephemeral and long-term secrets) held by U.
- $\mathsf{TestSession}(U^\rho, bid, sid)$: This oracle is used to model key secrecy. A random bit $b \in \{0, 1\}$ is selected to respond this query. If $b = 1$, the target session key is returned to $\mathcal{A}$. Otherwise, a randomly value picked from the secret session key space is returned.

Session Exposure. A session (U^ρ, bid, sid) is said to be *exposed* if the adversary makes the following queries.

- The adversary makes a $\mathsf{RevealKey}$ query on the session.

- The adversary makes a $\mathsf{Corrupt}$ query on U, or any partner with $(\vec{x}, f)$ satisfying $f_U(\vec{x}^{(out)}) = 1 \wedge f(\vec{x}_U^{(out)}) = 1$, before the session has expired.

- U is the client in the protocol and the adversary has made a $\mathsf{RevealState}$ query on the session.

- U is the server in the protocol and the adversary has made a $\mathsf{RevealState}$ query on the session, and also made a $\mathsf{RevealBroadcast}$ query on the session or a $\mathsf{Corrupt}$ query on U before the session has expired.

Session Freshness. A session (U^ρ, bid, sid) is said to be *fresh* if itself is not exposed and all its matching sessions are not exposed.

Definition C.1 Let $\mathsf{Succ}_{\mathsf{PriSrv}}^{\mathsf{Sec}}(\mathcal{A})$ denote the event that $\mathcal{A}$ makes a single $\mathsf{TestSession}$ query with the restriction that the queried session (U^ρ, bid, sid) is fresh, and finally outputs a bit $b' = b$, where b is the random value selected in the $\mathsf{TestSession}$ query. A private service discovery protocol PriSrv is secure if for any PPT adversary $\mathcal{A}$, there exists a negligible function ν such that $\mathsf{Adv}_{\mathsf{PriSrv}}^{\mathsf{Sec}}(\mathcal{A}) \overset{\mathsf{def}}{=} 2\mathsf{Pr}[\mathsf{Succ}_{\mathsf{PriSrv}}^{\mathsf{Sec}}(\mathcal{A})] - 1 \le \nu(\lambda)$.

1.2) Bilateral Anonymity

The bilateral anonymity property implies that no PPT service provider (or client) can learn anything about another participant's identifier and private attributes unless it satisfies the latter's authorization policy. The adversary is permitted to compromise multiple participants. This property should hold provided that the compromised participants do not satisfy the target's policy. We include the registration query oracle in the security model for the bilateral anonymity proof. In the following, we firstly define the security game to prove the client anonymity, which captures the property that no adversary can distinguish the interactions with $C_{i_0}^*$ or $C_{i_1}^*$ (challenge clients). Let n be the number of parties participating in the protocol execution experiment, which are denoted as $(P_1, \cdots, P_n)$. A special test party P_T is introduced at the beginning of the experiment whose identity is kept confidential from the adversary. We introduce two experiments Exp_0 and Exp_1, and select random $b \in \{0, 1\}$ at the beginning of the game. The experiment Exp_b proceeds as below.

Setup Phase. At the beginning of the experiment, adversary $\mathcal{A}$ submits a set of identities (with attributes and policies) $\{(uid_j, \vec{x}_j, f_j)\}_{j=1}^n$ for parties $(P_1, \cdots, P_n)$. For each $j \in [n]$, the challenger sets up anonymous credential and long-term secret attribute/policy keys for party P_j. $\mathcal{A}$ also submits two challenge clients $C_{i_0}^*$, $C_{i_1}^* \in \mathsf{C}$, where $C_{i_0}^*$ possesses $(uid_{i_0}^*, \vec{x}_{i_0}^*, f_{i_0}^*)$ and $C_{i_1}^*$ has $(uid_{i_1}^*, \vec{x}_{i_1}^*, f_{i_1}^*)$. It is required that $\vec{x}_{i_0}^* = \vec{x}_{i_1}^*$ and $f_{i_0}^* = f_{i_1}^*$. Note that the challenge tuples $(uid_{i_0}^*, \vec{x}_{i_0}^*, f_{i_0}^*)$ and $(uid_{i_1}^*, \vec{x}_{i_1}^*, f_{i_1}^*)$ are distinct from $\{(uid_j, \vec{x}_j, f_j)\}_{j=1}^n$. Then, the challenger associates

$(uid^*_{i_b}, \vec{x}^*_{i_b}, f^*_{i_b})$ with P_T, and executes the setup algorithm for P_T that is defined in the protocol.

Protocol Execution. Adversary $\mathcal{A}$ is allowed to issue the following queries.

• $\mathsf{Reg}(U^\rho, uid^\rho_U, \vec{x}^\rho_U, f^\rho_U)$: If $U^\rho \in (P_1, \cdots, P_n)$ with user identifier uid^ρ_U, attributes $\vec{x}^\rho_U$ and policy f^ρ_U is unregistered, it executes as the protocol definition, and returns the result to $\mathcal{A}$.

• Send, RevealBroadcast, RevealState, RevealKey and Corrupt are the same as the definition in "Service Discovery Security" model.

• Challenge. In Exp_b, the instance $C^*_{i_b}$ with $(uid^*_{i_b}, \vec{x}^*_{i_b}, f^*_{i_b})$ executes PriSrv by following the protocol steps. The restriction is that $\mathcal{A}$ does not issue any of the following queries:

- Reg query on $(uid^*_{i_0}, \vec{x}^*_{i_0}, f^*_{i_0})$ or $(uid^*_{i_1}, \vec{x}^*_{i_1}, f^*_{i_1})$ or any service provider whose attributes and policy satisfy $(\vec{x}^*_{i_0}, f^*_{i_0})$ or $(\vec{x}^*_{i_1}, f^*_{i_1})$;

- RevealState or RevealKey query on any (P_T, bid, sid) or its matching session;

- Corrupt query on $C^*_{i_0}$ or $C^*_{i_1}$ or any service provider whose attributes and policy satisfy $(\vec{x}^*_{i_0}, f^*_{i_0})$ or $(\vec{x}^*_{i_1}, f^*_{i_1})$.

- If $\mathcal{A}$ associates a policy $(\vec{x}_{S_j}, f_{S_j})$ with a service provider session (S^δ_j, bid, sid), it is required that either both $C^*_{i_0}$ and $C^*_{i_1}$ satisfy the policy, or neither of them satisfies the policy.

In any above query, if the query causes an instance to accept or termination, these outputs will be shown to $\mathcal{A}$.

Output phase. $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$ for b.

A service discovery protocol achieves client anonymity if no adversary can distinguish the experiments Exp_0 and Exp_1 with non-negligible advantage greater than 1/2, which captures the property that no active adversary can distinguish communications with a client $C^*_{i_0}$ from those with a client $C^*_{i_1}$. Here is the formal definition.

Definition C.2 Let $\mathsf{Succ}^{\text{anon-C}}_{\mathsf{PriSrv}}(\mathcal{A})$ denote the event that $\mathcal{A}$ outputs a bit $b' = b$. PriSrv satisfies ***client anonymity*** if for any PPT adversary $\mathcal{A}$, there exists a negligible function ν such that $\mathsf{Adv}^{\text{anon-C}}_{\mathsf{PriSrv}}(\mathcal{A}) \overset{\text{def}}{=} 2\Pr[\mathsf{Succ}^{\text{anon-C}}_{\mathsf{PriSrv}}(\mathcal{A})] - 1 \leq \nu(\lambda)$.

The security game for *service provider anonymity* is similar to that for the client anonymity, except for exchanging their roles and restriction that the adversary is not allowed to query Revealbroadcast for the challenge sessions. The concrete security model is omitted for brevity.

Definition C.3 Let $\mathsf{Succ}^{\text{anon-S}}_{\mathsf{PriSrv}}(\mathcal{A})$ denote the event that $\mathcal{A}$ outputs a bit $b' = b$. PriSrv satisfies ***service provider anonymity*** if for any PPT adversary $\mathcal{A}$, there exists a negligible function ν such that $\mathsf{Adv}^{\text{anon-S}}_{\mathsf{PriSrv}}(\mathcal{A}) \overset{\text{def}}{=} 2\Pr[\mathsf{Succ}^{\text{anon-S}}_{\mathsf{PriSrv}}(\mathcal{A})] - 1 \leq \nu(\lambda)$.

Definition C.4 PriSrv satisfies ***bilateral anonymity*** if it achieves *client anonymity* and *service provider anonymity*.

(2) Security Proof of PriSrv

Theorem 7.1. Suppose that the DDH assumption holds, $\mathcal{ACME}$ is secure, $\mathcal{MAC}$ is unforgeable, and H is a random oracle, then PriSrv is a secure service discovery protocol and satisfies bilateral anonymity.

We utilize three lemmas to prove the security of PriSrv in Theorem 7.1, which demonstrate PriSrv is secure and private in extended Canetti-Krawzyk key-exchange model (Lemma C.1), and it provides anonymity for both the client and the server (Lemma C.2 and Lemma C.3).

The security proof of PriSrv requires the underlying ACME scheme should be CCA secure. There are standard (and efficient) generic approaches (e.g., the Fujisaki-Okamoto transformation [59]) to transform our ACME construction to achieve CCA security.

Lemma C.1. (**Service discovery privacy with bilateral control**.) Suppose that DDH assumption holds on G_1 and G_2, $\mathcal{ACME}$ is secure, $\mathcal{MAC}$ is unforgeable, and H is random oracle, then PriSrv is a secure service discovery protocol with bilateral control.

Proof. Following similar proofs for key-exchange and service discovery protocols proposed in [32], [33], [38], [5], we assume selective security in the adversary's choice of the test session, i.e., at the beginning of the security game, the adversary commits to the following:

• The test session $(\alpha^*, bid, sid, \vec{x}_{\alpha^*}, f_{\alpha^*})$.

• The peer's identity, attributes and policy $(\beta^*, \vec{x}_{\beta^*}, f_{\beta^*})$.

• Whether α^* is the initializer or the responder of the test session.

Note that a selective security proof can be converted to an adaptive one at a security loss that increases polynomially in the number of parties and the number of sessions the adversary initiates.

We define a simulator $\mathcal{S} = \mathcal{S}(\mathcal{A})$. On input the number of parties n and an adversary $\mathcal{A}$, the simulator $\mathcal{S}$ simulates a series of security games for the protocol. In the selective security setting, the adversary begins by committing to a test session $(\overline{\alpha^*}, \overline{bid}, \overline{sid}, \overline{\vec{x}_{\alpha^*}}, \overline{f_{\alpha^*}})$, the peer $(\overline{\beta^*}, \overline{\vec{x}_{\beta^*}}, \overline{f_{\beta^*}})$ in the test session, and whether $\overline{\alpha^*}$ was the initiator or the responder in the test session. Then, $\mathcal{S}$ initializes the n parties by generating anonymous credential, attribute decryption key and policy decryption key for each of them. When $\mathcal{A}$ activates a party, $\mathcal{S}$ executes as in the protocol on behalf of the parties, and outputs the corresponding messages to $\mathcal{A}$ as well as the public outputs of each session.

Description of the simulator. We introduce several variants of $\mathcal{S}$, which are generally denoted as $\overline{\mathcal{S}}$. The simulator $\overline{\mathcal{S}}$ behaves similarly to $\mathcal{S}$ except the following differences.

(1) At the beginning of the simulation, the simulator chooses four exponents $\overline{z}, \overline{x}_1, \overline{x}_2, \overline{y} \xleftarrow{\$} \mathbb{Z}^*_p$ and a random session key $\overline{SSK}$. The specification of the keys will determine the different variants of the simulator $\overline{\mathcal{S}}$.

(2) In the selective security model, the adversary $\mathcal{A}$ commits to a test session $(\overline{\alpha^*}, \overline{bid}, \overline{sid}, \overline{\vec{x}_{\alpha^*}}, \overline{f_{\alpha^*}})$, the peer $(\overline{\beta^*}, \overline{\vec{x}_{\beta^*}}, \overline{f_{\beta^*}})$, and the role of $\overline{\alpha^*}$ in the session at the beginning of the experiment. Let $\overline{S} \in \{\overline{\alpha^*}, \overline{\beta^*}\}$ denote the server $\mathcal{A}$ commits to for a test session, and $\overline{C} \in \{\overline{\alpha^*}, \overline{\beta^*}\}$ the client to which it commits.

The simulator $\overline{S}$ simulates the execution of the PriSrv security game as $\mathcal{S}$, except for the following differences.

• If the adversary $\mathcal{A}$ activates $\overline{S}$ to initiate the broadcast $(\overline{S}, \overline{bid}, \overline{\vec{x}_S}, \overline{f_S})$, the simulator uses $\overline{z}$ as the semi-static DH exponent in the broadcast.

• If the adversary $\mathcal{A}$ activates $\overline{C}$ to initiate the session $(\overline{C}, \overline{bid}, \overline{sid}, \overline{\vec{x}_C}, \overline{f_C})$, the simulator uses $\overline{x}_1, \overline{x}_2$ as the ephemeral DH exponents of $\overline{C}$.

• If the adversary $\mathcal{A}$ activates $\overline{S}$ as a responder to the session $(\overline{S}, \overline{C}, \overline{bid}, \overline{sid}, \overline{\vec{x}_S}, \overline{f_S}, \overline{\vec{x}_C}, \overline{f_C})$, the simulator uses $\overline{y}$ as the ephemeral DH exponent of $\overline{S}$.

• It uses $\overline{SSK}$ in place of SSK whenever the shares $(h^{\overline{z}}, g^{\overline{x}_1}, h^{\overline{x}_2}, g^{\overline{y}})$ are used to derive the session key (that is, when the simulator needs to compute $H(\overline{X}_1^{\overline{y}}, \overline{X}_2^{\overline{z}})$).

(3) At the end of the protocol, $\mathcal{A}$ outputs a bit $b \in \{0, 1\}$. The simulator $\overline{S}$ outputs the same bit.

The security proof contains two cases, depending on whether $\mathcal{A}$ compromises the server's semi-static broadcast secret or not. We say the adversary is admissible as long as it is not the situation that both the server's epheral DH secret and the server's semi-static broadcast DH secret are compromised, which is similar to the security analysis in [60]. Specifically, the two cases in our proof are given below.

• **Case 1**: $\mathcal{A}$ neither issues a RevealBroadcast query on $(\overline{S}, \overline{bid}, \overline{\vec{x}_S}, \overline{f_S})$ nor corrupt $\overline{S}$ before the broadcast session expires (i.e., the semi-static broadcast DH secret is not compromised).

• **Case 2**: $\mathcal{A}$ does not issue a RevealState query on $(\overline{S}, \overline{bid}, \overline{\vec{x}_S}, \overline{f_S})$.

For each case, we define a series of hybrid games to show that each consecutive pair of hybrid games are computationally indistinguishable. Before the formal case analysis, we prove the following proposition.

Proposition C.1. Suppose a session $(\alpha^*, bid, sid, \vec{x}_{\alpha^*}, f_{\alpha^*})$ completes with a peer $(\beta^*, \vec{x}_{\beta^*}, f_{\beta^*})$. Assume that neither α^* nor β^* has been corrupted before the completion of $(\alpha^*, bid, sid, \vec{x}_{\alpha^*}, f_{\alpha^*})$. Then, assuming that $\mathcal{ACME}$ has authenticity, the following statements hold:

(1) If α^* is the client and β^* is the server, then $\mathcal{A}$ initiated a broadcast $(\beta^*, bid, \vec{x}_{\beta^*}, f_{\beta^*})$ and α^* must have been activated to initiate a session $(\alpha^*, bid, sid, \vec{x}_{\alpha^*}, f_{\alpha^*})$ with the broadcast message if and only if $f_{\alpha^*}(\vec{x}_{\beta^*}^{(out)}) = 1 \ \wedge \ f_{\beta^*}(\vec{x}_{\alpha^*}^{(out)}) = 1$.

(2) If α^* is the server and β^* is the client, the session $(\beta^*, bid, sid, \vec{x}_{\beta^*}, f_{\beta^*})$ cannot complete with a peer session $(\alpha'^*, bid, sid, \vec{x}_{\alpha'^*}, f_{\alpha'^*})$ such that $f_{\alpha'^*}(\vec{x}_{\beta^*}^{(out)}) \neq 1 \ \vee \ f_{\beta^*}(\vec{x}_{\alpha'^*}^{(out)}) \neq 1$.

Proof. We prove the two cases separately.

(1) When α^* is activated to initialize a session $(\alpha^*, bid, sid, \vec{x}_{\alpha^*}, f_{\alpha^*})$ with a broadcast message (bid', CT_B), it decrypts the broadcast ciphertext utilizing its private attribute and policy keys. If the decryption fails, it indicates that $f_{\alpha'^*}(\vec{x}_{\beta^*}^{(out)}) \neq 1 \ \vee \ f_{\beta^*}(\vec{x}_{\alpha'^*}^{(out)}) \neq 1$. Otherwise, α^* is an intended client to obtain the broadcast messages, who checks whether $bid' = bid$ and verifies the authenticity of CT_B for further communication. Since $(\alpha^*, bid, sid, \vec{x}_{\alpha^*}, f_{\alpha^*})$ completes with β^*, it must be the case that $f_{\alpha^*}(\vec{x}_{\beta^*}^{(out)}) = 1 \ \wedge \ f_{\beta^*}(\vec{x}_{\alpha^*}^{(out)}) = 1$. Since β^* has not been corrupted before the completion with $(\alpha^*, bid, sid, \vec{x}_{\alpha^*}, f_{\alpha^*})$, it generates at most one broadcast ciphertext containing bid. Thus, if $(\alpha^*, bid, sid, \vec{x}_{\alpha^*}, f_{\alpha^*})$ completes with β^*, it must have been initialized with broadcast message output by $(\beta^*, bid, \vec{x}_{\beta^*}, f_{\beta^*})$ since this is the only message that contains a valid ACME ciphertext from β^* with the broadcast identifier bid. Otherwise, $\mathcal{A}$ can be used to break the authenticity of $\mathcal{ACME}$.

(2) If β^* is activated to initiate the session $(\beta^*, bid, \vec{x}_{\beta^*}, f_{\beta^*})$ and the session completes with a peer α'^* with $(\vec{x}_{\alpha'^*}, f_{\alpha'^*})$ such that $f_{\alpha'^*}(\vec{x}_{\beta^*}^{(out)}) \neq 1 \ \vee \ f_{\beta^*}(\vec{x}_{\alpha'^*}^{(out)}) \neq 1$. In this case, an honest β^* would never succeed to decrypt the broadcast ciphertext and then generate the ACME ciphertext CT_{β^*}. Therefore, any adversary that can cause $(\alpha^*, bid, sid, \vec{x}_{\alpha^*}, f_{\alpha^*})$ to complete with peer β^*, and have $(\beta^*, bid, sid, \vec{x}_{\beta^*}, f_{\beta^*})$ complete with peer α'^* can break the authenticity of $\mathcal{ACME}$. □

Next we consider the two possible cases and prove that the adversary's advantage in both cases is negligible.

Case 1: $\mathcal{A}$ does not compromise the broadcast session and z is not disclosed.

In this case, the security proof relies on the server's broadcast secret for the privacy of the session. A series of hybrid games are defined.

• **Hybrid** H_0: This game is the same as a real interaction with the PriSrv protocol. A random bit $b \in \{0, 1\}$ is selected. When $b = 1$, the real session key is returned as a response to the TestSession query. Otherwise, a random key from the key space is returned as the session key.

• **Hybrid** H_1: This game is the same as H_0, except that $Z^{x_2} = X_2^z$ is replaced by a random value in group G_2.

• **Hybrid** H_2: This game is the same as H_1, except that $\bar{\mathcal{S}}$ also aborts if the session $(\overline{\beta^*}, \overline{bid}, \overline{sid}, \overline{\vec{x}_{\beta^*}}, \overline{f_{\beta^*}})$ does not match $(\overline{\alpha^*}, \overline{bid}, \overline{sid}, \overline{\vec{x}_{\alpha^*}}, \overline{f_{\alpha^*}})$.

• **Hybrid** H_3: This game is the same as H_2, except that $\overline{SSK}$ is replaced by a random number from the secret session key space.

In the following, we show that each consecutive pair of hybrid games described above are computationally indistinguishable.

Claim C.1. Hybrids H_0 and H_1 are computationally indistinguishable if the DDH assumption holds in group G_2.

Proof. Let $(\overline{\alpha^*}, \overline{bid}, \overline{sid}, \overline{\vec{x}_{\alpha^*}}, \overline{f_{\alpha^*}})$ be the session and $(\overline{\beta^*}, \overline{\vec{x}_{\beta^*}}, \overline{f_{\beta^*}})$ be the peer that the adversary commits to at the beginning of the experiment. By definition, this means that $(\overline{\alpha^*}, \overline{\beta^*}, \overline{bid}, \overline{sid}, \overline{\vec{x}_{\alpha^*}}, \overline{f_{\alpha^*}}, \overline{\vec{x}_{\beta^*}}, \overline{f_{\beta^*}})$ is the public output of the test session.

Let $\mathcal{A}$ be a distinguisher between H_0 and H_1. We use $\mathcal{A}$ to build a DDH adversary $\mathcal{B}$ as below. $\mathcal{B}$ is given a DDH challenge tuple $(h^{b_1}, h^{b_2}, h^{\gamma_2})$ over group G_2, where $\gamma_2 = b_1 b_2$ or γ_2 is a random number from $\mathbb{Z}_p^*$.

At the beginning of the simulation, $\mathcal{B}$ generates anonymous credential, private attribute and policy key (in the same manner as the simulator $\overline{S}$) for each of the n parties. $\mathcal{B}$ begins the simulation of the security game for $\mathcal{A}$. In the following, we use $\overline{C} \in \{\overline{\alpha^*}, \overline{\beta^*}\}$ to denote the client and $\overline{S} \in \{\overline{\alpha^*}, \overline{\beta^*}\}$ to denote the server in the test session.

• **Server broadcast queries**. If the adversary activates a server $\overline{S}$ to initiate the test broadcast session $(\overline{S}, \overline{bid}, \overline{\vec{x}_s}, \overline{f_s})$, the simulator uses h^{b_1} from the DDH challenge instance as the semi-static DH share in the broadcast message. For other broadcast queries, $\mathcal{B}$ selects a random DH exponent z to constructs the broadcast ciphertext exactly as in the real protocol.

• **Client initialization queries**. When $\mathcal{A}$ activates a party α^* to initiate a session $(\alpha^*, bid, sid, \vec{x}_{\alpha^*}, f_{\alpha^*})$, if $(\alpha^*, bid, sid, \ \vec{x}_{\alpha^*}, f_{\alpha^*}) \neq (\overline{C}, \overline{bid}, \overline{sid}, \overline{\vec{x}_c}, \overline{f_c})$, $\mathcal{B}$ selects a random DH exponent and generates the message exactly as in the real scheme. Otherwise, if $(\alpha^*, bid, sid, \ \vec{x}_{\alpha^*}, f_{\alpha^*}) = (\overline{C}, \overline{bid}, \overline{sid}, \overline{\vec{x}_c}, \overline{f_c})$, $\mathcal{B}$ sets h^{b_2} from the DDH challenge instance to be the DH share $X_2 = h^{x_2}$ in its message. The other computation steps follow the real experiment.

• **Server response queries**. When $\mathcal{A}$ activates a server S to respond to a session $(\alpha^*, bid, sid, \vec{x}_{\alpha^*}, f_{\alpha^*})$, $\mathcal{B}$ selects a random DH exponent y and generates the message exactly as in the real scheme.

• **Client finish queries**. When a client receives a response message for session $(\alpha^*, bid, sid, \vec{x}_{\alpha^*}, f_{\alpha^*})$, if $(\alpha^*, bid, sid, \vec{x}_{\alpha^*}, \ f_{\alpha^*}) \neq (\overline{C}, \overline{bid}, \overline{sid}, \overline{\vec{x}_c}, \overline{f_c})$, $\mathcal{B}$ constructs the outputs as in the real scheme (this is feasible since $\mathcal{B}$ selects the client's ephemeral DH share in this case). Otherwise, if $(\alpha^*, bid, sid, \vec{x}_{\alpha^*}, f_{\alpha^*}) = (\overline{C}, \overline{bid}, \overline{sid}, \overline{\vec{x}_c}, \overline{f_c})$, $\mathcal{B}$ runs the other computation steps following the real experiment except that it sets $SSK = H(X_1^y, h^{\gamma_2})$, where $X_1^y = Y^{x_1}$ (generated by $\mathcal{B}$) and h^{γ_2} is from the DDH challenge instance.

• RevealState **and** RevealKey **queries**. These are handled exactly as in H_0.

• Corrupt **queries**. If $\mathcal{A}$ corrupts a party U, $\mathcal{B}$ sends the anonymous credential and private attribute/policy keys of U to $\mathcal{A}$, as well as ephemeral secrets in the local storage of U.

$\mathcal{B}$ perfectly simulates H_0 if $\gamma_2 = b_1 b_2$, and $\mathcal{B}$ simulates H_1 if γ_2 is a random number. Then, if $\mathcal{A}$ can distinguish H_0 from H_1, $\mathcal{B}$ can succeed in the DDH game on G_2 with the same advantage. □

Claim C.2. Hybrids H_1 and H_2 are computationally indistinguishable if the ACME algorithm is private and MAC is unforgeable.

Proof. If an adversary $\mathcal{A}$ outputs a session $(\alpha^*, bid, sid, \vec{x}_{\alpha^*}, f_{\alpha^*})$ in the TestSession query, there must be a partner instance $(\beta^*, bid, sid, \vec{x}_{\beta^*}, f_{\beta^*})$. Otherwise, we can make use of the adversary $\mathcal{A}$ to break the privacy of $\mathcal{ACME}$ or the unforgeability of MAC.

We define an adversary $\mathcal{B}_0$ such that in the TestSession query $\mathcal{B}_0$ outputs a session $(\alpha^*, bid, sid, \vec{x}_{\alpha^*}, f_{\alpha^*})$, which has a matching session $(\beta^*, bid, sid, \vec{x}_{\beta^*}, f_{\beta^*})$. Given an adversary $\mathcal{A}$ against the PriSrv protocol in the security game, we build an adversary $\mathcal{B}_0$ as follows.

Adversary $\mathcal{B}_0$ answers all queries made by $\mathcal{A}$ using its own oracles. If $\mathcal{A}$ outputs an instance $(\alpha^*, bid, sid, \vec{x}_{\alpha^*}, f_{\alpha^*})$ that has no matching session, $\mathcal{B}_0$ aborts without any output. Otherwise, adversary $\mathcal{B}_0$ outputs $(\alpha^*, bid, sid, \vec{x}_{\alpha^*}, f_{\alpha^*})$ in the TestSession query, and returns to adversary $\mathcal{A}$ the response it receives.

Let E denote the event that the instance $(\alpha^*, bid, sid, \vec{x}_{\alpha^*}, f_{\alpha^*})$ in the TestSession query output by adversary $\mathcal{A}$ does not have a matching session. If event E does not happen, $\mathcal{B}_0$ and adversary $\mathcal{A}$ are the same. Otherwise, we can construct an encryption-aided forger $\mathcal{B}_1$ who aims to produce a forgery $\mathcal{MAC}.\mathsf{MAC}(K^*, M)$ for a secret key K^* that is encapsulated in an ACME ciphertext CT^* [58].

$\mathcal{B}_1$ is given pp, mpk and access to an oracle $\mathcal{O}_{\mathsf{Issue}}(\cdot)$ which creates anonymous credential for user U, an oracle $\mathcal{O}_{\mathsf{DKGen}}(\cdot)$ which creates attribute decryption key for $\vec{x}$, an oracle $\mathcal{O}_{\mathsf{PolGen}}(\cdot)$ which creates policy decryption key for f, an oracle $\mathcal{O}_{\mathsf{Dec}}(\cdot)$ which decrypts ciphertexts. Assume that adversary $\mathcal{A}$ performs at most q_I activations of parties with an incoming message.

Forger $\mathcal{B}_1$ randomly chooses $\ell \leftarrow [1, q_I]$ and simulates the security game for adversary $\mathcal{A}$ in the following cases.

• If adversary $\mathcal{A}$ does not make a TestSession query with an activation of α^*, forger $\mathcal{B}_1$ aborts.

• If β^* is not the matching session of α^*, forger $\mathcal{B}_1$ aborts.

• If $(\alpha^*, bid, sid, \vec{x}_{\alpha^*}, f_{\alpha^*})$ is not the ℓ-th activation, forger $\mathcal{B}_1$ aborts.

• If adversary $\mathcal{A}$ makes a Corrupt query on β^* or any partner β with $(\vec{x}_\beta, f_\beta)$ satisfying $f_{\alpha^*}(\vec{x}_\beta^{(out)}) = 1 \wedge f_\beta(\vec{x}_{\alpha^*}^{(out)}) = 1$, before the session has expired, forger $\mathcal{B}_1$ aborts.

• In the ℓ-th activation, $\mathcal{B}_1$ uses $(\mathsf{DK}_{\vec{x}_{\alpha^*}}, \mathsf{DK}_{f_{\alpha^*}})$ to derive the broadcast message $MSG_B^* \leftarrow \mathcal{ACME}.\mathsf{Dec}(\mathsf{DK}_{\vec{x}_{\alpha^*}}, \mathsf{DK}_{f_{\alpha^*}}, \ \mathsf{CT}_{\beta^*})$, where $MSG_B = (bid, Z, \cdots, K_{\alpha^*})$. $\mathcal{B}_1$ generates the ephemeral DH shares (X_1, X_2) for $(\alpha^*, bid, sid, \vec{x}_{\alpha^*}, f_{\alpha^*})$. $\mathcal{B}_1$ sets $M_{\alpha^*} = (\text{“}\alpha^* \rightarrow \beta^*\text{”}, bid, sid, X_1, X_2, Z)$ and creates the tag σ_{α^*} using K_{α^*} on M_{α^*}. $\mathcal{B}_1$ asks its challenger to return

$$\mathsf{CT}_{\alpha^*} = \mathcal{ACME}.\mathsf{Enc}(\mathsf{cred}_{\alpha^*}, \vec{x}_{\alpha^*}, f_{\alpha^*}, MSG_{\alpha^*})$$

where $MSG_{\alpha^*} = (K^*, M_{\alpha^*})$ and K^* is chosen by $\mathcal{B}_1$'s challenger. Forger $\mathcal{B}_1$ sets $\mathsf{CT}^* = \mathsf{CT}_{\alpha^*}$.

• If adversary $\mathcal{A}$ sends $(\mathsf{CT}, \cdots)$ to β^* where $\mathsf{CT} \neq \mathsf{CT}^*$, forger $\mathcal{B}_1$ makes a query to its decryption oracle $\mathcal{O}_{\mathsf{Dec}}(\cdot)$ on input CT, and proceeds as usual after getting the response from $\mathcal{O}_{\mathsf{Dec}}(\cdot)$.

• If adversary $\mathcal{A}$ sends (CT^*, M) to β^*, forger $\mathcal{B}_1$ issues a query to its oracle $\mathcal{O}_{\mathsf{MAC}}$ to generate the tag σ^* with regards to K^* and M. The restriction is that M does not contain (bid, sid) of challenge session.

• When adversary $\mathcal{A}$ sends the tag σ^* to the ℓ-th activation, forger $\mathcal{B}_1$ outputs the tag σ^* and the corresponding message as its forgery.

Therefore, we have $\epsilon = Pr[\mathcal{B}_1 \text{ succeeds}] \geq \frac{1}{q_I} Pr[E]$.

Given a forger $\mathcal{B}_1$, we now construct another adversary $\mathcal{B}_2$ against the anonymous credential matchmaking encryption scheme $\mathcal{ACME}$ in the security game, which is given the public parameter pp and has access to the attribute/policy decryption key generation and decryption oracle. When forger $\mathcal{B}_1$ asks for a challenger with input participant α^*, adversary $\mathcal{B}_2$ randomly chooses two keys K_0 and K_1, and asks its challenger with inputs (K_0, M_{α^*}) and (K_1, M_{α^*}). After obtaining the challenge CT^* (with respect to K_0 or K_1), adversary $\mathcal{B}_2$ sets CT^* as forger $\mathcal{B}_1$'s challenge. When forger $\mathcal{B}_1$ makes a query with a ciphertext $\mathsf{CT} \neq \mathsf{CT}^*$, adversary $\mathcal{B}_2$ makes a decryption query with input CT to its challenger. When forger $\mathcal{B}_1$ makes an $\mathcal{O}_{\mathsf{MAC}}$ query on a message M, adversary $\mathcal{B}_2$ returns $\mathcal{MAC}.\mathsf{MAC}(K_0, M)$ to forger $\mathcal{B}_1$. Finally, if forger $\mathcal{B}_1$ successfully makes a forgery $\mathcal{MAC}.\mathsf{MAC}(K_0, M_{\alpha^*})$, $\mathcal{B}_2$ outputs 0 meaning that CT^* is an encryption of (K_0, M_{α^*}). Otherwise, if forger $\mathcal{B}_1$ fails to make a forgery, adversary $\mathcal{B}_2$ outputs 1, meaning that CT^* is an encryption of (K_1, M_{α^*}). Hence, we have

$$\begin{aligned}
& \mathbf{Adv}^{\mathsf{ACME}}_{\mathcal{B}_2}(\lambda) \\
= \; & Pr[\mathcal{B}_2 \text{ outputs } 0|b=0] \cdot Pr[b=0] + \\
& Pr[\mathcal{B}_2 \text{ outputs } 0|b=1] \cdot Pr[b=1] - \frac{1}{2} \\
= \; & \frac{1}{2} Pr[\mathcal{B}_1 \text{ succeeds}|b=0] + \\
& \frac{1}{2}(1 - \frac{1}{2} Pr[\mathcal{B}_1 \text{ succeeds}|b=1]) - \frac{1}{2} \\
= \; & \frac{1}{2}(Pr[\mathcal{B}_1 \text{ succeeds}|b=0] - Pr[\mathcal{B}_1 \text{ succeeds}|b=1]) \\
= \; & \frac{1}{2}(\epsilon - \mathbf{Adv}^{\mathsf{MAC}}_{\mathcal{B}_1}(\lambda)).
\end{aligned}$$

The last line of the above equation is concluded from when $b = 0$, forger $\mathcal{B}_1$ is in the forgery game, and when $b = 1$, forger $\mathcal{B}_1$ is in the random message attack game.

Therefore, Hybrids H_1 and H_2 are computationally indistinguishable. □

Claim C.3. Hybrids H_2 and H_3 are computationally indistinguishable when the hash function H is a random oracle.

Proof. Since in H_2 we replaced Z^{x_2} with a random value h^{γ_2} from G_2, the probability that the adversary can make a hash query $H(Y^{x_1}, h^{\gamma_2})$ is negligible. When (Y^{x_1}, h^{γ_2}) is not queried, its hash value (i.e., the session key) is an unknown random value to the adversary, same as in H_3. The claim follows. □

Case 2: $\mathcal{A}$ has compromised z.

In this case, we rely on the ephemeral DH share y of the server to ensure confidentiality of the session key. The security proof is quite similar to that of Case 1. The hybrid experiments are described below.

- **Hybrid** H_0: This game is the same as a real interaction with the PriSrv protocol.
- **Hybrid** H_1: This game is the same as in Case 1, except that $Y^{x_1} = X_1^y$ is replaced by a random value in group G_1.
- **Hybrid** H_2: This game is the same as in Case 1.
- **Hybrid** H_3: This game is the same as in Case 1.

It suffices to prove that the hybrids H_0 and H_1 are computationally indistinguishable.

Claim C.4. Hybrids H_0 and H_1 are computationally indistinguishable if the DDH assumption holds in group G_1.

Proof. The proof follows the same arguments as in Case 1 except that the DDH tuple $(h^z, h^{x_2}, h^{\gamma_2})$ on group G_2 used in the proof of the Case 1 is replaced by the DDH tuple $(g^y, g^{x_1}, g^{\gamma_1})$ on group G_1. □

Integrating the above proofs for the two cases, we conclude that PriSrv realizes secure service discovery with bilateral control. □

Lemma C.2. (**Client anonymity**) PriSrv protocol satisfies client anonymity assuming the ACME scheme is private.

Proof. We define a simulator $\mathcal{S}$ that simulates the challenger for the adversary $\mathcal{A}$ in the client anonymity security game.

At the beginning of the simulation, $\mathcal{A}$ submits a set of identities (with attributes and policies) $\{(uid_j, \vec{x}_j, f_j)\}_{j=1}^n$ for parties $(P_1, \cdots, P_n)$, and two challenge tuples $(uid^*_{i_0}, \vec{x}^*_{i_0}, f^*_{i_0})$ and $(uid^*_{i_1}, \vec{x}^*_{i_1}, f^*_{i_1})$, which are distinct from $\{(uid_j, \vec{x}_j, f_j)\}_{j=1}^n$.

We define a series of hybrid experiments. During the protocol execution, the simulator responds to the adversary's queries according to these hybrid experiments.

- Hybrid H_1: This is the real experiment Exp_0, where the simulator responds to adversary's queries as described in Exp_0.
- Hybrid H_2: This is the real experiment Exp_1.

Next, we prove that Hybrids H_1 and H_2 are computationally indistinguishable if the underlying ACME is private.

Let q be an upper bound on the number of sessions, where $\mathcal{A}$ activates the test party P_T as the responder (the client). We define a sequence of $q+1$ hybrid experiments $\mathsf{H}_{1,0}, \cdots, \mathsf{H}_{1,q}$, where hybrid experiment $\mathsf{H}_{1,i}$ is defined as follows: $\mathsf{H}_{1,i}$ is same as H_1 except that for the first i times when P_T is activated as the responder (the client) of the broadcast ciphertext, P_T is instantiated using the credential of $(uid^*_{i_1}, \vec{x}^*_{i_1}, f^*_{i_1})$ for generating the response message. In all subsequent times, P_T P_T is instantiated using the credential of $(uid^*_{i_0}, \vec{x}^*_{i_0}, f^*_{i_0})$.

By construction, $H_1 \equiv H_{1,0}$ and $H_2 \equiv H_{1,q}$. We prove that for all $i \in [q]$, hybrid $\mathsf{H}_{1,i-1}$ and $\mathsf{H}_{1,i}$ are computationally indistinguishable assuming that the ACME scheme is private. Suppose $\mathcal{A}$ is able to distinguish $\mathsf{H}_{1,i-1}$ from $\mathsf{H}_{1,i}$, we use $\mathcal{A}$ to construct an adversary $\mathcal{B}$ against ACME in the security game.

First, $\mathcal{B}$ is given the public parameters mpk of the ACME scheme. Then, $\mathcal{B}$ begins running $\mathcal{A}$ and obtains set of identities (with attributes and policies) $\{(uid_j, \vec{x}_j, f_j)\}_{j=1}^n$ for parties $(P_1, \cdots, P_n)$, and two challenge tuples $(uid^*_{i_0}, \vec{x}^*_{i_0}, f^*_{i_0})$ and $(uid^*_{i_1}, \vec{x}^*_{i_1}, f^*_{i_1})$, which are distinct from $\{(uid_j, \vec{x}_j, f_j)\}_{j=1}^n$.

$\mathcal{B}$ simulates the setup procedure in H_1 by creating anonymous credentials, private attribute/policy keys for each party. Then, $\mathcal{B}$ sends mpk to $\mathcal{A}$ and begins simulating the protocol execution experiment for $\mathcal{A}$.

• **Server broadcast queries**. These are handled exactly as in H_1 and H_2.

• **Client initialization queries**. When $\mathcal{A}$ activates a client C to respond to a broadcast $(S, bid, \vec{x}_s, f_s)$, if $C \neq P_T$, algorithm $\mathcal{B}$ simulates the response as in the real scheme. If $C = P_T$, then let ℓ be the number of times $\mathcal{A}$ has activated P_T to respond to a broadcast. $\mathcal{B}$ queries the ACME key generation or decryption oracle to obtain a decrypted broadcast message, and performs the checks on the decrypted broadcast message. $\mathcal{B}$ then proceeds as below.

- If $\ell < i-1$, $\mathcal{B}$ constructs the response message as described in H_2, that is using $(uid^*_{i_1}, \vec{x}^*_{i_1}, f^*_{i_1})$ in client's response message.

- If $\ell \geq i$, $\mathcal{B}$ constructs the response message as described in H_1, that is using $(uid^*_{i_0}, \vec{x}^*_{i_0}, f^*_{i_0})$ in client's response message.

- If $\ell = i-1$, $\mathcal{B}$ selects random DH shares $x_1, x_2 \xleftarrow{\$} \mathbb{Z}_p^*$, and generates a MAC key K_s. It submits $(\mathsf{cred}_{i_0}, \vec{x}^*_{i_0}, f^*_{i_0}, MSG_c)$ and $(\mathsf{cred}_{i_1}, \vec{x}^*_{i_1}, f^*_{i_1}, MSG_c)$ to the ACME challenger, where $MSG_c = (K_s, M_c)$, $M_c = (\text{“}C \to S\text{”}, bid, sid, X_1 = g^{x_1}, X_2 = h^{x_2}, Z)$, cred_{i_b} is the anonymous credential for $uid^*_{i_b}$, $b \in \{0,1\}$. Then, it receives a ciphertext $\overline{CT}_c$ from the challenger. $\mathcal{B}$ runs MAC scheme to obtain σ_c from K_c and M_c. $\mathcal{B}$ outputs the response $(bid, sid, \sigma_c, \overline{CT}_c)$.

• **Server response queries**. When $\mathcal{A}$ delivers a message to server, $\mathcal{B}$ responds as below.

- $\mathcal{B}$ parses $\mathcal{A}$'s message as $(bid, sid, \sigma_c, CT_c)$.

- If $\mathcal{B}$ is in the pre-challenge phase, or if $\mathcal{B}$ is in the post-challenge phase, and either $CT_c \neq \overline{CT}_c$ or B is allowed to obtain the decryption key for CT_c, $\mathcal{B}$ queries the ACME decryption or key generation oracle to decrypt CT_c.

If $\mathcal{B}$ is in the post-challenge phase and $CT_c = \overline{CT}_c$, $\mathcal{B}$ simulates the response by using the $MSG_c = (K_s, M_c)$ it has chosen for generating $\overline{CT}_c$.

• **Client finish queries**. These are handled exactly as in H_1 and H_2. They are independent of the ACME parameters.

• **$\mathsf{RevealState}$ and $\mathsf{RevealKey}$ queries**. These are handled exactly as in H_1 and H_2.

• **$\mathsf{Corrupt}$ queries**. If $\mathcal{A}$ corrupts a party $P \neq P_T$, $\mathcal{B}$ queries the ACME challenger for the private attribute/policy keys for P, and returns the keys as well as ephemeral secrets in the local storage of P to $\mathcal{A}$.

At the end of the game, $\mathcal{A}$ outputs a guess for whether it is in H_1 or H_2. $\mathcal{B}$ forwards the guess to its security game.

To complete the proof, we show that $\mathcal{B}$ is an elegible ACME adversary in the privacy security game since $\mathcal{B}$ does not need to request its challenger to decrypt the challenge ciphertext or return secret keys that can decrypt the challenge ciphertext.

By construction, if $\mathcal{B}$ receives an encryption of $(uid^*_{i_0}, \vec{x}^*_{i_0}, f^*_{i_0})$ from the ACME challenger, then it has correctly simulated the client's response queries according to the specification of hybrid $\mathsf{H}_{1,i-1}$ for $\mathcal{A}$. If it receives an encryption of $(uid^*_{i_1}, \vec{x}^*_{i_1}, f^*_{i_1})$ from the ACME challenger, then it has correctly simulated the client's response queries according to the specification of hybrid $\mathsf{H}_{1,i}$ for $\mathcal{A}$.

Hence, if the ACME is private, then H_1 and H_2 are computationally indistinguishable.

Therefore, we have proved that PriSrv satisfies client anonymity. □

Lemma C.3. (**Server anonymity**) PriSrv protocol satisfies server anonymity assuming the ACME scheme is private.

Proof. This proof is similar to the proof of client anonymity (in Lemma F.7) except that P_T initiates broadcast as a server in the simulation. We omit the details of the proof for briefty. □