Enabling Efficient and Distributed Access Control for Pervasive Edge Computing Services

Lingshuang Liu, *Student Member, IEEE*, Cheng Huang, *Member, IEEE*, Dan Zhu, *Member, IEEE*, Dongxiao Liu, *Member, IEEE*, Jianbing Ni, *Senior Member, IEEE*, and Xuemin Shen, *Fellow, IEEE*

***Abstract*—In this paper, we propose an efficient and distributed service access control framework (E-DAC) in the pervasive edge computing (PEC) environment, where the resources of peer devices at the network edge are integrated to provide latency-sensitive computing services to the nearby devices on behalf of edge servers. E-DAC addresses the challenge of efficient and distributed service access control, comprising edge service authorization, service access authorization, and mutual authentication between edge servers and edge devices. In dong so, E-DAC first extends a key-aggregate cryptosystem to enable batch service authorization, in which a service provider can aggregate the authorization keys of different services to produce a constant-size aggregate key for an edge server. Second, E-DAC enables users to acquire authorization from the service provider for service access on edge servers by using efficient secret sharing. Third, edge servers and users can authenticate with each other without interacting with a centralized server, while enabling secure zero-round trip communication, so that the service data is protected and the communication bandwidth cost is low. In addition, the service provider is capable of efficiently revoking the authorization of the dropout or compromised edge servers or users in response to the dynamics of the PEC environment. Finally, we prove the security of service access control in E-DAC, including unforgeability of service authorization and confidentiality of service data, and conduct extensive analysis and experiments to demonstrate that E-DAC is highly computational and communication-efficient on service authorization, authentication, and revocation.**

***Index Terms*—Access control, authorization, key aggregation, mutual authentication, pervasive edge computing.**

I. Introduction

EDGE computing has gained significant attention for its ability to bring computation and storage closer to data sources, reducing network bandwidth usage and enabling latency-sensitive services, such as autonomous driving, industrial automation, and augmented/virtual reality (AR/VR) applications [2]. With the increasing power of end-user devices, edge computing has evolved from pre-deployed infrastructures like base stations and gateways to pervasive edge computing (PEC) [3], leveraging the resources of nearby peer devices such as smart phones, tablets, vehicles, and unmanned aerial vehicles (UAVs) to fulfill low-latency computation tasks for users. PEC offers real-time and flexible service composition based on the dynamic resources of peer devices, enabling rapid response to events [4]. For example, PEC allows prompt processing of video feed data from surveillance cameras on roads by nearby devices, eliminating the need for transmitting to a remote cloud server and enabling quick response to criminal activities and threats. Similarly, in autonomous driving, PEC facilitates real-time processing of on-board camera footage for immediate lane detection and alerts. In short, PEC enhances traditional centralized applications by enabling localized computation and fast decision-making at the edge, improving overall efficiency and responsiveness.

Due to the mobility of peer devices, the resources in PEC exhibit high dynamism. This means that the devices can dynamically join or leave the PEC resource pool that offers computing services to its nearby user devices, and their roles can switch between being an edge server providing services and a user accessing services. This dynamic nature presents the following new and intensified challenges for access control between users and edge servers: 1) How to efficiently authorize an edge server to provide a particular service or a set of services for users, and how to efficiently authorize a user to access a particular service offered by the edge server in highly dynamic PEC environment? Meanwhile, due to the mobility of devices, the short-lived connections between users and edge servers make an involving cloud server in the authentication process overwhelming. Traditional access control frameworks [5], [6] utilized in cloud computing, which rely on an "always-online" server for authentication, are not suitable for the distributed PEC environment due to the significant network latency. Therefore, there is an urgent need for a computation and communication-efficient solution that handles authorization and authentication for both edge servers and users to address these challenges in the distributed scenario. Additionally, considering the dynamic nature of PEC, i.e., both users and edge servers may join, leave, or turnover, quick revocation mechanisms for both users and

Manuscript received 1 September 2023; revised 13 February 2024; accepted 23 April 2024. Date of publication 2 May 2024; date of current version 5 November 2024. An earlier version of this paper was presented in part at the 2022 IEEE Global Communications Conference (GLOBECOM) [DOI: 10.1109/GLOBECOM48099.2022.10000715]. Recommended for acceptance by L. Guo. *(Corresponding Author: Cheng Huang.)*

Lingshuang Liu and Xuemin Shen are with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada (e-mail: l325liu@uwaterloo.ca; sshen@uwaterloo.ca).

Cheng Huang is with the School of Computer Science, Fudan University, Shanghai 200438, China (e-mail: chuang@fudan.edu.cn).

Dan Zhu is with the School of Cybersecurity, Northwestern Polytechnical University, Xi'an, Shaanxi 710060, China (e-mail: zhudan@nwpu.edu.cn).

Dongxiao Liu is with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, Sichuan 610054, China (e-mail: dongxiao.liu@uestc.edu.cn).

Jianbing Ni is with the Department of Electrical and Computer Engineering and Ingenuity Labs Research Institute, Queen's University, Kingston, Ontario K7L 3N6, Canada (e-mail: jianbing.ni@queensu.ca).

Digital Object Identifier 10.1109/TMC.2024.3395388

edge servers must be incorporated into the efficient distributed access control framework to ensure secure and timely control over service access of users and edge servers.

The popular distributed access control methods in edge computing are role-based access control [7], [8] and attribute-based encryption (ABE) based access control [9], [10]. These methods allow the service provider to be off-line after authorizing services to the edge servers. However, these access control mechanisms are typically static [8], making them challenging to revoke access for edge servers or users. Consequently, they are not well-suited for the dynamic PEC environment. With the advent of blockchain technology, several blockchain-based decentralized access control mechanisms [11], [12], [13] have been proposed to reduce dependency on cloud servers and mitigate the risk of single points of failure. However, these mechanisms face inherent challenges related to consensus, cost, and scalability [14]. Moreover, most of these solutions primarily focus on securing static content services [15], [16], where edge servers provide users with content allocated by a content service provider. In contrast, PEC aims to deliver real-time computing services by leveraging user input data and returning processing results promptly. To address the requirements of dynamics and low latency in PEC, Dougherty et al. [17] proposed a distributed access control framework called APECS for PEC services, based on multi-authority ABE. APECS supports service authorization and authentication between users and edge servers without relying on an "always-online" cloud server. It also enables the revocation of edge servers to handle turnover. Nonetheless, in APECS, 1) the size of the service authorization message increases linearly with the number of services and base stations, and 2) revoking a compromised edge server requires re-keying all edge servers under the same base station.

Contributions: We study highly efficient solutions for distributed authorization and authentication and propose an efficient and distributed access control framework (E-DAC), designed specifically for dynamic PEC services. It enables revocable service authorization for edge servers and users, while facilitating mutual authentication between them without relying on an "always-online" centralized authentication server by extending the key-aggregate cryptosystem (KAC) [18] and secret sharing-based broadcast encryption [19]. The main contributions of E-DAC are summarized in four-fold:

- E-DAC enables efficient service authorization for edge servers based on KAC, which supports flexible decryption delegation of any subset of the ciphertext with a constant-size decryption key. We expend KAC to achieve efficient service authorization for registered edge servers. This extension allows the service provider to authorize an edge server to offer a set of services using a scalable aggregate authorization key. By utilizing this key, an edge server can prove to the users that it is authorized to provide the specific service they seek to access. The service authorization is efficient as the size of the service authorization key for an edge server is constant, non-linear with the number of authorized services.
- E-DAC achieves service access authorization for users based on efficient secret sharing. Secret shares of a service key are created by the service provider for service access authorization to users. These shares are distributed to the subscribed users and the edge server to enable the edge server to verify the authorization to users. The users can randomize secret shares and demonstrate the ownership of secret shares to authenticate to the edge server for service access. The service access authorization is efficient on computation because of low computational overhead on secret share generation and reconstruction.
- E-DAC supports efficient mutual authentication between an edge server and a user device. To save the communication rounds between the edge server and the user device, secret zero-round trip (0-RTT) mutual authentication is achieved to integrate service message and authentication request into a single message, which can significantly reduce the communication costs and protect the service data of the authenticated user. The correct decryption of service data with the service key and the aggregate authorization key ensures mutual authentication between the user and the edge server.
- E-DAC provides efficient revocation of an edge server by exposing its associated secret share of the service key. This revocation process incurs no additional cost for other edge servers. Additionally, revoking a user entails adding their identity information to a revocation list, ensuring they are denied access to the PEC services.

The remainder of this paper is organized as follows. In Section II, we review the existing work on service access control in edge computing, and in Section III, we introduce the system model and present the security model. Section IV describes the detailed E-DAC, highlighting its components and functionalities and Section V gives the security proof of E-DAC, followed by performance evaluation results in Section VI. Finally, we summarize the paper in Section VII.

II. Related Work

Secure edge computing has gained increasing attention due to the proliferation of connected devices and the growing complexity of networks, creating a larger attack surface for cybercriminals. While the typical security and privacy threats have been studied [20], [21], the corresponding solutions are still being explored [22], [23].

Role-based access control is a simple and manageable approach to assigning permissions to users based on their role within an organization. Hou et al. [7] designed a fine-grained access control mechanism to ensure data security in mobile edge computing. The role-based access control mechanism integrates with a dynamic fine-grained trusted user grouping scheme based on attributes to assign roles to users according to their group credibility. However, role-based access control has low scalability and dynamism because permission assigning is needed for each role. Attribute-based access control [24] is a highly adaptable and customizable way to implement access control policies based on a wide range of user attributes, making it suitable for distributed or rapidly changing environments. Li et al. [25] proposed a verifiable ciphertext-policy ABE-based

access control scheme for edge computing-assisted Internet of Things (IoT). The desirable feature of outsourced decryption is supported to mitigate the computational cost of data users with limited resources. Feng et al. [26] designed ABE with parallel outsourced decryption based on Spark and MapReduce to further improve computational efficiency for service access authentication. The extension of attribute-based access control includes the protection of access control policy proposed by Li et al. [27], utilizing ciphertext-policy ABE with privacy to hide the access control structure of service access. Additionally, multi-authority access authorization introduced by Zhang et al. [28] achieves multi-attribute authorities in generating multi-attribute-based private keys for users based on non-interactive multi-authority ABE. Nevertheless, attribute-based access control is typically static, making it challenging to revoke access for edge servers or users. Li et al.'s ABE in [25] can support attribute revocation, which is conducted on the group of users with specific attributes, not designed for particular users.

Blockchain [29], [30] has been integrated into edge computing to enhance reliable user management and secure data access. For instance, blockchain is employed to develop an authentication and authorization scheme, fostering effective trust between users and providing resistance against single points of failure in edge computing [31]. Saha et al. [32] proposed a consortium blockchain-enabled access control scheme, achieving mutual authentication among smart devices and gateway nodes, as well as between gateway nodes and respective edge servers. Key management is conducted among edge servers and associated cloud servers to establish secret keys for secure communication. While both works focus on blockchain-based methods for user identity management, blockchain is also utilized for video sharing service subscriptions in vehicular edge computing [33]. Permissioned blockchain and smart contracts record access policies and subscription events, enabling user self-certification and event traceability. Despite offering a unique method for user identity and service management, blockchain-based access control faces challenges in infrastructure design, user scalability, and effective consensus.

Biometric features [34], [35] have emerged as key elements for authentication and access control. This approach assigns access rights based on specific biometric information, such as fingerprints, face, iris, DNA, or palm prints. Authentication and access control involve two stages: enrollment and verification. Initially, users' biometric templates are enrolled in the system, securely stored. Subsequently, a captured biometric sample is compared with the enrolled template to authenticate the user. While biometric features are unique and effective for authentication, their immutability poses a serious threat if samples are leaked, stored on smart cards, or servers.

Dougherty et al. [17] proposed a distributed access control framework, APECS, ensuring access control and data confidentiality for PEC services. It is based on token-based authorization and multi-authority ABE, enabling secure asynchronous access control without requiring an "always-on" service provider. However, it faces challenges of privacy leakage and inefficiency in service authorization. To address privacy leakage, Zhang et al. [36] proposed an anonymous and auditable distributed access control framework based on conditional anonymous authentication and auditable multi-authority ABE. It allows users to enjoy edge computing services anonymously if impeccable but lose identity privacy once behaving maliciously. In our preliminary work [1], we proposed an efficient and distributed service access control framework, extending the key-aggregate cryptosystem for batch service authorization. This paper extends our preliminary work to enhance its security and efficiency from the following aspects:

- We enhance the security of the preliminary work by addressing the security risk posed by edge servers that may share their own authorization keys with each other. Specifically, we modify the method for generating the authorization keys during edge registration, linking them to the public keys of the edge servers. If the private key of an edge server is secure, no one can impersonate the edge server to provide services or obtain its authorization key. This security enhancement is significant because it prevents an edge server from offering unauthorized services by obtaining a valid authorization key from another edge server.
- We improve the security of the preliminary work by preventing attacks from malicious users who share their partial service access credentials with others. The service key varies for each service, and the service type, user's identity, and expiration time are integrated to generate a secret share, preventing malicious users from colluding to recover the service key.
- We introduce the detailed security models of E-DAC, encompassing the unforgeability of edge service credentials, the unforgeability of service access credentials, and the semantic security of service data. Additionally, we provide detailed security proofs for E-DAC. These three properties can be reduced to well-known hard mathematical assumptions, including the Computational Diffie-Hellman (CDH) assumption, the Decisional Bilinear Diffie Hellman (DBDH) assumption, and the Decisional Bilinear Diffie–Hellman Exponent (BDHE) assumption. Provable security is crucial because it provides theoretical guarantees for the security of the proposed E-DAC.

III. System and Security Models

In this section, we present the system and security models of distributed access control in PEC.

A. System Model

The system model of dynamic PEC services is depicted in Fig. 1, which contains three types of parties: service providers, edge servers, and users.

- *Service Provider:* A service provider operates multiple cloud servers to offer a range of computing and storage services to its users. These services can be classified as either static or dynamic. Dynamic services require input data from users, such as videos, audio, or images, while static services, like content provision, do not require user input. In our framework, we assume that the service provider

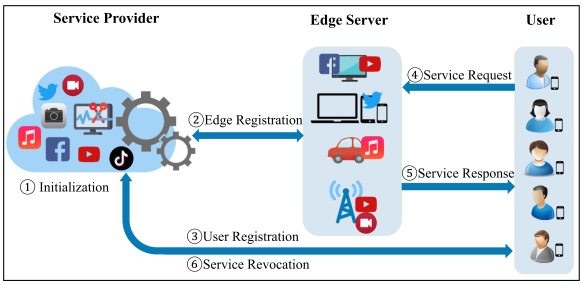


Fig. 1. System model.

primarily offers dynamic services to registered users. The service provider has the capability to authorize edge servers to deliver its services at the network edge exclusively to registered users. User registration is managed by the service provider, ensuring that only registered users can access the services on the edge servers. Moreover, the service provider has the authority to revoke service access of an edge server or a registered user if any misbehavior is detected.

- *Edge Server:* The edge server plays a crucial role by delivering the delegated services to registered users at the network edge. These edge servers can take the form of pre-deployed edge computing infrastructure, such as base stations and gateways. Alternatively, they can be end-user devices, such as smartphones, vehicles, drones, or IoT devices, that contribute their computing resources to the service pool for offering services.
- *User:* A user can register for the desired services through the service provider and subsequently gain access to the services on nearby edge servers or remote servers.

The service provider manages the registration and revocation of edge servers and users. After registration, the users and edge servers achieve 0-RTT mutual authentication with each other before the users accessing the services on the edge servers. According to these procedures, the distributed access control in PEC consists of six phases, namely, initialization, edge registration, user registration, service request, service response, and service revocation.

Initialization (INit): The service provider executes this algorithm to bootstrap the whole system to provide a set of services S. With the security parameter λ, the algorithm outputs the system parameter $\mathcal{P}$. The service provider chooses the master secret key msk and computes the public key pk for service initialization (Step ①).

Edge Registration (EReg): The edge server with an identity ID_e registers a set of services $S_e \subseteq S$ from the service provider. With the input of $(\mathcal{P}, msk, pk, ID_e, S_e)$, the service provider generates an edge service credential $Cred_s$ and returns the credential to the edge server for service authorization (Step ②).

User Registration (UReg): The user with an identity ID_u subscribes a set of services $S_u \subseteq S$ from the service provider. With the input of $(\mathcal{P}, msk, pk, ID_u, S_u)$, the service provider generates a service access credential $Cred_u$, and returns it to the user for service authorization (Step ③).

Service Request (SReq): To request a service $i \subseteq S_u$, with the input of $(\mathcal{P}, pk, i, m, ID_u, Cred_u)$, where m is the service data, the user generates a service request Req, containing the ciphertext of m, C, and sends the request to the network edge nodes (Step ④).

Service Response (SRes): With the input of $(\mathcal{P}, pk, Cred_s, Req)$, an edge server who receives the service request Req enforces implicit mutual authentication by using $Cred_s$ to decrypt C in Req. If the authentication (decryption) fails, the algorithm outputs failure; otherwise, the algorithm outputs m, thus the edge service is allowed to access the service data to respond the service (Step ⑤).

Service Revocation (SRev): If an authorized edge server ID_e is compromised, the service provider revokes its service privilege with the input of $(\mathcal{P}, pk, ID_e, Cred_s)$; If an authorized user ID_u is compromised, the service provider revokes its service privilege with the input of $(\mathcal{P}, pk, ID_u, Cred_u)$.

The distributed access control in PEC must possess *Correctness*, which means that 1) an honest registered user can produce a valid service request for a service with her service access credential to authenticate to an honest edge server if it has registered that service, and 2) an honest registered edge server can use its valid edge service credential to authenticate to and provide services for the user, if it has registered that service. Therefore, *Correctness* of distributed access control in PEC is maintained under the conditions that 1) both user and edge server have registered the service, and 2) the credentials of both user and edge server are valid, that is, the credentials are unexpired, unrevoked, and correctly issued by the service provider.

B. Security Threats

The service provider provides its services, so it prefers to maintain good reputation and attract more users, rather than deviating from the committed service policies in the terms of services. So we assume that the service provider honestly serves its customers (including edge servers and users) for monetary benefits. Also, the basic security countermeasures, including firewalls, intrusion prevention systems, and virus detection systems, are deployed on the cloud servers to prevent outsider attackers from compromising them. However, it is hard to guarantee that all users or edge servers are completely honest. They may have the following misbehavior to disrupt the services.

- *Threats from Users:* We assume that users are not fully trusted in the purpose of accessing services without valid access credentials. (a) Unauthorized users may manipulate service access credentials to generate legitimate service requests or replay requests made by authorized users. These actions are categorized as either forgery attacks or replay attacks. (b) Authorized users might exploit expired service access credentials to gain access to services on the edge servers. (c) Revoked users can still access services on the edge servers with their revoked service access credentials. In summary, a malicious user engages in these behaviors with the intention of obtaining a valid credential to successfully authenticate themselves with the edge server. This unauthorized access enables them to exploit resources on the edge servers for their purposes, leading to potential misuse and abuse of the system.

- *Threats from Edge Servers:* We assume that edge servers are not fully trusted in the purpose of providing services without valid credentials and accessing service data of users. (a) An unauthorized edge server has the ability to counterfeit an edge service credential, enabling it to gain access to sensitive service data belonging to users. (b) Even after being revoked, an edge server can continue to provide services to users by utilizing an outdated edge service credential. (c) A revoked edge server can collaborate with other revoked edge servers to illicitly acquire sensitive input data from users. In summary, a malicious or revoked edge server engages in these behaviors with the intent of masquerading as a legitimate service provider, accessing users' sensitive data, and potentially compromising their data confidentiality. This can lead to fraudulent activities by unauthorized parties, causing harm to the honest users.

The adversaries, including malicious users or edge servers, aim to obtain valid credentials (edge service credentials or service access credentials) to access or provide services. To achieve these goals, they may learn knowledge from others' credentials, expired credentials, and revoked credentials. Therefore, to model attack capabilities of adversaries, we enable an adversary to adaptively query an edge service credential or an service access credential on behalf of a malicious edge server or a malicious user, respectively. According to the phases of distributed access control in PEC, an adversary is allowed to make the following queries:

- *EdgeReg Query:* The adversary adaptively queries $(\mathcal{P}, pk, ID_e, S_e)$ to the *EdgeReg* oracle, and the oracle runs the Edge Registration algorithm to generate and return an edge service credential $Cred_s$.
- *UserReg Query:* The adversary adaptively queries $(\mathcal{P}, pk, ID_u, S_u)$ to the *UserReg* oracle, and the oracle runs the User Registration algorithm to generate and return a service access credential $Cred_u$.
- *ServReq Query:* The adversary adaptively queries $(\mathcal{P}, pk, m, ID_u, Cred_u)$ with the service index $i \in S_u$ to the *ServReq* oracle, and the oracle runs the Service Request algorithm to generate and return a service request Req.
- *UserRev Query:* The adversary adaptively queries $(\mathcal{P}, pk, ID_u, Cred_u)$ to the *UserRev* oracle, and the oracle adds it to the *userRevocationTable*.
- *ServRev Query:* The adversary adaptively queries $(\mathcal{P}, pk, ID_e, Cred_s)$ to the *ServRev* oracle, and the oracle revokes the service privilege and adds it to the *edgeRevocationTable*.
- *AuthVer Query:* The adversary adaptively queries the service request Req and the edge service credential $Cred_s$ to the *AuthVer* oracle, and the oracle returns failure or m to show whether Req can succeed the verification with $Cred_s$, i.e., $Cred_s$ can decrypt the ciphertext C.

C. Security Models

The goal of an adversary is to succeed the authentication of the edge servers impersonating a user to access the requested service or to succeed the authentication of the user impersonating an edge server to access the service data of the user. To safeguard PEC against potential adversaries, the access control of users and edge servers should be guaranteed.

- *User Access Control:* 1) Authorization: Prior to accessing the desired services on the edge server, a user must be authorized by the service provider using a service access credential. 2) Authentication: The edge server plays a crucial role in efficiently authenticating the user, ensuring that it possesses the necessary authorization through the service access credential. This step is essential to grant access to the services. 3) Revocation: To prevent resource consumption by malicious or inactive users, it is imperative to promptly revoke a user's service access authorization. This ensures that only authorized individuals can continue to utilize the resources provided by the service provider or edge servers.
- *Edge Server Access Control:* 1) Authorization: Before providing services to users, an edge server must obtain authorization from the service provider through an edge service credential. This step ensures that only authorized edge servers can offer services. 2) Authentication: The user plays a crucial role in efficiently authenticating the edge server to verify its authorization to provide services and access the user's input data. This authentication process ensures that users interact only with trusted and authorized edge servers. 3) Revocation: It is essential to promptly revoke the service authorization of an edge server to prevent malicious servers from accessing users' input data. This proactive revocation measure safeguards the privacy and security of users' sensitive information.

Authorization means that there is no adversary to produce valid credentials for a specific user or edge server, although it may acquire others' credentials or have expired or revoked credentials. To secure authorization, it is impossible for the adversary to forge a valid credential to serve as an edge server or impersonate a user. The unforgeability property should be guaranteed for the service access credentials and edge service credentials.

Unforgeability of Service Access Credentials: The advantage in forging a service access credential of a forger algorithm $\mathcal{A}$, given access to a set of oracles $\mathbb{S}^u =$ *UserReg, ServReq, UserRev, AuthVer* oracles is given as:

$$\mathbf{Adv}Auth^u_{\mathcal{A}} \stackrel{def}{=} \mathbf{Pr}\left[\begin{array}{c} SRes(\mathcal{P}, pk, Cred_s, Req) = m : \\ (\mathcal{P}, msk, pk) \leftarrow INit(\lambda); \\ Cred_s \leftarrow EReg(\mathcal{P}, msk, pk, ID_e, S_e); \\ S_u \cap S_e \neq \emptyset; \\ Cred_u \leftarrow \mathcal{A}^{\mathbb{S}^u}(\mathcal{P}, pk, ID_u, S_u); \\ i \in S_u \cap S_e; \\ Req \leftarrow \mathcal{A}^{\mathbb{S}^u}(\mathcal{P}, pk, m, i, Cred_u) \end{array}\right].$$

Here the adversary $\mathcal{A}$ is allowed to query the oracles adaptively: any of its queries may depend on previous answers, but

it cannot query a service access credential for the service i that is contained in S_e.

Definition 1: The service access credential is unforgable in distributed access control in PEC if the advantage to forge a service access credential for an adversary, i.e., $\mathbf{Adv}Auth^u_{\mathcal{A}}$, is negligible.

Unforgeability of Edge Service Credentials: The advantage in forging an edge service credential of a forger algorithm $\mathcal{A}$, given access to a set of oracles $\mathbb{S}^s =$ *UserReg, EdgeReg, EdgeRev, AuthVer* oracles is given as:

$$\mathbf{Adv}Auth^s_{\mathcal{A}} \stackrel{def}{=} \mathbf{Pr}\left[\begin{array}{c} SRes(\mathcal{P}, pk, Cred_s, Req) = m : \\ (\mathcal{P}, msk, pk) \leftarrow INit(\lambda); \\ Cred_u \leftarrow UReg(\mathcal{P}, msk, pk, ID_u, S_u); \\ i \in S_u \cap S_e; \\ Req \leftarrow SReq(\mathcal{P}, pk, m, i, Cred_u); \\ Cred_s \leftarrow \mathcal{A}^{\mathbb{S}^s}(\mathcal{P}, pk, ID_e, S_e) \end{array}\right].$$

Here the adversary $\mathcal{A}$ is allowed to query the oracles adaptively: any of its queries may depend on previous answers, but it cannot query an edge service credential for the service i that is requested by the user.

Definition 2: The edge service credential is unforgable in distributed access control in PEC if the advantage to forge an edge service credential for an adversary, i.e., $\mathbf{Adv}Auth^s_{\mathcal{A}}$, is negligible.

Mutual authentication is satisfied if both edge service credential and service access credential are valid. The verification of both credentials is achieved in the 0-RTT mutual authentication. During the authentication, the service data of the user is encrypted by the service access credential to produce the service request, which is decrypted by the edge service credential for mutual authentication. The success of correct decryption indicates that both credentials are valid, so the correctness of data encryption guarantees authentication correctness.

In addition, the confidentiality of service data should be supported for protecting the service data of users. To model the confidentiality of the service data, semantic security should be achieved.

Semantic Security of Service Data: The semantic security of the service data is the indistinguishability against chosen plaintext attacks. The advantage to distinguish two plaintexts for an adversary with the knowledge of the ciphertext in the service request and the queries of oracles $\mathbb{S}^d =$ *UserReg, EdgeReg, ServReq, UserRev, EdgeRev, AuthVer* oracles is given in (1) shown at the bottom of next page.

With the initial setup of the semantic security model and the service access credential $Cred_u$ of the targeted user, the challenge of indistinguishability against chosen plaintext attacks is illustrated using the given two messages (m_0, m_1) and the output of SReq Req generated from m_b. The response of the successful adversary should identify that Req is computed from m_{b^*}, where $b = b^*$. The adversary is allowed to query all possible oracles in $\mathbb{S}^d$ that can improve attack capabilities adaptively: any of its queries may depend on previous answers, but it cannot query an edge service credential for the service i that is requested by the user. Finally, if the adversary has more than a 1/2 probability of succeeding in the game, i.e., correctly outputting $b^* = b$, the adversary has the advantage of corrupting the semantic security of the service data.

Definition 3: The service data is confidential in distributed access control in PEC if the advantage to output a b^* for an adversary, such that $b^* = b$, i.e., $\mathbf{Adv}IND^d_{\mathcal{A}}$, is negligible.

D. Complexity Assumptions for Security

We introduce three complexity assumptions used to prove the security of the proposed E-DAC framework.

Let $\mathbb{G}$ and $\mathbb{G}_T$ are two multiplicative cyclic groups with a same prime order p. g is a generator of $\mathbb{G}$. $\mathbb{G}$ and $\mathbb{G}_T$ satisfy a bilinear mapping $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$.

Computational Diffie-Hellman (CDH) problem: given $(g, g^a, g^b) \in \mathbb{G}$, where $a, b \in \mathbb{Z}_p$, to compute g^{ab}.

CDH assumption: there is no probabilistic polynomial time algorithm $\mathcal{A}$ has a non-negligible advantage ϵ in solving the CDH problem that $\mathbf{Pr}[\mathcal{A}(g, g^a, g^b) = g^{ab}] \geq \epsilon$.

Decisional Bilinear Diffie Hellman (DBDH) problem: given $g, g^a, g^b, g^c \in \mathbb{G}$ and a random element $R \in \mathbb{G}_T$, to decide whether $e(g, g)^{abc} = R$.

DBDH assumption: there is no probabilistic polynomial time algorithm $\mathcal{A}$ has a non-negligible advantage ϵ in solving the DBDH problem that $|\mathbf{Pr}[\mathcal{A}(g, g^a, g^b, g^c, e(g, g)^{abc}) = 1] - \mathbf{Pr}[\mathcal{A}(g, g^a, g^b, g^c, R) = 1]| \geq \epsilon$.

Decisional Bilinear Diffie–Hellman Exponent (BDHE) problem: given a vector of $2n + 1$ elements $g, g^{\alpha}, g^{\alpha^2}, \ldots, g^{\alpha^n}, g^{\alpha^{n+2}}, g^{\alpha^{2n}} \in \mathbb{G}$ and a random element $R \in \mathbb{G}_T$, to decide whether $e(g, g)^{\alpha^{n+1}} = R$.

Decisional BDHE assumption: there is no probabilistic polynomial time algorithm $\mathcal{A}$ has a non-negligible advantage ϵ in solving the decisional BDHE problem that $|\mathbf{Pr}[\mathcal{A}(g, g^{\alpha}, g^{\alpha^2}, \ldots, g^{\alpha^n}, g^{\alpha^{n+2}}, g^{\alpha^{2n}}, e(g, g)^{\alpha^{n+1}}) = 1] - \mathbf{Pr}[\mathcal{A}(g, g^{\alpha}, g^{\alpha^2}, \ldots, g^{\alpha^n}, g^{\alpha^{n+2}}, g^{\alpha^{2n}}, R) = 1]| \geq \epsilon$.

IV. Proposed E-DAC

In this section, we present the proposed E-DAC for dynamic PEC services. The framework is derived from KAC [18], with the integration of revocable broadcast encryption [19]. KAC, proposed by Chu et al. [37], is designed for scalable data sharing in cloud storage with the distinguished feature of efficient and flexible data access delegation. In E-DAC, we extend it to support flexible service management and authorization of the service provider. Specifically, the decryption keys of the encrypted files in cloud data sharing are used as the access keys of the services for the edge server in E-DAC. Each service has an access key. With the access key of a service, the edge server can recover the service data sent by the registered user. The authorization key is produced by the service provider for the edge server by aggregating the access keys of the set of authorized services. The authorization key of the edge server has a constant size because of the aggregation of access keys. In addition, to prevent the forgery of an authorization key by combining two valid authorization keys, the authorization key

is generated with the public key of the edge server and the set of services delegated to the edge server, thus the combination of two authorization keys of different edge servers is not valid.

The secret sharing-based broadcast encryption [19] is utilized to achieve service access authorization for users. The service key of the service provider for a specific service is shared between a registered user and an edge server, and both possess different shares of the service key. The successful recovery of the service key in authentication achieves mutual authentication between the user and the edge server. Here, the service key and the authorization key are used by the edge server to decrypt the service data of the user, so that an "always-online" cloud authentication server, i.e., the service provider, is not involved in authentication. Furthermore, to revoke the service authorization to a dropout or rogue edge server, the service provider can release its secret share of the service key. After that, the user uses the released secret share to generate the service request, and the revoked edge server cannot access the service data in the request. The other honest edge servers do not need to update their individual authorization keys, thus the revocation cost is low.

The detailed construction of E-DAC is shown below.

A. Initialization

The system is initialized by a service provider $\mathcal{SP}$, who takes the security level parameter 1^λ and the number of its offered services n as input in the steps below.

1) Parameter Setup: The n services provided by $\mathcal{SP}$ are denoted as the index set $S = \{1, \ldots, n\}$. According to the security level parameter 1^λ, $\mathcal{SP}$ chooses a large prime p, where $2^\lambda \leq p \leq 2^{\lambda+1}$, and generates two multiplicative cyclic groups $\mathbb{G}, \mathbb{G}_T$ of the order p. g and g_0 are the generators chosen from the group $\mathbb{G}$. There is a bilinear mapping $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ for $\mathbb{G}$ and $\mathbb{G}_T$. Also, $\mathcal{SP}$ generates a hash function $H : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$. Next, with the index set of services, $\mathcal{SP}$ randomly chooses $\alpha \in \mathbb{Z}_p^*$ and calculates $g_i = g^{\alpha^i}$, where $i \in \{1, \ldots, n, n+2, \ldots, 2n\}$. Finally, $\mathcal{SP}$ outputs the system parameters $\mathcal{P} = \langle \mathbb{G}, \mathbb{G}_T, e, H, g, g_0, g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n} \rangle$ and erases α safely after parameter setup.

2) Key Generation: $\mathcal{SP}$ randomly chooses $\gamma \in \mathbb{Z}_p^*$ as the master secret key $msk = \gamma$ and computes g^γ as the public key $pk = g^\gamma$. msk is used to generate the authorization keys for edge servers in registration. Also, $\mathcal{SP}$ selects $s \in \mathbb{Z}_p^*$ as the service key and computes g^s. The service key is used to generate the secret shares for users in registration, and the service keys are different for different services.

3) Polynomial Generation: $\mathcal{SP}$ generates a polynomial $f(x)$ of a degree t, $f(x) = \sum_{i=0}^{t} a_i x^i$, where the coefficient $a_0 = s$ and other coefficients $a_i \in \mathbb{Z}_p^*$, for $i = 1, \ldots, t$ are random. t is the revocation threshold.

4) Service Provider Share Generation: According to the polynomial $f(x)$, $\mathcal{SP}$ computes $t-1$ secret shares $\langle (x_{sp_1}, f(x_{sp_1})), \ldots, (x_{sp_{t-1}}, f(x_{sp_{t-1}})) \rangle$, where x_{sp_i} are randomly selected from $\mathbb{Z}_p^*$, for $i = 1, \ldots, t-1$, and $x_{sp_i} \neq x_{sp_j}$, when $i \neq j$. With $t-1$ secret shares, $\mathcal{SP}$ computes $t-1$ protected shares $shr_{sp} = \langle (x_{sp_1}, g^{f(x_{sp_1})}), \ldots, (x_{sp_{t-1}}, g^{f(x_{sp_{t-1}})}) \rangle$.

B. Edge Registration

An edge server $\mathcal{E}$ can provide the services of the service provider $\mathcal{SP}$ to the connected users under the delegation of $\mathcal{SP}$, but it has to register the services on $\mathcal{SP}$ to obtain authorization in the following steps.

1) Registration Request: $\mathcal{E}$ selects a random value $\nu \in \mathbb{Z}_p^*$ to generate $V = g_0^\nu$ as its public key and sends its identity information ID_e, the public key V, and the index set of the requested services $S_e \subseteq S$ to $\mathcal{SP}$ via a secure channel, along with a zero-knowledge proof of the private key ν expressed in Camenisch-Stalder notation [38]: $\pi \leftarrow NIZK\{\nu : V = g_0^\nu\}$.

2) Aggregate Key Generation: Upon receiving the registration request from $\mathcal{E}$, $\mathcal{SP}$ verifies if π is valid. If not, $\mathcal{SP}$ aborts and returns failure. Otherwise, $\mathcal{SP}$ checks whether V has been appeared. If yes, i.e., the edge server ID_e has been registered, $\mathcal{SP}$ aborts and returns failure. Otherwise, based on the index set S_e with the service indices js, $\mathcal{SP}$ calculates the access keys of the services in S_e as g_{n+1-j}^γ, for $j \in S_e$, and further generates an aggregate authorization key K_{S_e} for $\mathcal{E}$ as $K_{S_e} = (V \prod_{j \in S_e} g_{n+1-j})^\gamma$. K_{S_e} is the authorization key for $\mathcal{E}$ on the services in S_e and used to decrypt the encrypted input data of the users who access the services in S_e.

3) Edge Share Generation: $\mathcal{SP}$ chooses $x_e \in \mathbb{Z}_p^*$ and computes $f(x_e)$ to obtain the edge share of the service key $(x_e, f(x_e))$ for $\mathcal{E}$, where $x_e \neq x_{sp_i}$, for $i = 1, \ldots, t-1$.

4) Edge Credential Generation: $\mathcal{SP}$ produces the edge service credential as $Cred_s = \langle ID_e, S_e, K_{S_e}, (x_e, f(x_e)) \rangle$ and returns it to $\mathcal{E}$ via a secure channel. $Cred_s$ is securely stored by $\mathcal{E}$. Meanwhile, $\mathcal{SP}$ maintains $\mathcal{E}$'s credential $Cred_s$ in a secure table $serverTable$.

C. User Registration

Before accessing some services $S_u \subseteq S$ offered by the service provider $\mathcal{SP}$, a user $\mathcal{U}$ needs to subscribe them, so the user registers these services on $\mathcal{SP}$ to acquire authorization.

$$\mathbf{Adv}IND_{\mathcal{A}}^{d} \stackrel{def}{=} \mathbf{Pr}\left[b = b^* : \begin{array}{c} (\mathcal{P}, msk, pk) \leftarrow INit(\lambda); \\ Cred_u \leftarrow UReg(\mathcal{P}, msk, pk, ID_u, S_u); \\ m_0, m_1 \leftarrow \mathcal{A}(\mathcal{P}, pk); \\ b \leftarrow \{0,1\}; \\ i \in S_u; \\ Req \leftarrow SReq(\mathcal{P}, pk, m_b, i, Cred_u); \\ m_{b^*} \leftarrow \mathcal{A}^{\mathbb{S}^d}(\mathcal{P}, pk, Req) \end{array} \right] - \frac{1}{2}. \tag{1}$$

1) Registration Request: To subscribe services, $\mathcal{U}$ sends its identity information ID_u and the set of the indices of the interested services $S_u \subseteq S$ to $\mathcal{SP}$ via a secure channel.

2) User Share Generation: For the set of service indices is, $\mathcal{SP}$ determines the expiration time of the service i in S_u, and computes $x_{u_i} = H(ID_u||i||T_i)$, where ID_u is the identity information of $\mathcal{U}$ and T_i is the expiration time of the service i in S_u. $\mathcal{SP}$ further computes the user share of each registered service $(x_{u_i}, f(x_{u_i}))$ and the protected shares $shr_{u_i} = (x_{u_i}, g^{f(x_{u_i})})$ for each service $i \in S_u$. $\mathcal{SP}$ sets $shr_u = \langle shr_{u_i} \rangle$ and $T = \langle T_i \rangle$ for $i \in S_u$.

3) User Credential Generation: $\mathcal{SP}$ generates the service access credential $Cred_u = \langle ID_u, S_u, g^s, shr_{sp}, shr_u, T \rangle$ from the user shares and the service provider returns it to $\mathcal{U}$ via a secure channel. Upon receiving $Cred_u$, $\mathcal{U}$ securely maintains $Cred_u$. Meanwhile, $\mathcal{SP}$ stores $Cred_u$ in a secure table $userTable$.

D. Service Request

The registered user $\mathcal{U}$ can request a subscribed service $i \in S_u$ from the service provider $\mathcal{SP}$ or an edge server $\mathcal{E}$ if it has been delegated to provide such service in the following steps:

1) Randomization: $\mathcal{U}$ selects a random value $k \in \mathbb{Z}_p^*$ to calculate g^k and randomizes the protected shares in $Cred_u$ as $shr_r = \langle (x_{u_i}, g^{kf(x_{u_i})}), (x_{sp_1}, g^{kf(x_{sp_1})}), \ldots, (x_{sp_{t-1}}, g^{kf(x_{sp_{t-1}})}) \rangle$.

2) Data Encryption: The input data $m \in \mathbb{Z}^*_{2^{|\mathbb{G}_T|-|p|}}$ is the service data that $\mathcal{U}$ submits to $\mathcal{SP}$ or $\mathcal{E}$ for accessing the requesting service i. Here $|\mathbb{G}_T|$ is the binary size of the elements in $\mathbb{G}_T$ and $|p|$ is the binary size of p. To achieve 0-RTT mutual authentication, $\mathcal{U}$ encrypts the input data m of the service i by randomly selecting $t \in \mathbb{Z}_p$ and computing the ciphertext of m as $C = \langle g^t, g_0^t, (pk \cdot g_i)^t, (m||H(m)) \cdot e(g_1, g_n)^t \cdot e(g^k, g^s)^t \rangle$.

3) Request Broadcast: $\mathcal{U}$ sets its service request $Req = \langle ID_u, i, shr_r, g^k, C, T_i, T_s \rangle$, where i is the service index, T_i is the expiration time of the service i, and T_s is a timestamp, and broadcasts the request Req into the edge network.

E. Service Response

After receiving a service request Req from a user $\mathcal{U}$, the edge server $\mathcal{E}$ executes the following way to respond the request.

1) Request Verification: From the service request Req, $\mathcal{E}$ extracts the index of the requested service i and checks if the requested service is in the scope of its capability. If it is not authorized to provide this service, $\mathcal{E}$ forwards Req to other edge servers; otherwise, it handles the request in the following steps:

$\mathcal{E}$ first compares the timestamp T_s with the current time T_c to ensure the request freshness. Then, $\mathcal{E}$ extracts $(x_{u_i}, g^{kf(x_{u_i})})$ from shr_r and verifies whether $H_1(ID_u||i||T_i) = x_{uid}$ to check user identity, service, and expiration time, respectively. If holds, $\mathcal{E}$ searches ID_u in its $userRevocationTable$ to ensure $\mathcal{U}$ has not been revoked and compares T_i with T_c to ensure that the service authorization is not expired. If any check fails, $\mathcal{E}$ returns failure and aborts. Otherwise, $\mathcal{E}$ executes data decryption.

2) Data Decryption: To decrypt the input data m, $\mathcal{E}$ recovers the protected and randomized version of the service key g^{ks} from its share and the shares in the service request Req as follows: $\mathcal{E}$ first utilizes its secret share of the service key $(x_e, f(x_e))$ and g^k to compute the $(t+1)$-th protected and randomized share $(x_e, (g^k)^{f(x_e)})$. With the t protected and randomized shares shr_r in Req, $\mathcal{E}$ recovers g^{ks} through polynomial interpolation:

$$g^{ks} = g^{ka_0} = g^{k\sum_{i=0}^{t} f(x_i)\lambda_i} = \prod_{i=0}^{t} (g^{kf(x_i)})^{\lambda_i},$$

where $\lambda_i = \prod_{j=0, j\neq i}^{t} \frac{-x_i}{x_j - x_i}$. Here $x_0 = x_{u_i}$, $x_1 = x_{sp_1}$, …, $x_{t-1} = x_{sp_{t-1}}$, $x_t = x_e$; $f(x_0) = f(x_{u_i})$, $f(x_1) = f(x_{sp_1})$, …, $f(x_{t-1}) = f(x_{sp_{t-1}})$, $f(x_t) = f(x_e)$. After recovering g^{ks}, $\mathcal{E}$ decrypts $C = \langle c_1, c_2, c_3, c_4 \rangle$ to obtain $m||H(m)$ by using K_{S_e} and g^{ks} as follows:

$$m||H(m) = c_4 \cdot \frac{e(K_{S_e} \cdot \prod_{j\in S_e, j\neq i} g_{n+1-j+i}, c_1)}{e(pk, c_2)^{\nu} e(\prod_{j\in S_e} g_{n+1-j}, c_3) \cdot e(g^t, g^{ks})}.$$

Finally, $\mathcal{E}$ verifies whether the recovered m is correct based on the hash value. If yes, $\mathcal{E}$ replies the request; otherwise, $\mathcal{E}$ aborts and returns failure.

3) Request Response: If the input data m is decrypted successfully, $\mathcal{E}$ fulfills u's requested service.

F. Service Revocation

Service revocation enables the service provider $\mathcal{SP}$ to withdraw the access ability of the users or the ability of the edge servers to provide services if any misbehavior is identified.

1) User Revocation: If the user $\mathcal{U}$'s authorization needs to be revoked, the service provider $\mathcal{SP}$ checks the set of the edge servers E who serve $\mathcal{U}$ from $serverTable$. Then, $\mathcal{SP}$ sends a revocation notification and $\mathcal{U}$'s identity to each edge server in E and deletes $\mathcal{U}$'s user credential in $userTable$. Once receiving the revocation notification, the edge servers add $\mathcal{U}$'s identity in their $userRevocationTable$.

2) Edge Server Revocation: To revoke the edge server $\mathcal{E}$'s service authorization, $\mathcal{SP}$ finds $\mathcal{E}$'s credential $Cred_s$, extracts the edge share $(x_e, f(x_e))$, and replaces one of the shares shr_{sp} with $(x_e, g^{f(x_e)})$, which is generated from the edge share $(x_e, f(x_e))$. Then, $\mathcal{SP}$ sends the updated shares shr^*_{sp} to all users. Because the number of the shares shr_{sp} is $t-1$, the maximal number of edge servers that can be revoked is $t-1$. If the number of revoked edge servers reaches $t-1$, $\mathcal{SP}$ should update the service key and re-distribute shares of the service key to users and edge servers.

Correctness: The correctness of the proposed E-DAC is demonstrated in (2) shown at the bottom of next page. The correctness ensures correct mutual authentication between the user and the edge server.

V. Security Analysis

In this section, we analyze the unforgeability of service access credentials, the unforgeability of edge service credentials, and the confidentiality of the service data of users. The mutual authentication is achieved if the correctness of E-DAC holds, which is built upon the fact that the service access credential of the user and the edge service credential of the edge server are

correctly produced. Here, we prove the unforgeabilty of both credentials of the user and the edge server.

Theorem 1: (Unforgeability of Service Access Credentials). Assume the CDH problem is hard, the service access credential in E-DAC is unforgeable against the chosen service attacks.

Proof: The unforgeability of service access credentials can be reduced to the CDH assumption, that is, within non-negligible advantage, if there is no probabilistic polynomial-time algorithm to solve CDH problem, then, the service access credential in E-DAC is unforgeable against the chosen service attacks. Suppose that an adversary $\mathcal{A}$ can break the unforgeability of service access credentials with a non-negligible probability, then we can construct an algorithm $\mathcal{B}$ to solve the CDH problem: given $g, g^a, g^b \in \mathbb{G}$, where $a, b \in \mathbb{Z}_p$, to compute $g^{ab} \in \mathbb{G}$. $\mathcal{B}$ simulates a challenger to interact with $\mathcal{A}$ in the following way:

In initialization, $\mathcal{B}$ setups the parameters $(\mathcal{P}, S, V, \gamma, g^\gamma)$ and calculates $g_0 = g^\beta$, $g_i = g^{\alpha^i}$, where $\alpha, \beta \in \mathbb{Z}_p^*$ and $i \in \{1, \ldots, n, n+2, \ldots, 2n\}$. The service key g^s is set as g^a, and the points $(x_{sp_1}, y_{sp_1}), \ldots, (x_{sp_{t-1}}, y_{sp_{t-1}})$ are randomly chosen. $\mathcal{B}$ also selects $S_e \in S$ to compute $K_{S_e} = (V \prod_{j \in S_e} g_{n+1-j})^\gamma$ and randomly chooses the point (x_e, y_e). Finally, $\mathcal{B}$ sends $(\mathcal{P}, S, g^\gamma, g^a, g^\beta, g^\alpha, \ldots, g^{\alpha^n}, g^{\alpha^{n+2}}, \ldots, g^{\alpha^{2n}})$ to $\mathcal{A}$.

$\mathcal{B}$ programs oracles to answer the queries from $\mathcal{A}$. To ensure the consistency, it maintains a list of tuples to keep the queries and corresponding responses. The queries are answered as follows:

- When receiving the $UserReg$ query $(\mathcal{P}, g^\gamma, ID_u, S_u)$, $\mathcal{B}$ randomly selects (x_{u_i}, y_{u_i}), computes $(x_{u_i}, g^{y_{u_i}})$, $(x_{sp_1}, g^{y_{sp_1}}), \ldots, (x_{sp_{t-1}}, g^{y_{sp_{t-1}}})$, and returns them to $\mathcal{A}$, along with g^a and a random T, that is, $Cred_u$.
- When receiving the $ServReq$ query $(\mathcal{P}, g^\gamma, m, ID_u, Cred_u, i)$, where $i \in S_u$, $\mathcal{B}$ randomizes $Cred_u$ to produce $shr_r = \langle (x_{u_i}, (g^b)^{y_{u_i}}), (x_{sp_1}, (g^b)^{y_{sp_1}}), \ldots, (x_{sp_{t-1}}, (g^b)^{y_{sp_{t-1}}}) \rangle$, uses (x_{u_i}, y_{u_i}), (x_e, y_e), $(x_{sp_1}, y_{sp_1}), \ldots, (x_{sp_{t-1}}, y_{sp_{t-1}})$ to reconstruct the polynomial $f(x) = s' + \sum_{i=1}^{t} a_t x^t$ and obtain s'. Then, $\mathcal{B}$ randomly chooses $t \in \mathbb{Z}_p^*$ to compute the ciphtext $C = \langle g^t, g_0^t, (pk \cdot g_i)^t, (m||H(m)) \cdot e(g_1, g_n)^t \cdot e(g^b, g^{s'})^t \rangle$. Finally, $\mathcal{B}$ returns $Req = \langle ID_u, i, shr_r, g^b, C, T, T_s \rangle$ to $\mathcal{A}$.
- When receiving the $UserRev$ query $(\mathcal{P}, pk, ID_u, Cred_u)$, $\mathcal{B}$ adds it to the $userRevocationTable$.
- When receiving the $AuthVer$ query Req, $\mathcal{B}$ uses $Cred_s = \langle S_e, K_{S_e}, (x_e, y_e) \rangle$ that is generated in initialization to verify the validity of Req and returns failure or m to $\mathcal{A}$.

Eventually, $\mathcal{A}$ produces a service access credential $Req^* = \langle ID_{u^*}, i^*, shr_r^*, g^b, C^*, T^*, T_s^* \rangle$, where $shr_r^* = ((x_{u_i}^*, (g^b)^{y_{u_i}^*}), (x_{sp_1}, (g^b)^{y_{sp_1}}), \ldots, (x_{sp_{t-1}}, (g^b)^{y_{sp_{t-1}}})$. If the service i^* has been queried or $i^* \notin S_e$, $\mathcal{B}$ returns failure. Otherwise, $\mathcal{B}$ uses $Cred_s = \langle S_e, K_{S_e}, (x_e, y_e) \rangle$ to verify whether Req^* is valid or not, and thereby determines whether $Cred_u^*$ is valid or not. If $Cred_u^*$ is invalid, $\mathcal{B}$ returns failure; otherwise, $\mathcal{B}$ reconstructs g^{ab} as:

$$g^{ab} = g^{b \sum_{i=0}^{t} f(x_i) \lambda_i} = \prod_{i=0}^{t} (g^{k f(x_i)})^{\lambda_i},$$

where $\lambda_i = \prod_{j=0, j \neq i}^{t} \frac{-x_i}{x_j - x_i}$. Here $x_0 = x_{u_i}^*$, $x_1 = x_{sp_1}$, $\ldots$, $x_{t-1} = x_{sp_{t-1}}$, $x_t = x_e$; $f(x_0) = y_{u_i}^*$, $f(x_1) = y_{sp_1}$, $\ldots$, $f(x_{t-1}) = y_{sp_{t-1}}$, $f(x_t) = y_e$.

Thus, the CDH problem has been solved. We prove that if there is an adversary forges the service access credential, the CDH problem can be solved. However, the advantage to solve the CDH problem is negligible, the probability to

$$
\begin{aligned}
& c_4 \cdot \frac{e(K_{S_e} \cdot \prod_{j \in S_e, j \neq i} g_{n+1-j+i}, c_1)}{e(pk, c_2)^\nu e(\prod_{j \in S_e} g_{n+1-j}, c_3) \cdot e(g^t, g^{ks})} \\
&= (m||H(m)) \cdot e(g_1, g_n)^t \cdot e(g^k, g^s)^t \frac{e(\prod_{j \in S_e} g_0^{\nu\gamma} g_{n+1-j}^\gamma, g^t) e(\prod_{j \in S_e, j \neq i} g_{n+1-j+i}, g^t)}{e(g^\gamma, g_0)^{t\nu} e(\prod_{j \in S_e} g_{n+1-j}, c_3) \cdot e(g^t, g^{ks})} \\
&= (m||H(m)) \cdot e(g_1, g_n)^t \frac{e(\prod_{j \in S_e} g_{n+1-j}^\gamma, g^t) e(\prod_{j \in S_e, j \neq i} g_{n+1-j+i}, g^t)}{e(\prod_{j \in S_e} g_{n+1-j}, (pk \cdot g_i)^t)} \\
&= (m||H(m)) \cdot e(g_1, g_n)^t \frac{e(\prod_{j \in S_e} g_{n+1-j}^\gamma, g^t) e(\prod_{j \in S_e, j \neq i} g_{n+1-j+i}, g^t)}{e(\prod_{j \in S_e} g_{n+1-j}, g^{\gamma t}) e(\prod_{j \in S_e} g_{n+1-j}, g_i^t)} \\
&= (m||H(m)) \cdot e(g_1, g_n)^t \frac{e(\prod_{j \in S_e, j \neq i} g_{n+1-j+i}, g^t)}{e(\prod_{j \in S_e} g_{n+1-j+i}, g^t)} \\
&= (m||H(m)) \cdot \frac{e(g_1, g_n)^t}{e(g_{n+1}, g^t)} \\
&= (m||H(m)) \cdot \frac{e(g^\alpha, g^{\alpha^n})^t}{e(g^{\alpha^{n+1}}, g^t)} \\
&= m||H(m). \qquad (2)
\end{aligned}
$$

generate a service access credential in E-DAC for an adversary is negligible.

Theorem 2: (Unforgeability of Edge Service Credentials) Assume the CDH problem is hard, the edge service credential in E-DAC is unforgeable against the chosen service attacks.

Proof: The unforgeability of edge service credentials can be reduced to the CDH assumption, that is, within non-negligible advantage, if there is no probabilistic polynomial-time algorithm to solve CDH problem, then, the edge service credential in E-DAC is unforgeable against chosen service attacks. Suppose that an adversary $\mathcal{A}$ can break the unforgeability of edge service credentials with a non-negligible probability, then we can construct an algorithm $\mathcal{B}$ to solve the CDH problem: given $g, g^a, g^b \in \mathbb{G}$, where $a, b \in \mathbb{Z}_p$, to compute $g^{ab} \in \mathbb{G}$. $\mathcal{B}$ simulates a challenger to interact with $\mathcal{A}$ in the following way:

In initialization, $\mathcal{B}$ setups the parameters $(\mathcal{P}, S, g^s)$ and the polynomial $f(x)$ of a degree t, where the coefficient $a_0 = s$ and other coefficients $a_i \in \mathbb{Z}_p^*$, for $i = 1, \ldots, t$, are randomly generated. $(x_{sp_1}, f(x_{sp_1})), \ldots, (x_{sp_{t-1}}, f(x_{sp_{t-1}}))$ are randomly chosen based on $f(x)$. The public key is set as g^a. $\mathcal{B}$ generates $g_0 = g^\beta$, $g_i = g^{\alpha^i}$, $h = g^b$, $h_0 = h^\beta$, and $h_i = (g^b)^{\alpha^i}$, where $\alpha, \beta \in \mathbb{Z}_p^*$ and $i \in \{1, \ldots, n, n+2, \ldots, 2n\}$.

$\mathcal{B}$ randomly selects $(x_{u_i}, f(x_{u_i}))$ and computes $(x_{u_i}, g^{f(x_{u_i})}), (x_{sp_1}, g^{f(x_{sp_1})}), \ldots, (x_{sp_{t-1}}, g^{f(x_{sp_{t-1}})})$, which is set as $Cred_u$, together with (ID_u, S_u, g^s, T). Then, $\mathcal{B}$ selects the challenging service i^* and a random value $k \in \mathbb{Z}_p^*$, and calculates shr_r in Req as $shr_r = \langle (x_{u_i}, g^{kf(x_{u_i})}), (x_{sp_1}, g^{kf(x_{sp_1})}), \ldots, (x_{sp_{t-1}}, g^{kf(x_{sp_{t-1}})}) \rangle$. Then, $\mathcal{B}$ randomly chooses $t \in \mathbb{Z}_p^*$ to compute $C = \langle h^t, (pk \cdot h_{i^*})^t, (m||H(m)) \cdot e(h_1, h_n)^t \cdot e(g^k, g^s)^t \rangle$, and sets $Req^* = \langle ID_u, i^*, shr_r, g^k, C, T, T_s \rangle$. Finally, $\mathcal{B}$ sends $(\mathcal{P}, S, g^a, g^s, (x_{sp_1}, f(x_{sp_1})), \ldots, (x_{sp_{t-1}}, f(x_{sp_{t-1}})), \{g_i, h_i\}_{i \in \{1,\ldots,n,n+2,\ldots,2n\}})$ to $\mathcal{A}$.

$\mathcal{B}$ programs oracles to answer the queries from $\mathcal{A}$. To ensure the consistency, it maintains a list of tuples to keep the queries and corresponding responses. The queries are answered as follows:

- When receiving the $EdgeReg$ query $(\mathcal{P}, g^a, g^{\beta v}, ID_e, i)$, where v is randomly chosen in $\mathbb{Z}_p^*$, $\mathcal{B}$ generates a random coin $c_i \in \{0, 1\}$, where the service $i \in S_e$. If $c_i = 0$, $\mathcal{B}$ computes $l_i = (g^a)^{\alpha^i}$, for $i \in \{1, \ldots, n, n+2, \ldots, 2n\}$, and $K_{S_e} = g^{a\beta v} \prod_{j \in S_e} l_{n+1-j}$, randomly chooses $(x_e, f(x_e))$, and returns $Cred_s = \langle ID_e, i, c_i, K_{S_e}, (x_e, f(x_e)) \rangle$ to $\mathcal{A}$; otherwise, $\mathcal{B}$ aborts and returns failure.
- When receiving the $UserReg$ query $(\mathcal{P}, g^a, ID_u, S_u)$, $\mathcal{B}$ randomly selects $(x_{u_i}, f(x_{u_i}))$ and computes $(x_{u_i}, g^{f(x_{u_i})}), (x_{sp_1}, g^{f(x_{sp_1})}), \ldots, (x_{sp_{t-1}}, g^{f(x_{sp_{t-1}})})$, which is set as $Cred_u$, together with (ID_u, S_u, g^s, T). Then, $\mathcal{B}$ selects the service $i \neq i^*$ and a random value $k \in \mathbb{Z}_p^*$, and calculates shr_r in Req as $shr_r = \langle (x_{u_i}, g^{kf(x_{u_i})}), (x_{sp_1}, g^{kf(x_{sp_1})}), \ldots, (x_{sp_{t-1}}, g^{kf(x_{sp_{t-1}})}) \rangle$. If $c_i = 0$, $\mathcal{B}$ randomly chooses $t \in \mathbb{Z}_p^*$ to compute $C = \langle g^t, g^{t\beta}, (pk \cdot g_i)^t, (m||h(m)) \cdot e(g_1, g_n)^t \cdot e(g^k, g^s)^t \rangle$. If $c_i = 1$, $\mathcal{B}$ randomly chooses $t \in \mathbb{Z}_p^*$ to compute $C = \langle h^t, h^{t\beta}, (pk \cdot h_i)^t, (m||h(m)) \cdot e(h_1, h_n)^t \cdot e(g^k, g^s)^t \rangle$. Finally, $\mathcal{B}$ sets $Req = \langle ID_u, i, shr_r, g^k, c_i, C, T, T_s \rangle$ and returns to $\mathcal{A}$.
- When receiving the $EdgeRev$ query $(\mathcal{P}, pk, ID_e, Cred_s)$, $\mathcal{B}$ sends the secret share of the edge server $(x_e, f(x_e))$ to all the users, and adds it to the *edgeRevocationTable*.
- When receiving the $AuthVer$ query $(Cred_s, Req)$, $\mathcal{B}$ checks whether $c_i = 0$ or $c_i = 1$. If $c_i = 0$, $\mathcal{B}$ verifies Req and $Cred_s$ by using Service Response algorithm, and returns failure or m to $\mathcal{A}$. If $c_i = 1$, $\mathcal{B}$ returns failure and aborts.

Eventually, $\mathcal{A}$ produces an edge service credential $Cred_s^* = \langle ID_e^*, S_e^*, c_{i^*}, K_{S_e^*}, (x_e^*, f(x_e^*)) \rangle$, where $i^* \in S_e^*$. $\mathcal{B}$ uses Req^* to verify whether $Cred_s^*$ is valid or not. If not, $\mathcal{B}$ returns failure and aborts; otherwise, $\mathcal{B}$ checks if $c_i = 1$. If not, $\mathcal{B}$ returns failure and aborts; otherwise, $\mathcal{B}$ computes $g^{ab} = (K_{S_e^*})^{(\beta v + \sum_{i^* \in S_e^*} \alpha^{n+1-i^*})^{-1}}$.

Thus, the CDH problem has been solved. We prove that if there is an adversary forges the edge service credential, the CDH problem can be solved. However, the advantage to solve the CDH problem is negligible, the probability to generate an edge service credential in E-DAC for an adversary is negligible.

Theorem 3: (*Semantic Security of Service Data*) Assume the decisional BDHE and DBDH problems are hard, the service data in E-DAC is semantically secure.

The confidentiality of the service data can be reduced to the decisional BDHE and DBDH assumptions, that is, within non-negligible advantage, if there is no probabilistic polynomial-time algorithm to solve both decisional BDHE and DBDH problems, then, the service data in E-DAC is semantically secure. To prove this statement, we divide the theorem 3 into two lemmas.

Lemma 1: Assume that the DBDH problem is hard, $m' = (m||H(m))e(g_1, g_n)^t$ in the ciphtext $C' = \langle g^t, g_0^t, (pk \cdot g_i)^t, m' \cdot e(g^k, g^s)^t \rangle$ is indistinguishable against chosen plaintext attacks.

Proof: Suppose that an adversary $\mathcal{A}$ can break the semantic security with a non-negligible probability, then we can construct an algorithm $\mathcal{B}$ to solve the DBDH problem: given $g, g^a, g^b, g^c \in \mathbb{G}$ and a random element $R \in \mathbb{G}_T$, to decide whether $e(g, g)^{abc} = R$. $\mathcal{B}$ simulates a challenger to interact with $\mathcal{A}$ in the following way:

In initialization, $\mathcal{B}$ setups the parameters $(\mathcal{P}, S, V, \gamma, g^\gamma)$ and calculates $g_0 = g^\beta$, $g_i = g^{\alpha^i}$, where $\alpha, \beta \in \mathbb{Z}_p^*$ and $i \in \{1, \ldots, n, n+2, \ldots, 2n\}$. The service key g^s is set as g^a, and the points $(x_{sp_1}, y_{sp_1}), \ldots, (x_{sp_{t-1}}, y_{sp_{t-1}})$ are randomly chosen. $\mathcal{B}$ also selects $S_e \in S$ to compute $K_{S_e} = (V \prod_{j \in S_e} g_{n+1-j})^\gamma$ and randomly chooses the point (x_e, y_e). Finally, $\mathcal{B}$ sends $(\mathcal{P}, S, g^\gamma, g^a, g^\beta, g^\alpha, \ldots, g^{\alpha^n}, g^{\alpha^{n+2}}, \ldots, g^{\alpha^{2n}})$ to $\mathcal{A}$.

$\mathcal{B}$ programs oracles to answer the queries from $\mathcal{A}$. To ensure the consistency, it maintains a list of tuples to keep the queries and corresponding responses. The queries are answered as follows:

- When receiving the $UserReg$ query $(\mathcal{P}, g^\gamma, ID_u, S_u)$, $\mathcal{B}$ randomly selects (x_{u_i}, y_{u_i}), computes $(x_{u_i}, g^{y_{u_i}})$, $(x_{sp_1}, g^{y_{sp_1}}), \ldots, (x_{sp_{t-1}}, g^{y_{sp_{t-1}}})$, and returns them to $\mathcal{A}$, along with g^a and a random T, that is, $Cred_u$.

- When receiving the $ServReq$ query $(\mathcal{P}, g^{\gamma}, m', Cred_u, i)$, where $i \in S_u$, $\mathcal{B}$ randomizes $Cred_u$ to produce $shr_r = \langle (x_{u_i}, (g^b)^{y_{u_i}}), (x_{sp_1}, (g^b)^{y_{sp_1}}), \ldots, (x_{sp_{t-1}}, (g^b)^{y_{sp_{t-1}}}) \rangle$, uses (x_{u_i}, y_{u_i}), (x_e, y_e), $(x_{sp_1}, y_{sp_1}), \ldots, (x_{sp_{t-1}}, y_{sp_{t-1}})$ to reconstruct the polynomial $f(x) = a' + \sum_{i=1}^{t} a_t x^t$ and obtain a'. Then, $\mathcal{B}$ computes the ciphtext $C' = \langle g^t, g^{t\beta}, (g^t)^{\gamma+\alpha^i}, m' \cdot e(g^b, g^t)^{a'} \rangle$, where $t \in \mathbb{Z}_p^*$. Finally, $\mathcal{B}$ returns $Req = \langle ID_u, i, shr_r, g^b, C', T, T_s \rangle$ to $\mathcal{A}$.
- When receiving the $UserRev$ query $(\mathcal{P}, pk, ID_u, Cred_u)$, $\mathcal{B}$ adds it to the $userRevocationTable$.
- When receiving the $AuthVer$ query Req, $\mathcal{B}$ uses $Cred_s = \langle S_e, K_{S_e}, (x_e, y_e) \rangle$ that is generated in initialization to verify the validity of Req and returns failure or m to $\mathcal{A}$.

For m_0' and m_1' chosen by $\mathcal{A}$, $\mathcal{B}$ randomly chooses $\nu \in \{0, 1\}$ and encrypts m_ν' as $C_\nu' = \langle g^c, g^{c\beta}, (g^c)^{\gamma+\alpha^i}, m_\nu' \cdot D \rangle$. $\mathcal{B}$ randomly generates $(x_{u_i}, (g^b)^{y_{u_i}}), (x_{sp_1}, (g^b)^{y_{sp_1}}), \ldots, (x_{sp_{t-1}}, (g^b)^{y_{sp_{t-1}}}) \rangle$ to generate shr_r. (C_ν', shr_r) are returned to $\mathcal{A}$, and $\mathcal{A}$ outputs ν^*. It is obvious that if $\mathcal{A}$ can correctly tell ν^*, such that $\nu^* = \nu$, $\mathcal{B}$ can distinguish whether $D = e(g, g)^{abc}$ or $D = R$, so as to solve the DBDH problem.

Lemma 2: Assume that the decisional DBHE problem is hard, the ciphtext $C = \langle g^t, g_0^t, (pk \cdot g_i)^t, (m||H(m)) \cdot e(g_1, g_n)^t \rangle$ is indistinguishable against chosen plaintext attacks.

Proof: Suppose that an adversary $\mathcal{A}$ can break the semantic security with a non-negligible probability, then we can construct an algorithm $\mathcal{B}$ to solve the decisional DBHE problem: given a vector of $2n + 1$ elements $(g, g^{\alpha}, g^{\alpha^2}, \ldots, g^{\alpha^n}, g^{\alpha^{n+2}}, g^{\alpha^{2n}}) \in \mathbb{G}$ and a random element $R \in \mathbb{G}_T$, to decide whether $e(g, g)^{\alpha^{n+1}} = R$. $\mathcal{B}$ simulates a challenger to interact with $\mathcal{A}$ in the following way:

In initialization, $\mathcal{B}$ setups the parameters $(\mathcal{P}, S, \gamma, g^{\gamma}, g^s)$ and the polynomial $f(x)$ of a degree t. $(x_{sp_1}, f(x_{sp_1})), \ldots, (x_{sp_{t-1}}, f(x_{sp_{t-1}}))$ are randomly chosen based on $f(x)$. $\mathcal{B}$ sets $g_0 = g^{\beta}$, $g_i = g^{\alpha^i}$, where $\beta \in \mathbb{Z}_p^*$ and $i \in \{1, \ldots, n, n+2, \ldots, 2n\}$. $\mathcal{B}$ sends them to $\mathcal{A}$.

$\mathcal{B}$ programs oracles to answer the queries from $\mathcal{A}$. To ensure the consistency, it maintains a list of tuples to keep the queries and corresponding responses. Because $\mathcal{B}$ possesses all the secret values (γ, s), except α, which is not needed in the generation of edge service credentials and service access credentials, $\mathcal{B}$ can answer all the oracles without any differences with the corresponding algorithms in E-DAC.

For m_0 and m_1 chosen by $\mathcal{A}$, $\mathcal{B}$ randomly chooses $\nu \in \{0, 1\}$ and encrypts m_ν as $C_\nu = \langle g^t, g^{t\beta}, (pk \cdot g_i)^t, (m_\nu||H(m_\nu)) \cdot D^t \rangle$, which is returned to $\mathcal{A}$. It is obvious that if $\mathcal{A}$ can correctly tell ν^*, such that $\nu^* = \nu$, $\mathcal{B}$ can distinguish whether $D = e(g, g)^{\alpha^{n+1}}$ or $D = R$, so as to solve the decisional DBHE problem.

VI. Performance Evaluation

In this section, we evaluate the computational and communication performance of E-DAC to demonstrate that our E-DAC outperforms the existing works.

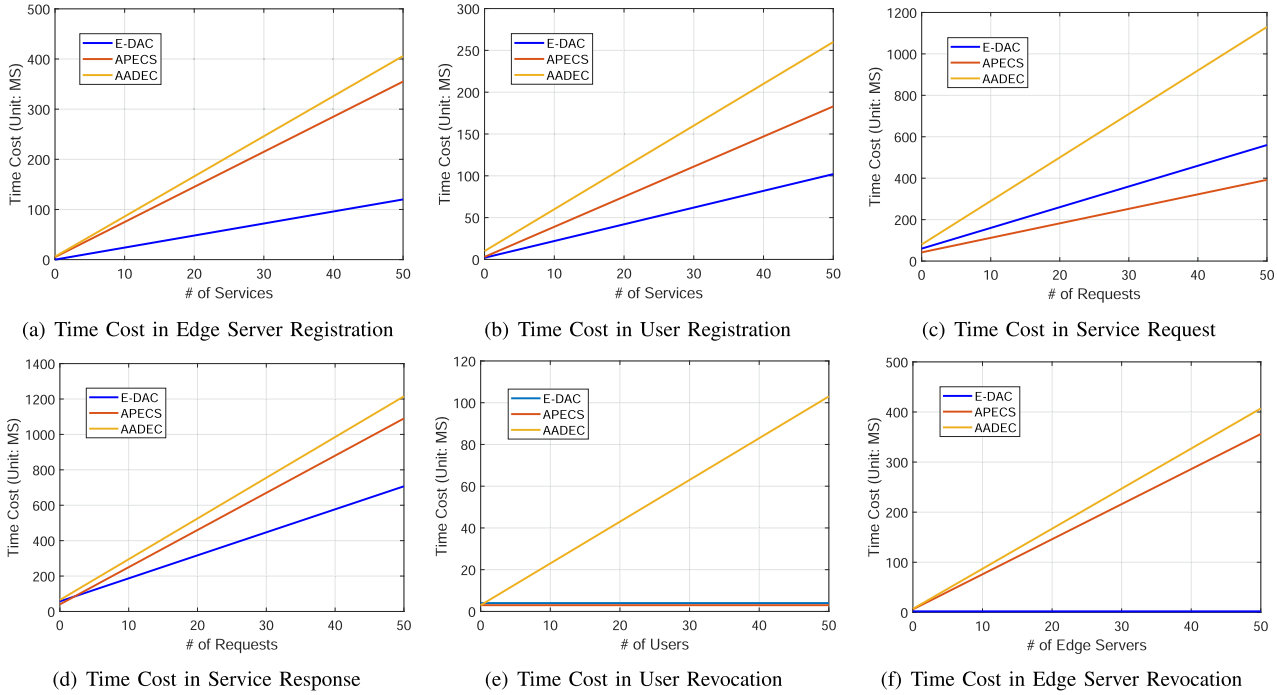
We show the computational overhead of E-DAC by quantifying the number of specific operations performed in $\mathbb{G}$ and $\mathbb{G}_T$. The operations under evaluation include exponentiation in $\mathbb{G}$, exponentiation in $\mathbb{G}_T$, bilinear pairing operations, polynomial evaluation, and multiplication in $\mathbb{G}$. Notably, other operations like the hash operation and multiplication in $\mathbb{Z}p^*$ are relatively less time-consuming in comparison. For convenience, we use $E_{\mathbb{G}}$, $E_{\mathbb{G}_T}$, BP, $Ploy$, and $Mul_{\mathbb{G}}$ to represent exponentiation in $\mathbb{G}$, exponentiation in $\mathbb{G}_T$, bilinear pairing, polynomial evaluation, and multiplication in $\mathbb{G}$, respectively. The third column of Table I displays the number of computations involved in each phase of E-DAC. Let $v = |S_e|$ be the number of registered services of the edge server, and $w = |S_u|$ be the number of registered services of the user. Notably, the revocation process demonstrates high efficiency, requiring no time-consuming operations for either the edge server or the user. Moreover, in the service request phase, the bilinear pairing can be pre-computed, allowing the user to calculate $e(g_1, g_n)$ and $e(g, g^s)$ in advance, thereby expediting the service request generation.

We conducted comprehensive real-world experiments to measure the time consumption of the service provider, the edge server, and the user. The platform comprises a Precision 7920 Tower Workstation and two ThinkPad X1 Carbon laptops. The Precision 7920 Tower Workstation is configured with an Intel Xeon Gold 6242R 4.1 GHz 20 C CPU and 128 GB RAM, and the ThinkPad X1 Carbon laptop is equipped with an Intel Core i7-1365 U vPro 3.90 GHz CPU and 32 GB RAM. The operations of the service provider are carried out on the Precision 7920 Tower Workstation, and the operations of the edge server and the user are conducted on two ThinkPad X1 Carbon laptops, respectively. The operating systems of these devices are Windows 11, and they are connected to a WiFi router to build a network. To implement number-theoretic based methods of cryptography, we utilized the JPBC library. For achieving bilinear pairing, we employed Type-A pairing. The elliptic curve is represented as $E : y^2 = x^3 + x$ and defined over F_p, with an embedding degree of $k = 6$. The parameter p is a large prime, approximately 256 bits in size. When $v = 1$ and $w = 1$, the time cost of an entity engaged in E-DAC is shown in the fourth column of Table I. This time cost denotes the duration taken by an entity to execute the necessary operations within each phase of E-DAC. Furthermore, the entire E-DAC is executed on the experimental platform. We gather the execution cost of each party in every phase of E-DAC on the platform. The results are presented in the sixth column of Table I. This cost is the real execution time cost of E-DAC on the platform, coving computing time, communication expenses, and other associated processing costs on the devices.

Meanwhile, we conducted a comparative analysis of the computational overhead between our proposed E-DAC and the existing schemes, APECS [17] and AADEC [36]. The results of this comparison are illustrated in Fig. 2. During edge server registration, E-DAC exhibits high efficiency in authorization key generation since it only involves access key aggregation and edge share generation performed by the service provider. In contrast, both APECS and AADEC require cooperation between

TABLE I
PERFORMANCE OF E-DAC

Phases	Entities	Computational Overhead	Algorithm Time Cost (Unit: ms)	Communication Cost (Bits)	Execution Cost (Unit: ms)
Initialization	Service Provider	$(2n+t+1)E_{\mathbb{G}}+(t-1)Ploy$	694	–	698
Edge Registration	Service Provider	$(v+1)E_{\mathbb{G}}+vMul_{\mathbb{G}}+Ploy$	16	833	315
	Edge Server	$2E_{\mathbb{G}}$	2	1346	
User Registration	Service Provider	$wE_{\mathbb{G}}+wPloy$	14	$384+512t$	287
	User	–	–	64	
Service Request	User	$(t+4)E_{\mathbb{G}}+2BP+E_{\mathbb{G}_T}+Mul_{\mathbb{G}}$	79	$2824+512t$	1153
Service Response	Edge Server	$(t+2)E_{\mathbb{G}}+4BP+(t+2v-2)Mul_{\mathbb{G}}$	93	–	
User Revocation	Service Provider	–	–	64	76
Edge Revocation	Service Provider	$E_{\mathbb{G}}$	1	512	107



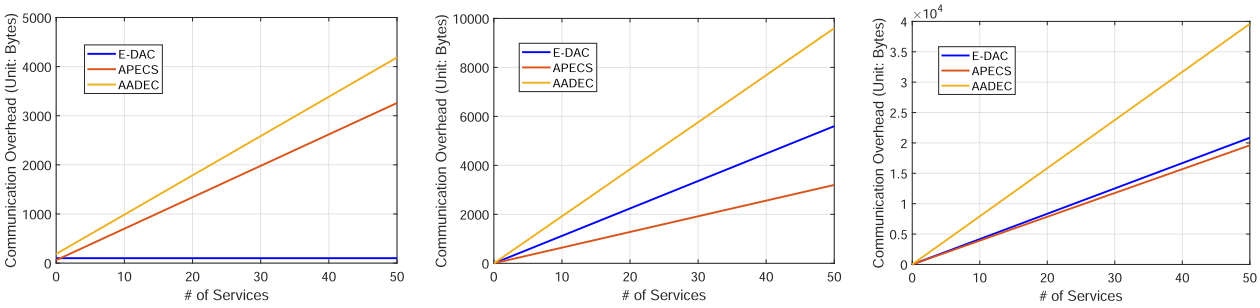
(a) Time Cost in Edge Server Registration (b) Time Cost in User Registration (c) Time Cost in Service Request

(d) Time Cost in Service Response (e) Time Cost in User Revocation (f) Time Cost in Edge Server Revocation

Fig. 2. Computational Overhead Comparison of E-DAC, APECS, and AADEC.

the service provider and multiple base stations to generate the authorization key. Assuming there are 5 base stations in APECS and AADEC, Fig. 2(a) demonstrates that E-DAC outperforms APECS and AADEC in edge server registration as the number of authorized services increases. Fig. 2(b) shows the performance of the service provider to generate credentials for registering users, and E-DAC is the most efficient one because it only requires several polynomial evaluation operations, which are less time-consuming than the exponentiation operations required in APECS and AADEC for creating attribute keys in ABE. For generating the service requests, a user needs to perform several exponentiation and bilinear pairing operations in E-DAC. This results in a heavier cost for our E-DAC than APECS, but E-DAC is still more efficient than AADEC. This overhead will not overwhelm the user device, as this operation only occurs once when the user needs to access a service. For handling concurrent service requests from users, we compare the time cost of service response for the edge server, as depicted in Fig. 2(d). Notably, with an increasing number of concurrent service requests, E-DAC demonstrates lower time cost on average for the edge server compared to APECS and AADEC. Additionally, Fig. 2(e) and (f) present the time cost for the service provider to revoke a user and an edge server in E-DAC, APECS, and AADEC, respectively. Both E-DAC and APECS are very efficient in user revocation. With an increasing number of edge servers under the same base station, in E-DAC, when an edge server is revoked, the service provider does not need to update the authorization keys of other edge servers. Conversely, in APECS and AADEC, the service provider must re-key other edge servers under the same base station. Therefore, E-DAC proves to be more efficient than APECS and AADEC in edge server revocation.

Furthermore, we present the communication overhead between the service provider, the edge server, and the user in fifth column of Table I. It is worth noting that the binary sizes of identity (ID_e or ID_u), service index, expiration time, and



(a) Communication Overhead for Edge Service Credentials (b) Communication Overhead for Service Access Credentials (c) Communication Overhead for Service Request

Fig. 3. Communication Overhead Comparison of E-DAC, APECS, and AADEC.

timestamp are all 64 bits. One notable feature of our E-DAC is that the edge service credential size remains constant, as shown in Fig. 3(a), thanks to the aggregation of the access keys of the services. On the contrary, in APECS and AADEC, the size of the edge service credential increases linearly with the number of authorized services and the number of base stations involved in service authorization. Fig. 3(b) and (c) illustrate the comparison of the size of service access credentials and service requests, respectively. Our E-DAC demonstrates shorter sizes for both service access credentials and service requests compared to AADEC, albeit slightly longer than APECS. The main reason is the secret sharing mechanism employed in creating service access credentials for users. While efficient in computation, it imposes a slightly higher burden on communication. In addition, although AADEC is less communication and computational efficient than E-DAC and APECS, it achieves user anonymity and auditability, which are not supported by E-DAC and APECS.

VII. Conclusion

In this paper, we have proposed an efficient and distributed access control framework (E-DAC) tailored for dynamic services in the PEC environment. By aggregating access keys of various services, E-DAC enhances flexibility and efficiency in service authorization. The incorporation of 0-RTT authentication significantly reduces network bandwidth costs and minimizes communication rounds between users and edge servers. Moreover, E-DAC achieves rapid mutual authentication without relying on a centralized authentication server, making it suitable for scenarios where user devices have short-lived connections to edge servers due to high mobility. Additionally, E-DAC supports the dynamics of edge servers through efficient service authorization and low-cost service revocation. Therefore, E-DAC is particularly suitable for the real-time applications, where frequent computations and low latency are critical, such as autonomous driving, industrial IoT, and AR/VR. For the future work, we will investigate the identity privacy leakage of users in authentication and develop anonymous distributed access control solutions for the dynamic PEC environment.

References

[1] L. Liu, C. Huang, D. Zhu, D. Liu, J. Ni, and X. S. Shen, "Secure and distributed access control for dynamic pervasive edge computing services," in *Proc. IEEE Glob. Commun. Conf.*, 2022, pp. 5487–5492.
[2] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," in *Proc. IEEE*, vol. 107, no. 8, pp. 1738–1762, Aug. 2019.
[3] R. Tourani et al., "Democratizing the edge: A pervasive edge computing framework," 2020, *arXiv: 2007.00641*.
[4] G. Luo et al., "Software-defined cooperative data sharing in edge computing assisted 5G-vanet," *IEEE Trans. Mobile Comput.*, vol. 20, no. 3, pp. 1212–1229, Mar. 2021.
[5] S. Xu, J. Ning, X. Huang, J. Zhou, and R. H. Deng, "Server-aided bilateral access control for secure data sharing with dynamic user groups," *IEEE Trans. Inf. Forensics Secur.*, vol. 16, pp. 4746–4761, 2021.
[6] G. Xu, H. Li, H. Ren, X. Lin, and X. Shen, "DNA similarity search with access control over encrypted cloud data," *IEEE Trans. Cloud Comput*, vol. 10, no. 2, pp. 1233–1252, Second Quarter, 2022.
[7] Y. Hou, S. Garg, L. Hui, D. N. K. Jayakody, R. Jin, and M. S. Hossain, "A data security enhanced access control mechanism in mobile edge computing," *IEEE Access*, vol. 8, pp. 136119–136130, 2020.
[8] Y. Liu, M. Xiao, S. Chen, F. Bai, J. Pan, and D. Zhang, "An intelligent edge-chain-enabled access control mechanism for IoV," *IEEE Internet Things J.*, vol. 8, no. 15, pp. 12231–12241, Aug. 2021.
[9] Q. Li, B. Xia, H. Huang, Y. Zhang, and T. Zhang, "TRAC: Traceable and revocable access control scheme for mhealth in 5G-enabled IIoT," *IEEE Trans. Ind. Inform.*, vol. 18, no. 5, pp. 3437–3448, May 2022.
[10] D. Yu, R.-H. Hsu, J. Lee, and S. Lee, "EC-SVC: Secure can bus in-vehicle communications with fine-grained access control based on edge computing," *IEEE Trans. Inf. Forensics Secur.*, vol. 17, pp. 1388–1403, 2022.
[11] H. Guo, W. Li, M. Nejad, and C.-C. Shen, "Access control for electronic health records with hybrid blockchain-edge architecture," in *Proc. Int. Conf. Blockchain*, 2019, pp. 44–51.
[12] D. Di Francesco Maesa, P. Mori, and L. Ricci, "Blockchain based access control," in *Proc. Distrib. Appl. Interoperable Syst.*, 2017, pp. 206–220.
[13] C. Huang et al., "Blockchain-assisted transparent cross-domain authorization and authentication for smart city," *IEEE Internet Things J.*, vol. 9, no. 18, pp. 17194–17209, Sep. 2022.
[14] H. Dang, T. T. A. Dinh, D. Loghin, E.-C. Chang, Q. Lin, and B. C. Ooi, "Towards scaling blockchain systems via sharding," in *Proc. ACM Int. Conf. Manage. Data*, 2019, pp. 123–140.
[15] S. Misra, R. Tourani, F. Natividad, T. Mick, N. E. Majd, and H. Huang, "AccConF: An access control framework for leveraging in-network cached data in the ICN-enabled wireless edge," *IEEE Trans. Dependable Secure Comput.*, vol. 16, no. 1, pp. 5–17, Jan./Feb. 2019.
[16] K. Xue et al., "A secure, efficient, and accountable edge-based access control framework for information centric networks," *IEEE/ACM Trans. Netw*, vol. 27, no. 3, pp. 1220–1233, Jun. 2019.

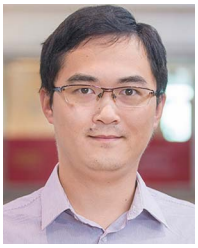
[17] S. Dougherty, R. Tourani, G. Panwar, R. Vishwanathan, S. Misra, and S. Srikanteswara, "APECS: A distributed access control framework for pervasive edge computing services," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2021, pp. 1405–1420.
[18] C.-K. Chu, S. S. Chow, W.-G. Tzeng, J. Zhou, and R. H. Deng, "Key-aggregate cryptosystem for scalable data sharing in cloud storage," *IEEE Trans. Parallel Distrib. Syst*, vol. 25, no. 2, pp. 468–477, Feb. 2014.
[19] W.-G. Tzeng and Z.-J. Tzeng, "A public-key traitor tracing scheme with revocation using dynamic shares," in *Proc. Public Key Cryptogr.*, 2001, pp. 207–224.
[20] M. Mukherjee, R. Matam, C. X. Mavromoustakis, H. Jiang, G. Mastorakis, and M. Guo, "Intelligent edge computing: Security and privacy challenges," *IEEE Commun. Mag.*, vol. 58, no. 9, pp. 26–31, Sep. 2020.
[21] J. Ni, K. Zhang, X. Lin, and X. Shen, "Securing fog computing for Internet of Things applications: Challenges and solutions," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 1, pp. 601–628, First Quarter 2018.
[22] P. Ranaweera, A. D. Jurcut, and M. Liyanage, "Survey on multi-access edge computing security and privacy," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 2, pp. 1078–1124, Second Quarter, 2021.
[23] Y. Li, Y. Yu, W. Susilo, Z. Hong, and M. Guizani, "Security and privacy for edge intelligence in 5G and beyond networks: Challenges and solutions," *IEEE Wireless Commun.*, vol. 28, no. 2, pp. 63–69, Apr. 2021.
[24] R.-H. Hsu, J. Lee, T. Q. Quek, and J.-C. Chen, "Reconfigurable security: Edge-computing-based framework for IoT," *IEEE Netw.*, vol. 32, no. 5, pp. 92–99, Sep./Oct. 2018.
[25] X. Li, T. Liu, C. Chen, Q. Cheng, X. Zhang, and N. Kumar, "A lightweight and verifiable access control scheme with constant size ciphertext in edge-computing-assisted IoT," *IEEE Internet Things J.*, vol. 9, no. 19, pp. 19227–19237, Oct. 2022.
[26] C. Feng, K. Yu, M. Aloqaily, M. Alazab, Z. Lv, and S. Mumtaz, "Attribute-based encryption with parallel outsourced decryption for edge intelligent IoV," *IEEE Trans. Veh. Technol.*, vol. 69, no. 11, pp. 13784–13795, Nov. 2020.
[27] X. Li, H. Tian, and J. Ning, "Secure online/offline attribute-based encryption for IoT users in cloud computing," in *Proc. Provable Secur.: 13th Int. Conf.*, Cairns, QLD, Australia, Springer, 2019, pp. 347–354.
[28] Z. Zhang, G. Huang, S. Hu, W. Zhang, Y. Wu, and Z. Qin, "FDO-ABE: A fully decentralized lightweight access control architecture for mobile edge computing," in *Proc. IEEE 6th Int. Conf. Comput. Commun. Syst.*, 2021, pp. 193–198.
[29] Y. Huang, J. Zhang, J. Duan, B. Xiao, F. Ye, and Y. Yang, "Resource allocation and consensus of blockchains in pervasive edge computing environments," *IEEE Trans. Mobile Comput.*, vol. 21, no. 9, pp. 3298–3311, Sep. 2022.
[30] R. Yang, F. R. Yu, P. Si, Z. Yang, and Y. Zhang, "Integrated blockchain and edge computing systems: A survey, some research issues and challenges," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 2, pp. 1508–1532, Second Quarter, 2019.
[31] L. Yu, M. He, H. Liang, L. Xiong, and Y. Liu, "A blockchain-based authentication and authorization scheme for distributed mobile cloud computing services," *Sensors*, vol. 23, no. 3, 2023, Art. no. 1264.
[32] S. Saha, D. Chattaraj, B. Bera, and A. Kumar Das, "Consortium blockchain-enabled access control mechanism in edge computing based generic Internet of Things environment," *Trans. Emerg. Telecommun. Technol.*, vol. 32, no. 6, 2021, Art. no. e3995.
[33] B. Jiang, Q. He, P. Liu, S. Maharjan, and Y. Zhang, "Blockchain empowered secure video sharing with access control for vehicular edge computing," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 9, pp. 9041–9054, Sep. 2023.
[34] D. Bhattacharyya et al., "Biometric authentication: A review," *Int. J. u-and e-Serv., Sci. Technol.*, vol. 2, no. 3, pp. 13–28, 2009.
[35] Q. Jiang, N. Zhang, J. Ni, J. Ma, X. Ma, and K.-K. R. Choo, "Unified biometric privacy preserving three-factor authentication and key agreement for cloud-assisted autonomous vehicles," *IEEE Trans. Veh. Technol.*, vol. 69, no. 9, pp. 9390–9401, Sep. 2020.
[36] X. Zhou, D. He, J. Ning, M. Luo, and X. Huang, "AADEC: Anonymous and auditable distributed access control for edge computing services," *IEEE Trans. Inf. Forensics Security*, vol. 18, pp. 290–303, 2022.
[37] S. Patranabis, Y. Shrivastava, and D. Mukhopadhyay, "Provably secure key-aggregate cryptosystems with broadcast aggregate keys for online data sharing on the cloud," *IEEE Trans. Comput.*, vol. 66, no. 5, pp. 891–904, May 2017.
[38] J. Camenisch and M. Stadler, "Efficient group signature schemes for large groups," in *Proc. Adv. Cryptol.: 17th Annu. Int. Cryptol. Conf.*, Santa Barbara, CA, USA, Springer, 1997, pp. 410–424.

Lingshuang Liu (Student Member, IEEE) received the BEng and MS degrees from the School of Computer Science and Engineering, University of Electronic Science and Technology of China, in 2014 and 2017, respectively. She is working toward the PhD degree with the Department of Electrical and Computer Engineering, University of Waterloo, Canada. Her research interests include security and privacy in communication networks and artificial intelligence.

Cheng Huang (Member, IEEE) received the BEng and MS degrees in information security from Xidian University, China, in 2013 and 2016, respectively, and the PhD degree in electrical and computer engineering from the University of Waterloo, ON, Canada, in 2020. He is currently an associate professor with the School of Computer Science, Fudan University. Before joining Fudan University, he was a research fellow with the Department of Electrical and Computer Engineering, University of Waterloo from 2020 to 2023. His research interests lie in the areas of security and privacy in vehicular networks, data security, and secure computation. He has published more than 60 papers in prestigious journals and conferences, including *IEEE Transactions on Dependable and Secure Computing*, *IEEE Journal on Selected Areas in Communications*, *IEEE Transactions on Vehicular Technology*, and *IEEE Transactions on Industrial Informatics*, and has received Best Paper Awards from ICCC'15, ICC'18, GLOBECOM'22, and ICCC'23. He serves as the associate editor for Peer-to-Peer Networking and Applications (Springer), the symposium chair of IEEE GLOBECOM'24, and has served as the publicity chair of ICA3PP'22, PST'23, SustainCom'23, and as a TPC member of many international conferences.

Dan Zhu (Member, IEEE) received the BS and PhD degrees from the School of Telecommunications Engineering and the School of Cyber Engineering, Xidian University, Xi'an, China, in 2017 and 2022, respectively. She is currently a associate professor with the School of Cybersecurity, Northwestern Polytechnical University, Xi'an China. Her research interests include applied cryptography, data security and privacy preservation.

Dongxiao Liu (Member, IEEE) received the PhD degree from the Department of Electrical and Computer Engineering, University of Waterloo, Canada, in 2020. He was a post-doctoral research fellow with the Department of Electrical and Computer Engineering, University of Waterloo from 2020 to 2023. He is now with the School of Computer Science and Engineering, University of Electronic Science and Technology of China. His research interests include security and privacy in communication networks and blockchain.



Jianbing Ni (Senior Member, IEEE) received the PhD degree in electrical and computer engineering from the University of Waterloo, Waterloo, Canada, in 2018. He is currently an assistant professor with the Department of Electrical and Computer Engineering, Queen's University, Kingston, Canada. His research interests are applied cryptography and network security, with current focus on edge computing, artificial intelligence, and Blockchain technology.

Xuemin (Sherman) Shen (Fellow, IEEE) received the PhD degree in electrical engineering from Rutgers University, New Brunswick, NJ, USA, in 1990. He is a University professor with the Department of Electrical and Computer Engineering, University of Waterloo, Canada. His research focuses on network resource management, wireless network security, Internet of Things, AI for networks, and vehicular networks. He is a registered professional engineer of Ontario, Canada, an Engineering Institute of Canada fellow, a Canadian Academy of Engineering fellow, a Royal Society of Canada fellow, a Chinese Academy of Engineering Foreign member, International fellow of the Engineering Academy of Japan, and a distinguished lecturer of the IEEE Vehicular Technology Society and Communications Society. He received "West Lake Friendship Award" from Zhejiang Province, in 2023, President's Excellence in Research from the University of Waterloo, in 2022, the Canadian Award for Telecommunications Research from the Canadian Society of Information Theory (CSIT), in 2021, the R.A. Fessenden Award in 2019 from IEEE, Canada, Award of Merit from the Federation of Chinese Canadian Professionals (Ontario), in 2019, James Evans Avant Garde Award, in 2018 from the IEEE Vehicular Technology Society, Joseph LoCicero Award in 2015 and Education Award in 2017 from the IEEE Communications Society (ComSoc), and Technical Recognition Award from Wireless Communications Technical Committee (2019) and AHSN Technical Committee (2013). He has also received the Excellent Graduate Supervision Award in 2006 from the University of Waterloo and the Premier's Research Excellence Award (PREA) in 2003 from the Province of Ontario, Canada. He was the president of the IEEE Communications Society, the vice president for Technical & Educational Activities, vice president for Publications, member-at-large on the Board of Governors, chair of the Distinguished Lecturer Selection Committee, and member of IEEE Fellow Selection Committee of the ComSoc. He served as the editor-in-chief of the *IEEE Internet of Things Journal*, *IEEE Network*, and *IET Communications*.