01 Jan 2022

Federated Secure Data Sharing by Edge-Cloud Computing Model*

Arijit Karati

Sajal K. Das
Missouri University of Science and Technology, sdas@mst.edu

Recommended Citation
A. Karati and S. K. Das, "Federated Secure Data Sharing by Edge-Cloud Computing Model*," *2022 IEEE Global Communications Conference, GLOBECOM 2022 - Proceedings*, pp. 1362 - 1367, Institute of Electrical and Electronics Engineers, Jan 2022.
The definitive version is available at https://doi.org/10.1109/GLOBECOM48099.2022.10001335

Federated Secure Data Sharing by Edge-Cloud Computing Model*

Arijit Karati
Department of Computer Science and Engineering
National Sun Yat-sen University
Kaohsiung, Taiwan
arijit.karati@mail.cse.nsysu.edu.tw

Sajal K. Das
Department of Computer Science
Missouri University of Science and Technology
Rolla, USA
sdas@mst.edu

GLOBECOM 2022 - 2022 IEEE Global Communications Conference | 978-1-6654-3540-6/22/$31.00 ©2022 IEEE | DOI: 10.1109/GLOBECOM48099.2022.10001335

***Abstract*—Data sharing by cloud computing enjoys benefits in management, access control, and scalability. However, it suffers from certain drawbacks, such as high latency of downloading data, non-unified data access control management, and no user data privacy. Edge computing provides the feasibility to overcome the drawbacks mentioned above. Therefore, providing a security framework for edge computing becomes a prime focus for researchers. This work introduces a new key-aggregate cryptosystem for edge-cloud-based data sharing integrating cloud storage services. The proposed protocol secures data and provides anonymous authentication across multiple cloud platforms, key management flexibility for user data privacy, and revocability. Performance assessment in feasibility and usability paves satisfactory results. Therefore, this work directs a new horizon to detailed new edge-computing-based data sharing services based on the proposed protocol for low latency, secure unified access control, and user data privacy in the modern edge enabled reality.**

***Index Terms*—data confidentiality, access control, privacy preserving authentication; user revocation; edge computing.**

I. Introduction

Technology has diligently optimized the gaps in human-machine interactions over several decades. Although existing technologies, including cloud, fog, and IoT, are stacked on top of each other to provide better remote data access, consumers and businesses alike are skeptical by the user experience due to high latency of specific actions performed on the cloud [1].

For instance, under a congested network condition, several users are interested in downloading a popular large file from the cloud. So, they make an individual request to the cloud and download the queried file. However, mobile applications may have communication latency due to the large distance between the cloud and the users. Further, cloud services operated by individual providers make *unified access control* infeasible due to several user accounts for multiple cloud platforms. Edge computing [2] as modern innovation improves user experience by putting hosting services, workloads, and vast quantities of data at the network's edge [3]. Edge devices, unlike the cloud, are much closer to end-users, enabling efficient analysis at the edge for upstream and downstream data on behalf of IoT and cloud, respectively. Fig. 1 depicts a scenario where users reside in a potent edge-assisted cloud (EC) environment, and data flow seamlessly from one cloud to users' proximal locations. The edge between the cloud and users execute specific activities to lower cloud overhead. The edge controller

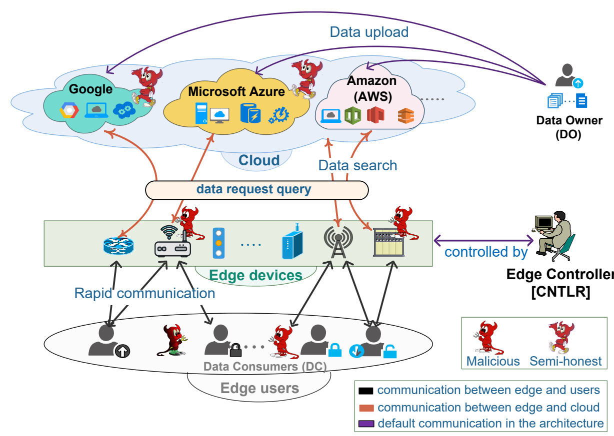


Fig. 1: Overview of the Edge-Cloud data sharing

(CNTLR) usually keeps track of such devices. When a user uploads a file to a specific cloud, an edge device, in response to a consumer request, downloads such file from the cloud and stores it at the edge for future use. The mobile edge computing (MEC) [4] server is often seen as CNTLR that aids cloud abilities at the edge of mobile networks. Thus, it boosts efficiency by outsourcing intensive calculations like credential management and data transfer to the edge. Further, it eases the distance data travel, lowering bandwidth and latency issues.

A. Security Requirements in the Edge-Cloud Data Sharing

Typically, edge devices are openly deployed in specific locations based on communication needs. In such a scenario, the following security issues could exist in edge-cloud model.

- **Confidentiality against cloud and edge devices:** Sensitive data is delivered publicly from the semi-honest cloud to the consumer through edge devices. Therefore, data should not be exposed to unwanted devices during transmission, and even attackers cannot bypass edge.

*This work was supported in part by the National Science and Technology Council (NSTC) under grant NSTC 111-2222-E-110-008-, the Ministry of Science and Technology grant MOST 110-2222-E-110-006- and assisted by CANSEC-LAB@NSYSU in Taiwan; and by the National Science Foundation (NSF) grants under award numbers 2008878 and 2030624 in the United States.

- **Federated access control by users and edge devices:** It is implemented when data is uploaded to a remote space with attributes and policies. An entity can access data if it complies with the underlying policy. Even though consumers have access to cloud data, none can divulge data if an edge device perceives a user as malevolent.
- **User authentication:** It is required during a) *file upload in remote space*, and b) *users' requests to download files*. In the latter case, the user's anonymity is a critical aspect.
- **User revocation:** Reneging all the access privileges of a deceitful user is an essential aspect that must be supported in real-time secure edge-cloud data sharing protocol.

In a typical EC model, an edge or intermediate device is with limited permission on deciding consumer access, while the owner has almost full control over whether or not to grant access. Thus, federated security and its privileges are confined to edge devices. Even if an edge device flags a user as malicious, cloud data might be bypassed the edge layer and decrypted by an insider attacker with its valid secret key. This is because the owner assigns decryption privileges, and the edge controls user subscriptions. Thus, the edge layer can not prevent harmful actions until the owner takes a specific action. Considering the above factors, we pose the question:

Can we design an edge-assisted cloud-based data sharing system where data is shared across multiple clouds under several file classes, and authorized subscribers can download them anonymously and securely through nearby edge devices without knowing the specific file location?

To the best of our knowledge, no such comprehensive EC model exist that addresses the question mentioned above.

B. Novel Contributions

The major contribution of this paper is a new key-aggregate cryptosystem for edge-cloud (KACEC) data sharing where data are downloaded (encrypted under distinct file classes) by several consumers from nearby edge devices. Within the context of this new framework, we contribute as follows:

- **Strong privacy protection:** The KACEC provides edge subscribers to retrieve cloud files via *anonymous authentication* using specific credentials. It is achieved by $\mathrm{SF_U}$.
- **Robust data confidentiality:** Our certificateless KACEC achieves *data confidentiality* under specified file classes.
- **Access control and user revocation:** The KACEC distributes specific activities ($\mathrm{SF_{KAC}}$) over multiple entities rather than one entity. Even if an attacker bypasses edge, it cannot decrypt data. Besides, the CNTLR *revokes* user's functionality if it finds any individual malicious activity.
- **No file location required to download:** The consumer needs not necessarily to know where the queried file exists. Thus, it enhances the privacy of the cloud and data, as well. Here, the named data networking (NDN) is considered to make the cloud anonymous to its consumer.

Nonetheless, the KACEC is compared with related schemes and shows superior CCA security and extensive safety features while integrating numerous cloud platforms, reducing latency with adequate computation, transmission, and storage costs.

The rest of this paper is laid out as follows: Section II lists prior works with potential limits, Section III exhibits technical preliminaries, and Section IV delves into the KACEC in-depth. Sections V and VI explain security and performance aspects of the KACEC. Finally, this paper concludes in Section VII.

II. Related Work

Due to the open nature of wireless links, the attacker may intercept, replay, and even tamper with the transmitted data [5]. The authors in [6] established a new cooperative paradigm for 5G networks using MEC resources to increase edge caching capability. For higher security in cloud mail applications, an ID-based broadcast proxy re-encryption (PRE) was proposed in [7]. However, it is secure against a chosen-plaintext attack (CPA), which is weaker than a chosen-ciphertext attack (CCA). In [8], the authors devised a revocable ID-based broadcast PRE to handle the key revocation issue. However, it is CPA resistant. Later, the authors in [9] designed a cross-layer monitoring system for locating and isolating components in multi-cloud deployments, alleviating service degradation issues. However, it fails to meet several safety standards including user anonymity and revocation. Similarly, a novel method proposed in [10] in a multi-cloud setting aims to enhance allocation trust while lowering communication delay; however, it does not solve user security and privacy issues. Although the PRE method in [11] improves user privacy, the re-key and re-ciphertext sizes do not remain consistent when the number of recipients grows. The authors in [12] proposed a clustering approach for Internet of vehicles applications based on edge computing for faster interaction. Besides, a low-energy edge-cloud collaborative architecture has been devised in [13] that is ideal for large-scale, time-sensitive face tracking systems. Recently, the authors in [14] designed a multi-authority and multi-cloud keyword search approach based on the consortium blockchain and attribute-based encryption. It is worth noting that the majority of the protocols listed above require remote storage. However, such approaches are built for a single use case and may not be relevant to other edge-based scenarios, posing integration and security hurdles, particularly when several nosy receivers receive data from multiple clouds.

III. Preliminaries

A. Notion of the Key-Aggregate Cryptosystem (KAC)

The ID-based KAC consists five following algorithms:

1) Setup: For security parameter k as input, it produces the public/master-secret key pair $(param, MSK)$.
2) KeyGen: On input ID_x, $params$ and MSK, it produces a secret key d for user x which later helps to build consumer's access rights for a set of classes $S_y \subset [1, n]$.
3) Encrypt: For input $param$, message M and class i, it returns ciphertext C that belongs to class i.
4) Extract: On input MSK, S_y, and an authorization token S'_y, it outputs the aggregate key SK_y to delegate the decrypting ability for the consumer.
5) Decrypt: For input SK_y, S_y, respective ciphertext C and file class i, it outputs M if $i \in S_y$.

B. Threat Model and Assumptions

Data can be compromised in many ways. For instance, uploaded data may be tampered with by several clouds, edge devices, end-users, or any untrusted entity during open transmission. Several attackers, as depicted in Fig. 1, engage in breaching the reliability are divided into two kinds, a) *malicious*, and b) *semi-honest* attackers. In a typical scenario, malicious activity may be observed at the end-user level, while the semi-honest adversary are at the edge and cloud. We list the adversary's capabilities and certain assumptions below.

- The CNTLR is expected to be a powerful, trustworthy entity that provides services to its subscribers via edge devices. Therefore, end-users are unable to engage directly with CNTLR during data transmission.
- User registration takes place over a secure channel using the transport layer security (TLS) protocol, whereas other services take effect over an insecure channel.
- An attacker ($\mathcal{A}$) is aware of the protocol design, however, it cannot reveal user secret key from user's private space.
- $\mathcal{A}$ can download cloud data bypassing the edge layer.

IV. Proposed KACEC Protocol for Secure Edge-Cloud Data Sharing

This section apprises the system architecture, including security functions (SFs) and devices' interactions. The architecture consists of several entities, namely the CNTLR, edge device (ED), NDN, cloud storage, data owner (DO), and data consumers (DCs). Here, clouds act as the sources of *file*s, while the NDN helps locate files anonymously. Nearby EDs facilitate every DC. The MEC server as CNTLR upholds the facility of performing certain logical operations on data and the credential management for both DCs and EDs. Before illustrating the data sharing process in the KACEC, we explain the cryptographic key computation and distribution briefly.

A. Key Issuance in the KACEC Protocol

The relation between different entities, including the key-generation center (KGC), is shown in Fig. 2. The DO uploads several files (categorized by file classes) accessible by the DCs. In the key distribution scenario, several entities participate in dealing with different keys required to download a *file* successfully. The KGC is a trusted party generates the global parameter as $params = \{g, g_1 = g^{\beta}, h = g^{\gamma}, T = e(g,h)^{\alpha}\}$ and secret key $MSK = \{\alpha, \beta, \gamma, s, F(\cdot)\}$ where α, β, γ, s are chosen at random. Note, $e : G_1 \times G_1 \rightarrow G_2$ is as admissible bilinear pairing [8] for two cyclic groups G_1 and G_2 with prime order q. To initiate communication, KGC sets $r = F(ID_{DC})$ and distributes DC's keys $usk = \{d = g^r, \{SK_i = h^{\frac{\alpha+r}{\beta+s^{-1}+H(i||ID_{DO})}}\}_{\forall i \in S_{DC}}\}$ based on a set of file classes S_{DC} granted by DO. For random e, KGC transmits a secret key $(g^e, s\gamma)$ through which CNTLR generates its control key $ctrl\_key = \{s\gamma, g^{\delta}\}$ and declares its public key $\{g^e, g^{e\delta}\}$. Now, CNTLR for every unique identity installs a secret key ($esk_j = e_j$) in the safe space of subscribed ED_j while makes g^{ee_j} as public key. All entities now participate in data transfer.

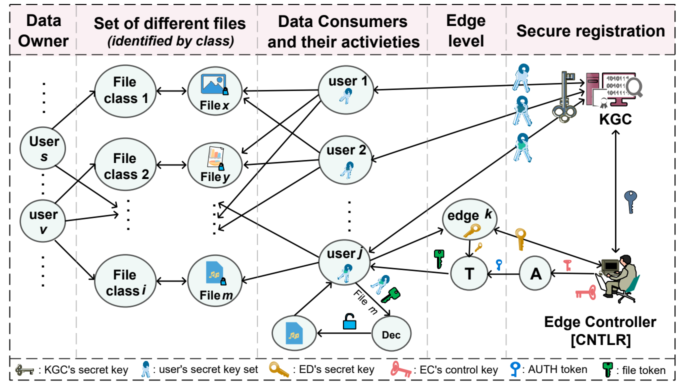


Fig. 2: Key management in the proposed KACEC protocol

B. Data Sharing in the KACEC Protocol

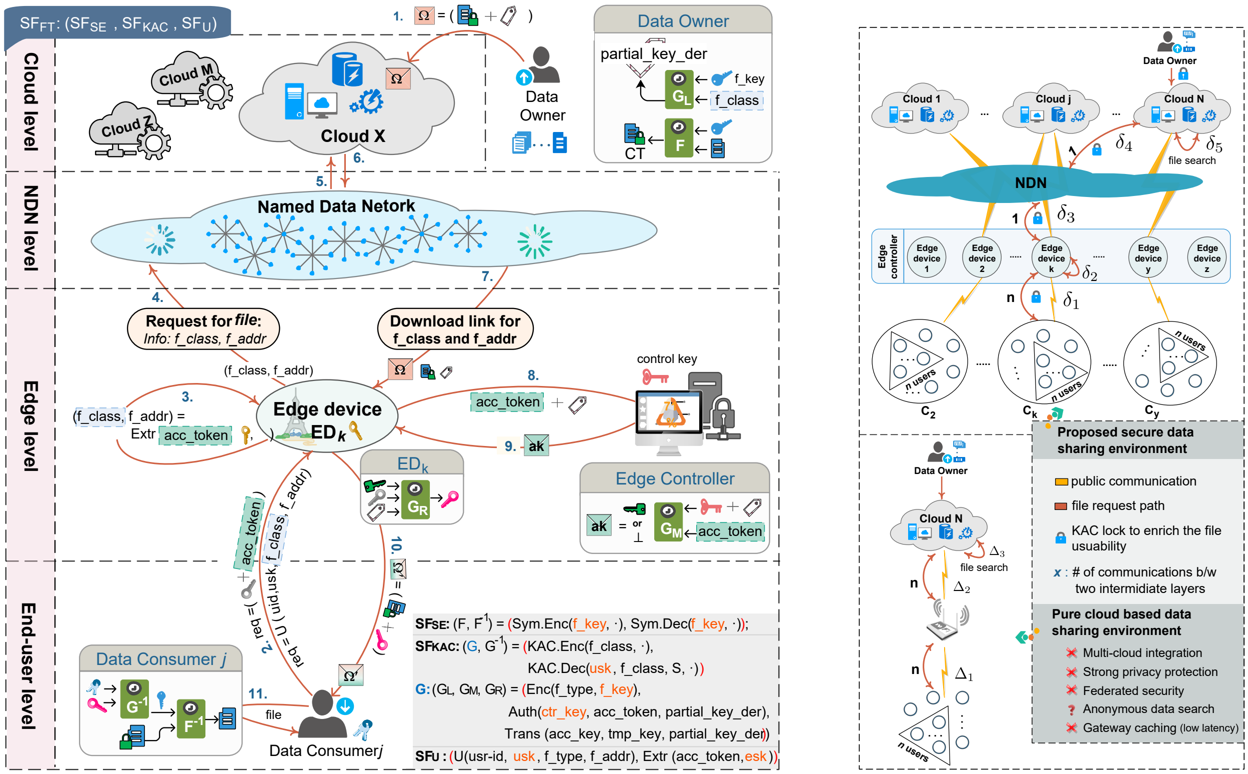
Secure file transmission shown in Fig. 3a is discussed below.

1) **File encryption, $\mathbf{\Omega = (CT, partial\_key\_der)}$**: The DO chooses a $L-$size *file* and a secret key f_key. It then uploads $\Omega = (CT, partial\_key\_der)$ to the cloud for a standard symmetric algorithm (denoted as SF_{SE}), where
   - ⋆ $\mathbf{CT \leftarrow F(f\_key, file)}$: The output is CT for 128-bit AES *encryption* function F with 128-bit f_key.

   Now, f_key is encrypted by the KAC under a specified file class mentioned as SF_{KAC}. Here, we divided the task of SF_{KAC} into three sub-tasks, named as G_L, G_M and, G_R. The first one is executed by the DO as below, and the remainings are defined in Steps 9 and 10.
   - ⋆ $\mathbf{partial\_key\_der \leftarrow G_L(f\_key, f\_class)}$: DO sets $C_1 = f\_key \cdot T^t$, $C_2 = \left[g_1 \cdot g^{f\_class}\right]^t$, $C_3 = h^t$ for $t \in_R Z_p^*$. It sends $partial\_key\_der = \langle C_1, C_2, C_3 \rangle$.
2) **File request, $\mathbf{req \leftarrow U(uid, usk, f\_class, f\_addr)}$**: When *file* is required to be downloaded from the cloud, consumer DC_j sends a request token $req = (tmp\_key, acc\_token)$ to ED_i as below:
   - ⋆ for $l \in_R Z_p^*$, sets $R = H(ID_j) \oplus H\left(e(g^{e\delta}, g^l)\right)$ and computes $K = e(g^{ee_i}, g^l)$, $X = g^{el}$ and $Y = h^l$.
   - ⋆ sets $acc\_token = \langle CT_K, X, Y \rangle$ where the ciphertext is $CT_K = AES(K, R \parallel f\_class \parallel f\_add)$.
   - ⋆ sets *temp_key* = $\langle Y, g^{l(\beta + f\_class)}, (SK_{f\_class})^{\frac{1}{l}}, d^{\frac{1}{l}} \rangle$.
3) **Extract, $\mathbf{(f\_class, f\_addr) \leftarrow Extr(acc\_token, esk)}$**: On such request, ED_k decrypts CT_K and collects file details (f_class and f_addr) for symmetric key $K = e(X, esk_k)$. It forwards f_addr to the NDN. Note that f_addr as cryptographic hash digest does not reveals the file location or the source cloud information.

4-7) **Data request for file_class *i* with (f_class, f_addr)**: ED_k transmits f_class and f_addr to the NDN to receive a download link for the same file (Ω) from the anonymous cloud to the requested ED_k anonymously.

8) Now, ED_k forwards $partial\_key\_der$ along with DC's acc_token to the CNTLR.
9) **Authorization, $\mathbf{op \leftarrow G_M(ctrl\_key, acc\_token, partial\_key\_der)}$**: If DC is revoked with its credential, i.e., $(R' \in RL) = TRUE$ where $R' = R \oplus e(X, g^{\delta})$, then



(a) Secure and anonymous data transmission in edge-cloud model

(b) Comparison with pure cloud computing

Fig. 3: Overview of the proposed KACEC protocol

CNTLR sets $op = \perp$. Otherwise, it sets $op = ak$ where token $ak = \langle R_1, Y_1 \rangle$ is generated as

$$\star\ R_1 = \left[(C_3)^{\frac{H(ID_j)}{s\gamma}} \right]^{e_i} \text{ and } Y_1 = \left[Y^{\frac{H(ID_j)}{s\gamma}} \right]^{e_i}$$

10) **Key Transform key_der $\leftarrow$ $\mathbf{G_R}$(acc_key, tmp_key, $\mathbf{\Omega}$):** If ak is $\perp$, ED_i treates the user as a revoked user and thus, *aborts* the session. Else, it runs G_R to generate the missing element as

$\star$ if $e\big(g^{l(\beta+f\_class)} \cdot Y_1^{\frac{1}{e_i}}, (SK_{f\_class})^{\frac{1}{l}}\big) \neq T \cdot e(Y, d^{\frac{1}{l}})$ – sets $key\_der = \perp$ and ABORT the connection.

$\star$ Else, sets $C_2' = C_s \cdot (R_1)^{\frac{1}{e_i}}$, $key\_der = \langle C_1, C_2', C_3 \rangle$.

Finally, it sends $\Omega' = (CT, key\_der)$ to DC_j.

11) **File decryption:** If the session is not aborted, then DC_j receives Ω'. It can reveal $file$ through KAC decryption (G^{-1}) and symmetric AES decryption (F^{-1}) as

$\star$ $f\_key \leftarrow G^{-1}(key\_der, usk)$: With valid secret usk for f_class, it gets $f\_key = C_1 \cdot \frac{e(d, C_3)}{e(C_2', SK_{f\_class})}$.

$\star$ reveals $file = F^{-1}(f\_key, CT)$.

The CNTLR tracks users' access through the capability list (CL) [15]. It works with a revocation list (RL) of suspicious users. When a new harmful action is detected, it updates CL and RL. The transmission order among entities is shown in Fig. 3a. The KAC *encryption* is shown through G in SF_{KAC}, wherein three functions are a) *data-only-encryption* G_L, b) *identity-inclusion* G_M, and c) *key-transformation* G_R. Now, the security followed by performance benefits are discussed.

V. Security Analysis of the KACEC Protocol

The KACEC's distinct keys are supplied at device registration to provide security during public communication. Besides, the below functions as shown in Fig. 3a protect user access.

- SF_{SE}: utilized to encrypt *file* using a symmetric key.
- SF_{KAC}: restricts unauthorized users from accessing the symmetric key encrypted under chosen file class.
- SF_U: aids in anonymous authentication for a file request.

Theorem 1. *Assume (t, q, ϵ)-Decision DHI holds in G_1. Then, the KACEC is (t', ϵ)-CCA secure in polynomial time $t' = t + O(q(qT_e + T_{bp}))$ with advantage $\epsilon' \geq \epsilon \cdot (1 - \frac{1}{q})^{q_k + q_e + q_d}$ where q_k, q_e, q_d are the KeyGen, Extract, Decrypt queries, and T_e, T_{bp} are time for exponentiation and pairings, respectively.*

Proof. Omitted security proof due to space constraints. □

VI. Performance Evaluation

The efficiency of the KACEC is illustrated by simulating the cryptographic operations at various levels. Fig. 3b depicts two scenarios where the former shows the KACEC, and the latter shows a typical cloud-based file sharing. To make comparisons easier, we consider n users download *file* in both the scenarios. Thus, the cloud's overhead in the KACEC is *one* for n users. In second scenario, however, it is n distinct connections due to multiple *file* requests from distinct users at different times. We denote δ_i as the time for initiating request and response between two entities in the first scenario while Δ_i is set for the

TABLE I: Benchmark time executed on several devices

Devices	**Curve type-A**				**AES cryptosystem with CTR**		**File download (for 1MB data)**
	$\mathbf{T_P}$	$\mathbf{T_{E_{G_1}}}$	$\mathbf{T_{E_{G_2}}}$	$\mathbf{T_{inv}}$	**Encryption (for 1MB data)**	**Decryption (for 1MB data)**	
PC/desktop†	4.20 ms	6.00 ms	0.56 ms	0.02 ms	0.69 ms	0.90 ms	18.09 ms
Laptop‡	14.00 ms	21.00 ms	2.00 ms	0.10 ms	6.15 ms	7.44 ms	186.67 ms
Raspberry PI*	230.00 ms	184.00 ms	32.50 ms	0.95 ms	54.05 ms	56.40 ms	1883.00 ms

TABLE II: Computation cost of different entities for L (=700 Megabytes) size *file*

Entity	**Tasks performed**	**Processing time**	**Total time**
PKG (device: †)	Compute a user's key for file class set S ($\lvert S\rvert = 10$)	0.06s	0.06s
Owner Side $[DO_x]$ (device: $\star$)	Encrypt L size *file*	38.70s	20 minutes
	Upload *file* to the Cloud	1178.00s	
Edge side $[ED_k]$ (devices: ‡, †))	Verify user request and other processing	0.18s	2 minutes
	Download *file* from the Cloud for a given file class i	130.70s	
Consumer side $[DC_j]$ (device: $\star$)	Generate AUTH token for file request (class=i)	1.96s	22.6 minutes
	Download *file* from the nearby ED_k	1318.00s	
	Decrypt locally and read *file*	39.70s	

latter. Thus, the required time in the KACEC is $\delta = \sum_{i=1}^{5} \delta_i$. Under the different level of secure computations in separately configured devices, we show that $\delta < \Delta(= \sum_{i=1}^{3} \Delta_i)$. The detailed experiment setup and the result are discussed now.

A. Experimental Setup

During simulation, we consider Google Drive cloud facilitator and two ASUS M840SA desktops (†Intel Core i7-8700 CPU@3.20GHz with 8GB RAM running on Ubuntu18.04LTS) as CNTLR and KGC, respectively. Besides, a SONY VPCEH- 15EN laptop (‡Intel Core i3-2310M CPU@2.10 GHz with 4GB RAM running on Ubuntu18.04LTS) as ED, and two Raspberry Pi-3 devices (*type B+, ARM Cortex-A53 CPU@1.4 GHz with 1 GB RAM running on Kali Linux) as DO and DC are chosen. Additionally, we provide much higher bandwidth, nearly 10 MB/s through LAN, at the ED side while setting a lower bandwidth, nearly 1MB/s via WiFi, for DC to view a suitable edge-cloud scenario. In this setting, CNTLR associated with KGC provides device registration and users' authorization. We evaluate the security overheads of the KACEC. For this, every intended DC initially contacts nearby ED to download $L-$size *file*. For the analysis purpose, we set L=700 MB (megabytes) as large *file*, and DO uploads the same in the cloud. We apply the Gradle plugin in ED to interact with the cloud. Without loss of generality, we used JPBC library [16] to execute certain operations (op), viz. bilinear pairing (P), modular exponentiation (E_X for group X), and modular inverse (inv), in the aforementioned devices. We discuss the runtime complexity of such operations (as T_{op}) now.

B. Result Discussion

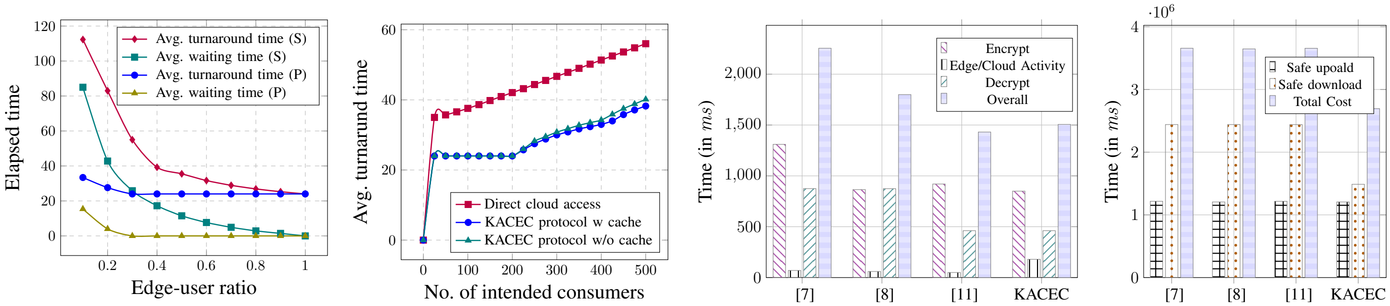
The execution time of the cryptographic operations is examined by considering the mean of thirty consecutive runs with discrete inputs. Table I shows the benchmark time of such operations run on different devices where the pairing is computed with preprocessing functionality. To achieve the faster pairing, we pick the Type-A curve with group size 512-bit and the embedding degree is 2. For the purpose of analysis, we assume $|G_1| = |G_2| = |G_T|$. Now, we explain the time required to download a $L-$size *file* by a DC considering the following computation, communication, and storage costs.

1) Computation Cost: Table II shows the incurred overheads. To encrypt a $L-$size *file*, DO spends nearly 38 seconds (*secs*), and performs 19 minutes (*mins*) to upload *file* in the cloud. On *file* request from DC, the edge layer executes several cryptographic operations for 183 milliseconds (*ms*) to verify and process a DC's request. On a successful validation, the DC downloads *file* in approximately 21 mins from the ED, and decrypts *file* contents successfully in 41 additional seconds.

2) Communication Cost: This factor considers the minimum bytes (B) required to transfer from the DC to the ED and further to the cloud, and vice versa. The PI device considers a group element 128B and an integer 20B long. Thus, the DO sends nearly 384B along with $L-$size encrypted *file* to the cloud. Now, if any DC wants to decrypt *file*, then it requests the ED with a file class along with an authentication token, which is 768B. For a valid user, the ED transmits approximately 384B with $L-$size during the public transmission to DC.

3) Storage Cost: Since the KACEC focuses on encrypted file sharing, the overhead considers the minimum space required to store various cryptographic keys. Here, the CNTLR and ED require 916B and 660B to store their public and private keys, respectively. Besides, DC needs 1920B to store the keys (for ten different classes) in its private memory to maintain a fine-grained access control during the file-sharing process.

Under the same configuration, DC_i spends nearly $\delta = 24$ mins in downloading and decrypting a $L-$size encrypted file using the KACEC (see Fig. 3b1), while nearly $\Delta = 35$ mins in direct communication with the cloud (see Fig. 3b2). Fig 4 discusses a detailed performance of the KACEC. Fig. 4a shows a performance orchestration both in sequential and parallel modes where *ratio* indicates the average load of a single ED for a total hundred different users. For instance, ratio 0.2 refers to twenty EDs processing requests of a hundred DCs with an average load of five. Consider, ED_j receives users' requests in every minute. Fig. 4b depicts a comparison between traditional cloud computing and the KACEC. For this, fifty EDs with parallel execution are considered. Here, square-marked line shows the overload of the multiple user requests directly to the cloud while circle-marked and triangle-marked depict the overhead of caching-enable and caching-disable modes of KACEC.



(a) The KACEC in sequential (S) and parallel (P) modes execution

(b) Potency of the KACEC over pure cloud computing

(c) Additional security overhead comparisons of related techniques

(d) Data latency of related schemes

Fig. 4: Performance measurement and comparison of the KACEC protocol for L-size (=700 Megabytes) remote *file*

TABLE III: Security features comparisons of related schemes

Schemes	Data encryption	User revocation	Anonymous authentication	Anonymous data search	Fine-grained access control
Work [7]	✓	✗	✗	✗	✗
Work [8]	✓	✓	✗	✗	✗
Work [11]	✓	✓	✗	✗	✗
KACEC	✓	✓	✓	✓	✓

✓: Achieved; ✗: Does not achieved. Note: security properties discussed in Sec. I-A

Note that the edge devices, rather than the users, communicate with the cloud in the KACEC. Thus, the cloud communicates with a fixed number of EDs − increasing availability − even if the number of intended users grows rapidly. Further, the KACEC is compared with existing works [7], [8], [11] under specific factors. As shown in Fig. 4c, the KACEC outperforms existing works in encryption and decryption, which are nearly 850 ms and 461 ms, respectively. Although the work [11] is the only one that requires lesser computation costs (about 1431 ms), it lacks a wide set of security aspects. Fig. 4d depicts that the latency (about 24 mins) is substantially lesser than others for $L-$size encrypted data. Besides, Table III compares features where the KACEC supports all the major security properties, such as encrypted data exchange, user revocation, anonymous authentication and search, and fine-grained access control. Hence, compared to methods in [7], [8], [11], the KACEC achieves a comprehensive set of security features (about 60% more) supporting multiple cloud platform integration, poses an adequate computation burden during data sharing, and upholds about 30% faster customer data delivery.

VII. Conclusion

This article introduced the KACEC data sharing protocol alleviating the security hurdles in the edge-cloud scenario. The KACEC allows multiple data owners to securely share multiple files based on the distinct file classes for multiple receivers in multiple remote spaces. It achieves several promising security elements: confidentiality, unified access control, anonymous authenticity, and revocation. Further, we achieved anonymous data search with NDN, allowing users to avoid remembering the exact file location. Besides, the performance of the KACEC is assessed under a suitable scenario and shows its benefits in latency, data security, storage, and access control. This is the first work exhibiting the secure file sharing aspect in edge-cloud computing that achieves higher security notions to the best of our understanding. In the future, we will extend this work by enforcing secure caching optimization and scalable load balancing at the edge in a fully-untrusted edge-cloud grid.

References

[1] E. C. Cejudo and M. S. Siddiqui, “An optimization framework for edge-to-cloud offloading of kubernetes pods in V2X scenarios,” in *2021 IEEE Globecom Workshops*. IEEE, 2021, pp. 1–6.
[2] P. Sun and A. Boukerche, “Toward the design of an efficient transparent traffic environment based on vehicular edge computing,” in *IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2021, pp. 1–6.
[3] R. Xie, Q. Tang, S. Qiao, H. Zhu, F. R. Yu, and T. Huang, “When serverless computing meets edge computing: Architecture, challenges, and open issues,” *IEEE Wireless Commun.*, 2021.
[4] S. Rani, D. Gupta, S. Garg, M. Jalilpiran, and M. S. Hossain, “Consumer electronic devices: Evolution and edge security solutions,” *IEEE Consum. Electron. Mag.*, 2021.
[5] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, “Mobile edge computing: A survey,” *IEEE IoT J.*, vol. 5, no. 1, pp. 450–465, 2018.
[6] K. Zhang, S. Leng, Y. He, S. Maharjan, and Y. Zhang, “Cooperative content caching in 5G networks with mobile edge computing,” *IEEE Wireless Commun.*, vol. 25, no. 3, pp. 80–87, 2018.
[7] P. Xu, T. Jiao, Q. Wu, W. Wang, and H. Jin, “Conditional identity-based broadcast proxy re-encryption and its application to cloud email,” *IEEE Trans. Comput.*, vol. 65, no. 1, pp. 66–79, 2016.
[8] G. Chunpeng, Z. Liu, J. Xia, and F. Liming, “Revocable identity-based broadcast proxy re-encryption for data sharing in clouds,” *IEEE Trans. Dependable Secure Comput.*, 2019.
[9] A. Sabbioni, A. Bujari, L. Foschini, and A. Corradi, “An efficient and reliable multi-cloud provider monitoring solution,” in *IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2020, pp. 1–6.
[10] A. B. Alam, Z. M. Fadlullah, and S. Choudhury, “A resource allocation model based on trust evaluation in multi-cloud environments,” *IEEE Access*, vol. 9, pp. 105 577–105 587, 2021.
[11] S. Maiti and S. Misra, “P2B: privacy preserving identity-based broadcast proxy re-encryption,” *IEEE Trans. Veh. Technol.*, vol. 69, no. 5, pp. 5610–5617, 2020.
[12] D. Zhang, C. Gong, T. Zhang, J. Zhang, and M. Piao, “A new algorithm of clustering AODV based on edge computing strategy in IOV,” *Wireless Networks*, vol. 27, no. 4, pp. 2891–2908, 2021.
[13] X. Liu, J. Yang, C. Zou, Q. Chen, X. Yan, Y. Chen, and C. Cai, “Collaborative edge computing with fpga-based cnn accelerators for energy-efficient and time-aware face tracking system,” *IEEE Trans. Computat. Social Syst.*, 2021.
[14] Q. Wu, T. Lai, L. Zhang, Y. Mu, and F. Rezaeibagha, “Blockchain-enabled multi-authorization and multi-cloud attribute-based keyword search over encrypted data in the cloud,” *Journal of Systems Architecture*, p. 102569, 2022.
[15] A. Stambouli and L. Logrippo, “Data flow analysis from capability lists, with application to rbac,” *Inf. Process. Lett.*, vol. 141, pp. 30–40, 2019.
[16] A. De Caro and V. Iovino, “jPBC: Java pairing based cryptography,” in *Proc. of the 16th IEEE Symposium on Computers and Commun.* Greece, June 28 - July 1: IEEE, 2011, pp. 850–855.