Privacy-Friendly Flexible IoT Health Data Processing with User-Centric Access Control

Khlood Jastaniah, Ning Zhang, and Mustafa A. Mustafa

Abstract—This paper proposes a novel Single and Multiple user(s) data Aggregation (SAMA) scheme designed to support privacy preserving aggregation of health data collected from users’ IoT wearables. SAMA also deploys a user-centric approach to support flexible fine-grain access control. It achieves this by deploying two key ideas. First, it uses multi-key Homomorphic cryptosystem (variant Paillier) to allow flexibility in accommodating both single and multi-user data processing as well as preserving the privacy of users while processing their IoT health data. Second, it uses ciphertext-policy attribute-based encryption to support flexible access control, which ensures users are able to grant data access securely and selectively. Formal security and privacy analyses have shown that SAMA supports data confidentiality and authorisation. The scheme has also been analysed in terms of computational and communication overheads to demonstrate that it is more efficient than the relevant state-of-the-art solutions.

Index Terms—IoT, wearable, Multi-key Homomorphic encryption, Attribute based encryption, Access control, Privacy.

I. INTRODUCTION

IoT wearable devices are becoming mainstream as people realize the benefits they bring to their life along with their low and affordable prices due to rapid development in sensors, wireless communication, and cloud computing technologies. These devices are equipped with sensors and communication capabilities to collect (in real-time) users’ health-related data (e.g., heart rate, oxygen saturation), activities (e.g., steps count, sleep quality), and environment (e.g., location, humidity) [1].

Modern healthcare systems can utilize data generated from IoT wearable devices to support analytic models. Such models could be used to provide services (i) to individuals, e.g., personalized treatments such as monitoring a patient remotely and diagnosing diseases early by detecting health anomalies, and (ii) to the wider public for purposes such as predicting the spread of disease by analysing data collected from multiple individuals [2], [3]. For instance, a study by Stanford Healthcare Innovation Lab [4] uses data collected from many commercial wearables, such as Apple Watch and Fitbit, to identify symptoms of people infected with Coronavirus at an early stage and also record the geographic spread of the virus.

Moreover, many data recipients, including healthcare providers, researchers, insurance companies, family members, and friends may need to have (or benefit from having) access to the results of these analytic models. Therefore, healthcare systems should support wearable data processing and sharing with multiple and diverse data recipients [5].

Healthcare provision via wearable devices has led to an increased number of applications that collect, store and analyse (usually with the assistance of cloud providers) user sensitive wearable data at an unprecedented scale and depth [2], [6], [7]. However, this approach comes with mounting concerns over users’ data privacy (data confidentiality). The concerns lead to two major issues. First, although data collection could be done via secure channels, processing of user wearable data is typically performed in plaintext and governed by service providers who may outsource/discard the wearable data or analytic results to third parties [8], [9]. Second, users have no control over who access and share the collected and processed data [10]–[12]. In addition, unauthorised exposure of personal health data violate the GDPR [13] and HIPPA [14] regulations that advocates for users’ privacy protection and access control. Therefore, it is important to achieve secure data processing and sharing, adopting user-centric approach, which gives access control over data in the hands of users rather than service providers.

There are already attempts to tackle the aforementioned issues which can be generalised into two approaches. The first approach is based on homomorphic encryption schemes, while the second approach uses attribute-based encryption schemes. Existing solutions related to the first approach either support secure data processing over data collected only from a single user [15]–[20] or only from multiple users [12], [21]–[26], but they do not efficiently support system models for both single and multiple user data processing scenarios. In addition, the solutions using the second approach [11], [12], [27]–[29] assume that data processing entities – typically third-party services providers – are trustworthy.

In modern healthcare systems, the enormous amount of wearable sensitive health data generated and some of the processing jobs are typically outsourced or delegated to third-party service providers, such as cloud providers due to user’s constrained devices. In such cases, measures should be in place such that any threats from the third-party service providers can be addressed. Moreover, data owners should define fine-grained access control to both raw data and computed/aggregated data and specify who can access which data items (i.e. user-centric access control). In addition, there are multiple entities (e.g., healthcare professionals, researchers) who require to process and access different sets of data of specific individuals and/or of a group of people for different
legitimate purposes [2], [12], and both types of data processing and access (single and multiple users data) should be supported and granted based on a user-centric approach [8]. Therefore, there is a need for secure data processing that can accommodate single and multiple user data cases, while supporting user-centric fine-grained data access capabilities.

To fill in this research gap, we propose a novel privacy-preserving Single And Multiple user data Aggregation (SAMA) scheme that supports single and multiple users data processing over encrypted data and realises data sharing with fine-grain access control based on a user-centric approach. To the best of our knowledge, this paper is the first attempt to combine single and multiple users data processing and sharing the processing results across multiple entities with the focus on a user-centric design approach. To this end, the novel contributions of this work are three-fold:

- The design of SAMA – a novel privacy-preserving scheme to support secure aggregation of data collected from both single and multiple users and secure data sharing with fine-grain access control based on a user-centric approach. The secure aggregation of data is ensured by using variant Paillier homomorphic encryption (VP-HE) scheme in a multi-key environment such that data from individual owners (users) are encrypted by multiple users’ public keys in a twin-cloud architecture and data processing can be carried out over the encrypted data. The fine-grained access control of the processing result is supported by using Ciphertext-policy Attribute-based Encryption (CP-ABE), which gives data owners full control of the access rights over their data.

- The investigation of the SAMA scheme both theoretically (in terms of security) and through simulation (in terms of computational and communication costs) – our results indicate that SAMA satisfies the specified set of security and privacy requirements with lower computational and communication cost in user and data recipients side compared with [30].

The rest of this paper is organised as follows. Section II discusses the related work. Sections III and IV show design preliminaries and main building blocks used in the design of SAMA. This is followed by a detailed design of the SAMA scheme in Section V. Sections VI and VII detail SAMA’s security/privacy analysis and performance evaluation, respectively. Finally, Section VIII concludes the paper. The acronyms used in the paper are shown in Table I.

### II. RELATED WORK

Efforts have already been made to preserve the confidentiality of user data while data is being processed by deploying different advanced cryptographic techniques. One of the most widely used techniques is Homomorphic Encryption (HE) which allows operations on encrypted data. There are mainly two types of HE schemes: Fully HE (FHE) and partial HE (PHE). FHE schemes support an arbitrary number of operations over encrypted data [31]. However, they are still impractical as they require high computational resources. Hence they are not suitable for use in wearable/portable devices that have limited computational capabilities. PHE schemes, on the other hand, support a limited number of operations, addition or multiplication, on encrypted data. Such PHE schemes (e.g., Paillier [32]) are suitable for resource constrained devices.

Some of the existing schemes that deploy HE [15]–[20] have considered secure processing of data provided only by one (single) user, while other schemes [12], [21]–[26] support secure processing of data coming only from different (multiple) users. However, in modern healthcare systems, in many cases, data recipients need to access the processing results of both single and multiple users’ data. In such cases, the use of the above schemes has limitations. Firstly, HE proposals cannot efficiently support both single and multiple users data processing scenarios. Secondly, HE cannot support data sharing with multiple data recipients who require access to the same processing result.

To support secure and user-centric access control, there are proposals [11], [12], [27]–[29] adopting ABE schemes [33]. These proposals allow users to choose who can access their data, hence supporting fine-grained access control and multi-user access. ABE schemes can be classified into two types: ciphertext-policy ABE (CP-ABE) [34] and key-policy ABE (KP-ABE) [35] schemes. The main difference between the two types is the following. In the CP-ABE scheme, access structures are embedded with ciphertexts and users’ attributes are embedded with the users’ private keys, while with the KP-ABE scheme, the access structure is associated with the private keys of users and the ciphertexts are associated with attributes. Therefore, with the KP-ABE schemes, users do not have control over who can access the data; they can only control attributes assignments [34]. ABE schemes, on their own, do not support computations over encrypted data.

There are some existing proposals which combine secure data processing with access control. Ding et al. [30], [36] proposed a flexible access control over the computation results of encrypted multiple users’ data by combining ABE with HE schemes. The computation supports addition, subtraction, multiplication, division, etc. However, these proposals do not efficiently support processing over data of both single and multiple user(s) nor user-centric access control. Ruj and Naya [37] combined Paillier HE with ABE to support privacy preserving data aggregation and access control in the smart grid. However, in their proposal, the aggregated data needs to be decrypted and then re-encrypted with an access policy by a trusted authority, hence this solution places unconditional

### TABLE I: Acronyms.

| Acronym | Meaning                                      |
|---------|----------------------------------------------|
| SAMA    | Single And Multiple User Data Aggregation    |
| DR      | Data Recipient                               |
| CSP     | Cloud Service Provider                       |
| KA      | Key Authority                                |
| HE      | Homomorphic Encryption                       |
| PE      | Paillier Encryption                          |
| VP-HE   | Variant Paillier Homomorphic Encryption       |
| ABE     | Attribute Based Encryption                   |
| CP-ABE  | Ciphertext-Policy Attribute Based Encryption |

### Acronym Meaning

- **SAMA**: Single And Multiple User Data Aggregation
- **DR**: Data Recipient
- **CSP**: Cloud Service Provider
- **KA**: Key Authority
- **HE**: Homomorphic Encryption
- **PE**: Paillier Encryption
- **VP-HE**: Variant Paillier Homomorphic Encryption
- **ABE**: Attribute Based Encryption
- **CP-ABE**: Ciphertext-Policy Attribute Based Encryption
trust on the data manager. Tang et al. [38] proposed privacy-preserving fog-assisted health data sharing that supports a flexible user-centric approach using ABE. Patients send the abnormal values encrypted by symmetric encryption scheme and define the access policy by encrypting the symmetric key with ABE. It also supports naïve Bayes disease classification over the encrypted data at the fog node; however, it does not effectively support processing over data from multiple users. Pang and Wang [39] propose privacy preserving data mining operations on outsourced data from multiple parties under multi-key environments using VP-HE. The proposal supports sharing of processed data only with a data recipient (miner); however, it does not support user-centric and fine-grained data sharing with multiple users.

In summary, the state-of-the-art research in privacy preserving data processing either focuses on the single user or multiple user(s) data processing; they do not support both use-cases systematically. Furthermore, there are limited efforts on exploring the integration of privacy preserving data processing with fine-grained user-centric access control to support secure data processing and secure data sharing access among multiple users. This paper aims to address this knowledge gap, to design a solution that can efficiently support both single and multiple user data processing and fine-grained data sharing in a user-centric manner while protecting users’ (data owners) data privacy.

III. PRELIMINARIES

In this section, we introduce the system and threat model, assumptions, notations, and design requirements of SAMA.

A. System Model

The system model used by SAMA consists of the following entities (see Fig. 1). Users are data owners who possess wearables and are willing to share the data collected from their wearables with various data recipients for their own personal benefits or for the collective benefit of society. Users’ wearable data is usually collected and shared via their smartphone (gateway). Data recipients (DRs) are data consumers who wish to utilise users’ wearable data in order to provide (personalised) services to users or society. Example DRs could be individuals such as the users themselves, their family members, friends, professionals (e.g., named GPs), organisations such as hospitals, research centers, insurance, or charities, etc. Two cloud service providers store and process data on behalf of users: Cloud A ($CSPA$) provides users with storage and processing for users’ data, and manages access requests, while Cloud B ($CSPB$) cooperates with $CSPA$ in data computations and access control. A Key authority ($KA$) plays the role of a key management organisation.

B. Threat Model

This section describes the threat model of the proposed SAMA scheme as follows. Users are trustworthy but curious. They make legitimate requests to access users’ data, but they may be curious to access or find out other users’ data. The CSPs are semi-honest (honest-but-curious) entities. They follow the protocol as per the specifications, yet they are curious about the sensitive information of users or any aggregated user data. The $KA$ is considered a trustworthy entity. It performs all its duties honestly and never colludes with any other entities. The external adversary bounded by computational resources (not having access to quantum computers) is considered as untrustworthy or even malicious. The external attackers may utilize different kinds of network eavesdropping attacks and/or modify the data in transit or try to gain unauthorized access in an attempt to disrupt the system or the cloud servers.

C. Assumptions

The following assumptions are considered in the SAMA design. The communication channels among all entities are encrypted and authenticated. $CSPA$ and $CSPB$ do not collude with each other or with any other entities as they have a legal responsibility to prevent leakage of the users’ sensitive data. All entities’ identities are verified by the key authority before obtaining the encryption and decryption keys.

D. Design Requirements

The proposed system should satisfy the following functional, security and privacy, and performance requirements.

1) Functional Requirements:

(F1) Flexible data processing: SAMA should support both single and multiple user(s) data aggregation using the same system model and without substantially increasing the computational and communication overhead.

(F2) Fine-grain access control: SAMA should support a flexible access policy for users and facilitate granting different access rights to a set of data recipients.

(F3) User-centric: each user should control who is authorized to access the raw data collected from their wearables as well as the aggregated data that includes their raw data.
2) Security and Privacy Requirements:
(S1) Data confidentiality: users’ raw and aggregated data should be protected from unauthorised disclosure.
(S2) Authorisation: only authorised DRs should be able to access users’ aggregated data based on the user-defined access policy.

3) Performance Requirements:
(P1) Efficient: SAMA should be viable for wearables which are devices with limited computational capabilities.

IV. BUILDING BLOCKS

This section reviews briefly the Paillier cryptosystem [32], the Variant-Paillier in Multikey cryptosystem [39], and CP-ABE [34], which are used in the SAMA scheme design. The notations used throughout the paper are presented in Table II.

A. Pailler Cryptosystem

Pailler cryptosystem [32] is a practical additive homomorphic encryption scheme proven to be semantically secure.

1) Pailler in Single-Key Environment: It consists of three algorithms: key generation algorithm (GenPE), encryption algorithm(EncPE), and decryption algorithm(DecPE).

- GenPE(k) → ppk, psk: Given a security parameter k, select two large prime numbers p and q. Compute n = p · q, and λ = lcm(p − 1, q − 1). Define L(x) = (x − 1)/n. Select a generator g ∈ Z∗ n. Compute μ = (Lg^λ mod n^2)^−1 mod n. The public key is ppk = (n, g) and the private key is psk = (λ, μ).

- EncPE(ppk, m) → c: Given a message m ∈ Z and a public key ppk = (n, g), choose a random number r ∈ Z∗ n, and compute the ciphertext c = EncPE(ppk, m) = gm^r · rn mod n^2.

- DecPE(psk, c) → m: Given a ciphertext c and a private key psk = (λ, μ), recover the message m = DecPE(psk, c) = L(c^λ mod n^2) · μ mod n.

2) Variant-Paillier in Multi-Key Environment: The variant Pailler scheme [39] is one of the recent variations of the Pailler cryptosystem. It is similar to the original scheme [32] with a slight modification in the key generation algorithm, which makes it compatible to work in multi-users environment by generating a different public-private key pair for each user with two trapdoor decryption algorithms. The scheme comprises four algorithms: key generation (GenVP), encryption (EncVP), decryption with a weak secret key (Decwpk), and decryption with a strong secret key (Decsk).

- GenVP(k) → vpk, wsk, ssk: Given a security parameter k, choose k + 1 small odd prime factors u, v1, . . . , vk and choose two large prime factors vp and vq in which p and q are large primes with the same bit length. Compute p and q as p = 2uvv1v2 . . . vivk + 1 and q = 2uvv1v2 . . . vivk + 1. Calculate n = p · q and λ = lcm(p − 1, q − 1). Choose t as a number or a product of multiple numbers from the set {v1, v2, . . . , vk}, and t|λ naturally exists. Choose a random integer g ∈ Z∗ n that satisfies gn^t = 1 mod n^2, and gcd(L(g^λ mod n^2), n) = 1. Define L(x) = (x − 1)/n. Compute h = g^nx×t mod n^2. The public key is vpk = (n, g, h), the weak secret key is wsk = t and the strong secret key is ssk = λ.

- EncVP(vpk, m) → c: Given a message m ∈ Z and a public key vpk = (n, g, h), choose a random number r ∈ Zn, and compute the ciphertext c = EncVP(vpk, m) = gm^r · hr mod n^2.

- WDecVP(wsk, c) → m: The decryption algorithm with a weak secret key decrypts only the ciphertext encrypted with the associated public key. Given wsk and c, the ciphertext can be decrypted as m = WDecVP(wsk, c) = L(c^λ mod n^2) · r mod n.

- SDecVP(ssk, c) → m: The decryption algorithm with a strong secret key decrypts the ciphertexts encrypted with any public key of the scheme. Given ssk and c, the ciphertext can be decrypted as m = SDecVP(ssk, c) = L(c^λ mod n^2) · μ mod n.

B. Ciphertext-Policy Attribute Based Encryption

The CP-ABE is a type of public-key encryption in which the ciphertext is associated with an access policy and user keys are dependent upon attributes to support fine-grained access control [39]. It consists of four main algorithms: a setup algorithm (Setup), encryption algorithm (EncABE), key generation algorithm (GenABE), and decryption algorithm (DecABE).

- Setup(s, U) → pk, mk: Given a security parameter s and a universe of attributes U, the setup algorithm outputs the public parameters pk and a master key mk.
V. THE SAMA SCHEME

In this section, we propose our novel secure data aggregation scheme, SAMA, that works on single and multiple user(s) data with flexible data sharing, adopting a user-centric approach. First, we give an overview of SAMA and explain the system initialisation before presenting it in detail.

A. Overview of the SAMA Scheme

The SAMA scheme mainly makes use of a combination of the Paillier HE and CP-ABE schemes and consists of three main phases: (i) user access policy setting, (ii) data uploading, and (iii) data access request and processing, as shown in Fig. 2.

At the user access policy setting phase, the system initialises itself, and defines a user-centric fine-grained access policy, allowing users to define two types of access policies: single (AP\textsubscript{S}) and multiple (AP\textsubscript{M}) user(s) data access policy and send them to CSP\textsubscript{A}. This allows CSP\textsubscript{A} to process and share users’ data with DR\textsubscript{s} according to users’ preferences. In the data uploading phase, every user encrypts their data with their VP-HE public key and sends the resulting ciphertext to CSP\textsubscript{A}.

During the data access request and processing phase, CSP\textsubscript{A} receives requests from DR to access the (aggregated) data of users. These requests are processed by the CSP\textsubscript{s} and the results are shared with the corresponding requesters. There can be three different types of requests, coming either by the users themselves for accessing their own data or from the DR\textsubscript{s} requesting data of a single user or multiple users.

Upon receiving a request from a user, the CSP\textsubscript{A} aggregates the user’s encrypted data and the result is sent back to the user. The user can then use their own VP-HE weak secret key to obtain their aggregated data. If the request is received by DR for a single user’s data, CSP\textsubscript{A} aggregates the user’s encrypted data, masks it, and sends the masked encrypted data to CSP\textsubscript{B}. CSP\textsubscript{B} then performs strong decryption to obtain the masked data, encrypts this result (masked aggregated data) with a Paillier public key and encrypts the Paillier private key using CP-ABE with the access policy AP\textsubscript{S}, and sends both ciphertexts to CSP\textsubscript{A}. However, if the request is received by DR for multiple users data processing, the process is slightly different. CSP\textsubscript{A} gets the encrypted data of users, masks the data and sends the masked data to the CSP\textsubscript{B}. CSP\textsubscript{B} then performs strong decryption on the received ciphertexts, aggregate them, encrypts the result with a Paillier public key, and encrypts the Paillier private key using CP-ABE with the access policy AP\textsubscript{M}. In both cases, CSP\textsubscript{B} sends both ciphertexts to CSP\textsubscript{A}, CSP\textsubscript{A} then performs de-masking on the received ciphertext and sends the encrypted result (aggregated data) and CP-ABE ciphertext to DR. Finally, the authorized DR who satisfies the access policy will be able to decrypt the CP-ABE ciphertext and obtain the Paillier private key to decrypt the ciphertext of the final result (users aggregate data).

B. System Initialisation

The system initialisation step comprises two phases: system parameters setup and cryptographic key generation and distribution. All the entities’ keys are listed in Table III.

1) System Parameters Setup: In this phase, system parameters of the three encryption schemes are set.

- VP-HE setup: The KA sets a security parameter $k$ and chooses two large prime numbers $p$ and $q$ such that $L(p) = L(q) = k$, $L$ is the bit length of the input data.
- Paillier setup: The KA selects the security parameter $k'$, such that $k' > k$. It then chooses two large prime numbers $p$ and $q$. Then, the key generation algorithm is initiated as explained in the Section IV-A1.

![Fig. 2: An overview of the SAMA scheme.](image-url)
Fig. 3: User access policy setting and data uploading phase.

- ABE setup: The KA generates security parameters $s$ and $U$ attributes, which are used to generate $pk$ and $mk$ using the Setup algorithm described in Section IV-B.

2) System Key Generation and Distribution: This phase is divided into three steps outlined below.
- VP-HE Key Generation: The KA generates a unique $ssk$ and distinct variant Paillier homomorphic public/private key pair $(vpk_i, wsk_i)$ for every user $U_i$, $i = 1, \ldots, N_U$, using the $KGen_{VP}$ algorithm described in Section IV-A2.
- Paillier Key Generation: The KA generates a distinct Paillier homomorphic public/private key pair $(ppk_j, psk_j)$, for each request that comes from the same or any $DR$, using the $KGen_{P}$ algorithm described in Section IV-A1.
- ABE Key Generation: The KA generates a distinct private key $sk_j$ for every $DR_j$, using $KGen_{ABE}$ as described in Section IV-B. $DR_j$ obtains $sk_j$ from the KA, which embeds her/his attributes/roles.

C. SAMA in Detail

The SAMA scheme consists of three main phases: (1) User access policy setting, (2) Data uploading, and (3) Data access request and processing.

1) User Access Policy Setting: This phase shown in Fig. 3 is usually performed at the setup stage. It allows users to set their access policy for data aggregation and sharing requirements and share it with $CSPA$. It includes three steps: a) define access policy, b) activate notifications, and c) update access policy.

a) Define access policy: Generally, the user defines two types of access policy: (i) single-user data aggregation and sharing access policy ($AP_S$) and (ii) multiple-users data aggregation and sharing access policy ($AP_M$).

(i) $AP_S$ allows users to control who can access the aggregated results of their own data. Therefore, only the authorized

$DR$ with specific attributes satisfying the access policy can have access to the final aggregated result.

(ii) $AP_M$ allows users to determine whether they agree their data to be aggregated with other users’ data and the aggregated result can be accessed by $DR$s. In other words, each user defines his/her sharing preferences and gives consent to allow use of their individual wearable data in aggregation along with other users’ wearable data. Note that $AP_M$ does not authorise $CSPA$ to share any specific individual raw data with anyone. It only allows $CSPA$ to use the encrypted data of users whose sharing preferences match with the attributes of $DR$s who have requested access to data.

b) Activate notification: Users can select to receive regular notifications, which is a summary of all single and multi-user data requests to access their data received by $DR$s. Through the summary, users can check how many data access requests were granted/rejected. This will also allow users to monitor who has requested access to their data and whose requests were granted/rejected. Regular notification can be switched on/off by the user and can also be set to be received as daily/weekly/monthly data access summaries. $CSPA$ is responsible to follow users’ notification selections.

c) Update access policy: The $CSPA$ provides the users the ability to update their access policy periodically or based on demand. Users also have the option to update their pre-defined access policies ($AP_S$ and/or $AP_M$) based on their notifications details.

2) Data Uploading: During this phase, users upload their data to $CSP$s regularly. More specifically, users encrypt their wearable data $m_i$ with their variant-Paillier public key, $vpk_i$, to obtain $C_{vpk_i} = Enc_{VP}(vpk_i, m_i)$ and send the encrypted data to $CSPA$. This phase is the same for single and multi-user aggregated data sharing, as shown in Fig. 3.

3) Data Access Request and Processing: In this phase, there can be three different types of data access requests for users’ aggregated data as follows: a) Users request access to their own (aggregated) data, b) $DR$s request access to aggregated data of a single user and c) $DR$s request access to aggregated data of multiple users. The requests coming from users are directly handled by $CSPA$, while the requests coming from $DR$s are handled by both $CSP$s.

a) User access request for own (aggregated) data: A user

Fig. 4: Data access request and processing phase - user request.
requests CSP\textsubscript{A} to aggregate his/her own encrypted wearable data and provide the processed result, as shown in Fig. 4. Upon receiving the request to aggregate \( N_{req} \) data points, CSP\textsubscript{A} aggregates the users’ data (i.e., it performs additive homomorphic operations by multiplying the encrypted data of the user) to get \( \sum_{i=1}^{N_{req}} m_i \text{ppk}_i = \prod_{i=1}^{N_{req}} \text{Cvpk}_i \), where \([\text{data}]\) denotes encrypted data. The result then is sent to the user. Then, the user can decrypt \( \sum_{i=1}^{N_{req}} m_i \text{ppk}_i \), with his/her own weak secret key \( \text{wsk}_i \) to obtain the aggregated data as \( \sum_{i=1}^{N_{req}} m_i \text{ppk}_i = \text{Dec}_{\text{ppk}_i}(\text{wsk}_i, (\sum_{i=1}^{N_{req}} m_i)_\text{ppk}_i) \).

b) DR access request for single-user data processing: A DR requests access to the aggregated data of a (specific) single user. For example, a doctor requires access to the aggregated data of a specific patient to monitor his/her health condition. The aggregated data can be accessed only by DRs (e.g., doctors, friends, etc) whose attributes satisfy the fine grain access policy \( AP\textsubscript{S} \) set by the user. This phase, as shown in Fig. 5, is divided into the following five steps:

(i) Handling DR request: After a DR has issued a request to access the aggregated data of a single user, the CSP\textsubscript{A} performs the same additive homomorphic operations, as in Step a) explained above. The result is a ciphertext of the aggregated data: \( \sum_{i=1}^{N_{req}} m_i \text{vpk}_i \).

(ii) Masking: CSP\textsubscript{A} then masks the aggregated data. More specifically, it generates a random number \( r_U \), and encrypts it with the user’s VP-HE public key, \( \text{vpk}_i \), to obtain \( \text{Enc}_{\text{vpk}}(r_U, \text{vpk}_i) \). The ciphertext \( \text{Enc}_{\text{vpk}}(r_U, \text{vpk}_i) \) is multiplied with the ciphertext of the aggregated data \( \sum_{i=1}^{N_{req}} m_i \text{vpk}_i \), to get a ciphertext of the masked aggregated data \( \sum_{i=1}^{N_{req}} m_i + r_U \text{vpk}_i = \sum_{i=1}^{N_{req}} m_i \text{vpk}_i \). The result is then sent to CSP\textsubscript{B} along with the \( AP\textsubscript{S} \) set by the user.

(iii) Preparing the processing result: Upon receiving the result, CSP\textsubscript{B} decrypts it using its strong decryption key \( \text{ssk} \) to get the masked aggregate data \( \sum_{i=1}^{N_{req}} m_i + r_U \text{vpk}_i = \text{Dec}_{\text{vpk}}(\text{ssk}, \sum_{i=1}^{N_{req}} m_i + r_U \text{vpk}_i) \). Then, a new Paillier key pair \( \text{ppk}_j, \text{psk}_j \) is generated by KA (based on CSP\textsubscript{B} request) and the new key pair is sent back to CSP\textsubscript{B}. The new Paillier public key, \( \text{vpk}_j \), is used to encrypt the masked aggregated data to get \( \sum_{i=1}^{N_{req}} m_i + r_U \text{vpk}_i = \text{Enc}_{\text{vpk}}(\text{ppk}_j, \sum_{i=1}^{N_{req}} m_i + r_U \text{vpk}_i) \), while the new Paillier private key \( \text{psk}_j \) is encrypted by the user defined access policy \( AP\textsubscript{S} \) to get \( \text{Enc}_{\text{psk}}(\text{psk}_j, AP\textsubscript{S}) \). Finally, the two generated ciphertexts \( (\sum_{i=1}^{N_{req}} m_i + r_U \text{vpk}_i, \text{Enc}_{\text{psk}}(\text{psk}_j, AP\textsubscript{S})) \) are sent to CSP\textsubscript{A}.

(iv) De-masking: When CSP\textsubscript{A} receives the two ciphertexts, it initiates the de-masking process. It encrypts the random number \( r_U \) (used previously in the masking process) with \( \text{ppk}_j \) to obtain \( \text{Enc}_{\text{ppk}}(r_U, \text{ppk}_j) \). Then, CSP\textsubscript{A} calculates the additive inverse of \( \sum_{i=1}^{N_{req}} m_i \text{vpk}_i \), generating \( \text{Dec}_{\text{ppk}}(\text{ppk}_j, \text{vpk}_i) \). Finally, it de-masks the aggregated data as follows: \( \sum_{i=1}^{N_{req}} m_i \text{vpk}_i = \sum_{i=1}^{N_{req}} m_i + r_U \text{vpk}_i \).

(v) DR access the processing result: DR can access the processing result only if the DR’s key attributes satisfy the user’ \( AP\textsubscript{S} \). Hence, DR can decrypt and obtain \( \text{psk}_j \) by using its ABE secret key \( \text{psk}_j = \text{Dec}_{\text{ABE}}(\text{pk}, \text{psk}_j, AP\textsubscript{S}, \text{sk}) \). Finally, it uses \( \text{psk}_j \) to obtain the initially requested aggregated data of the user: \( \sum_{i=1}^{N_{req}} m_i = \text{Dec}_{\text{ABE}}(\text{pk}, \text{psk}_j, \sum_{i=1}^{N_{req}} m_i \text{ppk}_i) \).

c) DR access request for multi-user data processing: A DR requests access to aggregated data of multiple users. For example, a researcher may require access to the aggregated data of a specific set of patients (users) who, for instance, suffer from the same disease. The aggregated data can be accessed only by DRs whose attributes satisfy the fine grain access policy \( AP\textsubscript{M} \) of the users whose data is requested. This phase is also shown in Fig. 5 and consists of the following steps.

(i) Handling DR request: Upon receiving a request to access aggregated data of multiple users, the CSP\textsubscript{A} initiates the process by comparing users’ \( AP\textsubscript{M} \) with DR attributes. It then selects users whose \( AP\textsubscript{M} \) matches with DR request. For simplicity, let us assume that CSP\textsubscript{A} selects \( N \) users.

(ii) Masking: CSP\textsubscript{A} starts the masking process by generating a random number for every user’s data used in the aggregation. It then encrypts these generated random numbers with the corresponding users’ variant Paillier public keys, \( \text{vpk}_i \), generating \( \text{Enc}_{\text{vpk}}(r_U, \text{vpk}_i) \). Next, each encrypted random number is multiplied with the respective user’s encrypted data, \( \text{m}_i \text{vpk}_i \), to obtain \( \text{m}_i + r_U \text{vpk}_i = \text{m}_i \text{vpk}_i * r_U \text{vpk}_i \). Finally, the \( N \) masked ciphertexts are sent to CSP\textsubscript{B} along with the \( AP\textsubscript{M} \) set by the user for further processing.

(iii) Preparing the processing result: this step consists of the outlined sub-steps below:

- The CSP\textsubscript{B} decrypts all the received masked ciphertexts with the variant Paillier strong secret key \( \text{ssk} \) to obtain the individual users’ masked data: \( \text{m}_i + r_U \text{vpk}_i = \text{Dec}_{\text{vpk}}(\text{ssk}, \text{m}_i + r_U \text{vpk}_i) \). Then, it performs an addition operation to get the masked aggregation as follows: \( \sum_{i=1}^{N} \text{m}_i + \sum_{i=1}^{N} r_U \text{vpk}_i = \sum_{i=1}^{N} \text{m}_i + r_U \).
- KA generates a new Paillier public-private key.
The functional requirements achieved by SAMA in comparison with related schemes [30], [37]–[39] are summarized in Table IV. Compared to these schemes, SAMA achieves all the specified functional requirements.

### D. Functional Requirements Comparison

The functional requirements achieved by SAMA in comparison with related schemes [30], [37]–[39] are summarized in Table IV. Compared to these schemes, SAMA achieves all the specified functional requirements.

### VI. SECURITY ANALYSIS

In this section, we perform a security analysis which includes the security of the cryptosystems used (Paillier, variant Paillier, and CP-ABE), the security of the SAMA scheme, and the security requirements of SAMA.

#### A. Security of the Cryptosystems

The security of the Paillier cryptosystem [32] depends on the hardness of the Composite Residuosity Class Problem in the standard model. The scheme is semi-secure against chosen-plaintext attack as the Decisional Composite Residuosity assumption holds. The variant Paillier [39] is similar to the Paillier encryption with a slight change in the key generation algorithm (described in Section IV-A2). Hence, its security follows directly from the security of the Paillier cryptosystem, which is proven to satisfy the semantic security in the standard model under the assumption of the intractability of the Composite Residuosity Class Hard problem [39]. Moreover, the CP-ABE is secure under the generic elliptic curve bi-linear group and random Oracle model assumptions [34]. Therefore, the SAMA scheme builds its security on the proven security of the Paillier, variant Paillier, and CP-ABE cryptosystems.

#### B. Security of the SAMA Scheme

The security analysis of the SAMA scheme is based on the simulation paradigm with the presence of semi-honest (honest-but-curious and non-colluding) adversaries. To prove that the execution view of the IDEAL world is computationally indistinguishable from the execution view of the REAL world, we construct four simulators (Sim_U, Sim_CSPA, Sim_CSPB, and Sim_DRI), which represents four entities U, CSPA, CSPB, and DR. These simulators simulate the execution of the following adversaries Adv_U, Adv_CSPA, Adv_CSPB, and Adv_DRI that compromise U, CSPA, CSPB, and DR, respectively. Note that KA is excluded as it is assumed to be a trustworthy entity.

**Theorem 1.** The SAMA scheme can securely retrieve the aggregation result plaintext of the addition computations over encrypted data in the presence of semi-honest adversaries.

**Proof:** We prove the security of the SAMA scheme by considering the case with two data inputs.

1) **Sim_U:** The Sim_U encrypts the provided inputs $m_1$ and $m_2$ using VP-HE and returns both ciphertexts to Adv_U. The simulation view of the IDEAL world of Adv_U is computationally indistinguishable from the REAL world view owing to the semantic security of VP-HE.

2) **Sim_CSPA:** The Sim_CSPA simulates Adv_CSPA in single and multiple user(s) data processing scenarios. In the single-user data case, Sim_CSPA multiplies the provided ciphertexts and then encrypts a random number $r$ with VP-HE. Next, it multiplies the encrypted random number with the result of the multiplication of the ciphertexts. Later, the same random number $r$ is encrypted with the public key of the Paillier scheme and its ciphertext is raised to $n-1$ and multiplied with the given ciphertext. In the multiple users data case, Sim_CSPA generates two random numbers $r_1$ and $r_2$, encrypts them with the public key of the VP-HE and multiplies the encrypted random numbers with the ciphertexts (encrypted $m_1$ and $m_2$), respectively. Later, the same random numbers are encrypted with the public key of the Paillier scheme, and the results are raised to $n-1$ and multiplied with the given ciphertext. In both cases, the Adv_CSPA receives the output ciphertexts from Sim_CSPA. Therefore, the REAL and IDEAL views of Adv_CSPA are computationally indistinguishable owing to the semantic security of VP-HE and Paillier encryption.

3) **Sim_CSPB:** The execution view of CSP_B in the REAL world is given by both ciphertext of $(m_1+r_1)$ and $(m_2+r_2)$, which are used to obtain $m_1+r_1$ and $m_2+r_2$ by executing decryption with the strong secret key on these ciphertexts ($r_1$ and $r_2$ are random integers in $Z_n$). The execution view of

| Requirement                        | Sim_U | Sim_CSPA | Sim_CSPB | Adv_U |
|------------------------------------|-------|----------|----------|-------|
| Single-user data processing        | ✓     | ✓        | ✓        | ✓     |
| Multiple users data processing     | ✓     | ✓        | ✓        | ✓     |
| Fine-grain access control          | ✓     | ✓        | ✓        | ✓     |
| User-centric                       | ✓     | ✓        | ✓        | ✓     |
CSP_B in the IDEAL world has two ciphertexts randomly selected in the Z_p^2. The Sim_{CSP_B} simulates Adv_{CSP_B} in both single and multiple user(s) data processing scenarios. In the single-user data case, Sim_{CSP_B} simulates Adv_{CSP_B} as follows. The Sim_{CSP_B} runs the strong decryption algorithm and obtains m'_1 + m'_2 + r'_1 and then the decryption result undergoes further encryption by the public key of Pailler encryption to obtain a new ciphertext. In the multiple users data case, Sim_{CSP_B} runs the strong decryption algorithm and obtains m'_1 + r'_1 and m'_2 + r'_2. Then, the Sim_{CSP_B} aggregates the decryption results, and then the aggregated result is further encrypted by Pailler encryption public key to obtain a ciphertext. Next, in both cases, a randomly generated number is encrypted with CP-ABE. Then, the two ciphertexts (generated by the Pailler and CP-ABE schemes) are provided as a result by Sim_{CSP_B} to Adv_{CSP_B}. These ciphertexts are computationally indistinguishable between the REAL and IDEAL world of Adv_{CSP_B} since the CSP_B is honest and the semantic security of VP-HE and Pailler cryptosystem, and the security of CP-ABE.

4) Sim_{DR}: The Sim_{DR} randomly selects chosen ciphertexts (besides not having access to challenged data), decrypts, and sends them to Adv_{DR} to gain data information. The view of the Adv_{DR} is the decrypted result without any other information irrespective of how many times the adversary access the Sim_{DR}. Due to the security of CP-ABE and the semantic security of the Pailler scheme, both REAL and IDEAL world views are indistinguishable. Since the user data encryption process and DR decryption process are common for both single and multi-user data processing in the SAMA scheme, the security proof of Adv_{V} and Adv_{DR} is common for both single and multi-user scenarios.

C. Analysis against Security Requirements

1) Data Confidentiality: Every user encrypts his/her data using his/her VP-HE public key vpk_i. CSP_A then performs homomorphic addition operation over encrypted data, and delivers the processing result ciphertext with the encrypted private Pailler key psk using CP-ABE to the DR. Only authorized DRs can obtain psk and hence have access to the user data. Furthermore, the SAMA scheme conceals users’ raw data by adding random numbers at CSP_A, i.e., masking the processed data, hence preserving the privacy of the user(s) data at CSP_B. Moreover, the Pailler cryptosystem is semantically secure and the CP-ABE is secure under the generic elliptic curve bi-linear group model as discussed in VI-A. In addition, the communication channels among all the entities (user, CSP_A, CSP_B, and DR) are secure (e.g., encrypted using SSL). Therefore, based on all of the above, only the authorised entities (i.e., the user or DR) can access the processing result and all the unauthorised internal or external entities who might eavesdrop messages sent and/or collect information can only access the ciphertext of the users (satisfying (S1)).

2) Authorisation: SAMA uses CP-ABE to implement secure fine grain access control, where the processing result is encrypted by the user defined access policies and the decryption key is associated with the attributes of the recipients. The user-centric access policy has been applied in the design of the SAMA scheme, which allows users to define their access policies to securely and selectively grant DRs access to the processing result. Thus, the processing result is encrypted using APs and APs, which are access policies set by users to determine their sharing preferences for sharing the single and multiple users data processing results. Hence, the private key of DR (sk) is required to decrypt the encrypted processing result using CP-ABE and only the authorised DR who satisfies the access policy can access the key and thereby decrypt the processing result. Thus, using CP-ABE, SAMA provides user-centric fine grain access control and only authorized DR can access the processing result (satisfying (S2)).

VII. Performance Evaluation

In this section, we evaluate the performance of the SAMA scheme in terms of the computational complexity and communication overheads incurred among all entities in the system. We also compare the performance of SAMA with the performance of the most relevant work [30].

A. Computational Complexity

The computationally expensive operations considered in the SAMA scheme are the modular exponentiation and multiplication operations, denoted as ModExp and ModMul, respectively. We ignore the fixed numbers of modular additions in our analysis as their computational cost compared to ModExp and ModMul is negligible. In our analyses we also use the following parameters: BiPair is the cost of a bilinear pairing in ABE; |γ| + 1 is the number of attributes in the access policy tree and θ is the number of attributes needed to satisfy the access policy. Furthermore, we divide the complexity of SAMA to computations related into HE for data aggregation and ABE for access control.

1) Computational Complexity of HE Data Aggregation: In our analysis, we split the computational complexity into four parts: the complexity at each of the entities.

Computations at User Side: This is a common step for single and multiple user(s) data cases. At each reporting time slot, each user encrypts their data by their VP-HE public key vpk_i to generate a ciphertext used for data processing/analyzing. This encryption requires two modular exponentiation operations, hence the computational complexity at the user side is:

\[ 2 \ast \text{ModExp} \]

Computations at CSPs: This includes operations performed by CSP_A and CSP_B. As these operations are slightly different for the single and multiple user(s) data processing scenarios, we analyse them separately.

For the single-user data processing case, CSP_A performs additive homomorphic encryption on the received user ciphertexts \((N_m - 1) \ast \text{ModMul}\), generates a random number \(r\), encrypt it with the user’s VP public key vpk_i \((2 \ast \text{ModExp})\), multiplies the results of the homomorphic addition with the encrypted random number \(\text{ModMul}\) and sends it to CSP_B. Next, CSP_A re-encrypts the generated random number \(r\) by vpk_i \((2 \ast \text{ModExp})\), calculates the additive inverse of \(r\) \((\text{ModExp})\), and then multiplies it with...
the encrypted processing result \((\text{ModMul})\) to remove the masking from the original data. Thus, \(CSPA\) performs in total: \((N_m + 1) * \text{ModMul} + 5 * \text{ModExp}\). \(CSP_B\) performs strong decryption using \(ssk\) on the received ciphertexts \(2 * \text{ModExp} + \text{ModMul}\). It then encrypts the aggregated masked result with \(ppk_j\) (\(2 * \text{ModExp}\)), and encrypts \(psk_j\) with CP-ABE using \(AP_{S}\) \(((|\gamma| + 1) * \text{Exp})\). Hence, \(CSP_B\) performs in total: \(4 * \text{ModExp} + \text{ModMul} + (|\gamma| + 1) * \text{Exp}\).

In total, the computational cost at \(CSPs\) in a single-user data processing case is: \((N_m + 2) * \text{ModMul} + 9 * \text{ModExp} + (|\gamma| + 1) * \text{Exp}\).

For the multiple users data processing case, \(CSPA\) generates a random number for every user’s data \((N\ users)\), encrypts them using the VP public key of the corresponding user, \(vpk_i\), \((N * 2 * \text{ModExp})\), and then multiplies the resulting ciphertexts with the ciphertexts received from users \((N * \text{ModMul})\). Later, it aggregates all the generated random numbers, encrypts it using \(ppk_j\) \((2 * \text{ModExp})\), calculates the additive inverse of the aggregation result \((\text{ModExp})\), and then multiplies the aggregation result ciphertext with the received ciphertext from \(CSP_B\) \((\text{ModMul})\) to remove the masking from the original data. Thus, the computational cost of \(CSPA\) in multiple users data processing case is: \((N * 2 + 3) * \text{ModExp} + (N + 1) * \text{ModMul}\). \(CSP_B\) performs strong decryption using \(ssk\) for all \(N\) received ciphertexts \((N * (2 * \text{ModExp} + \text{ModMul}))\), and then aggregates the decryption result. Next, it encrypts the addition result using the Paillier public key \(ppk_j\) \((2 * \text{ModExp})\), and then encrypts \(psk_j\) with CP-ABE using \(AP_{S}\) \(((|\gamma| + 1) * \text{Exp})\). Hence, the total computational cost of \(CSP_B\) in multiple users data processing case is: \((2 * N + 2) * \text{ModExp} + N * \text{ModMul} + (|\gamma| + 1) * \text{Exp}\).

Therefore, in total, computational complexity of both \(CSPs\) in multiple users data processing case is: \((4 * N + 5) * \text{ModExp} + (2 * N + 1) * \text{ModMul} + (|\gamma| + 1) * \text{Exp}\).

For the multiple users data processing case, \(DR\) decrypts a ABE ciphertext using his/her \(sk\) to obtain the Paillier decryption key \(psk_j\) (at most \(|\gamma|\)\(\text{BiPair}\)), and then uses it to decrypt the encrypted processing result \((2 * \text{ModExp} + \text{ModMul})\). In total, this gives a computational cost at \(DR\): \((2 * \text{ModExp} + \text{ModMul} + |\gamma| * \text{BiPair})\).

We compare the total computational costs of each entity in SAMA with the addition scheme of [30] in Table V.

**2) Computational Complexity of Access Control:** We assume that there are \(|U|\) universal attributes, in which \(|\gamma|\) attributes are in the access policy tree \(\tau\), and at most \(|\theta|\) attributes should be satisfied in the access policy tree \(\tau\) to decrypt the ciphertext. The \(\text{Setup}()\) will generate the public parameters using the given system parameters and attributes \(U\). This requires \(|U| + 1\) exponentiations and one bi-linear pairing. The \(\text{Enc}_{ABE}()\) requires two exponential operations for each leaf in the ciphertext’s access tree \(\tau\), which needs \(|\gamma| * 1 * \text{Exp},\) whereas the \(\text{KGen}_{ABE}()\) algorithm requires two exponential operations for every attribute given to the user. Also, the private key consists of two group elements for every attribute. Finally, \(\text{Dec}_{ABE}()\) requires two pairings for every leaf of the access tree \(\tau\) matched by a private key attribute and at most one exponentiation for each node along a path from that leaf to the root node.

The \(\text{Setup}()\) only need to be executed once. Thus, its computational complexity can be neglected in both single and multiple users data processing cases. Further, \(\text{Enc}_{ABE}()\) is performed only once to encrypt the private key of the encrypted final result in both single and multi-user scenarios, also its computational cost is negligible. Moreover, \(\text{Setup}()\) and \(\text{KGen}_{ABE}()\) are performed at \(KA\) and \(\text{Enc}_{ABE}()\) by \(CSP_B\), which means users will not be burdened with the computational cost. Although the \(\text{Dec}_{ABE}()\) algorithm is performed by \(DR\) which incurs some computational cost, it is an essential requirement to provide an authorised \(DR\) access to the final result with fine grained access control.

**B. Communication Overhead**

There are two types of communication overhead incurred in the SAMA scheme: overhead due to occasional data communication and overhead due to regular data communication. The former overhead captures the data sent occasionally, e.g., \(AP\) \((AP_{S}, AP_{M})\) uploads/updates and notifications. The latter overhead includes the regular data communication patterns within SAMA, such as data upload, data requests, and data exchanged between cloud providers when data is being processed. Since the former overhead is negligible compared to the latter overhead, here we focus only on the communication overhead due to regular data communication patterns.

To ease the analyses, we divide the communication overhead introduced by the SAMA scheme into three parts: overhead incurred (1) between users and \(CSPs\) denoted as (Users-to-\(CSPs\)), (2) between \(CSPs\) (Between-\(CSPs\)), and (3) between \(CSPs\) and \(DRs\) (\(CSPs\)-to-\(DRs\)).

1) **Users-to-\(CSPs\):** This is a common step for single and multiple users data cases. At each data reporting time slot, each user \(U_i\) sends one ciphertext to \(CSPA\). As each ciphertext has a length of \(2 * L(n)\) (operations are performed under \(mod\ n^2\)), the total communication overhead for this part in single and multiple users data processing is: \(N * 2 * L(n)\).

2) **Between-\(CSPs\):** The communication between \(CSPs\) in single-user data processing is as follows. \(CSPA\) sends one

| Entity | Computation of Single-user Data Processing | Computation of Multiple Users Data Processing | Computation of Addition in [30] |
|--------|------------------------------------------|-----------------------------------------------|-------------------------------|
| User   | \(2 * \text{ModExp}\) as this is a common step | \((4 * N + 5) * \text{ModExp} + (2 * N + 1) * \text{ModMul} + (|\gamma| + 1) * \text{Exp}\) | \(N_m * \text{ModMul} + 7 * \text{ModExp} + 2 * (|\gamma| + 1) * \text{Exp}\) |
| \(CSP\) | \((N_m + 2) * \text{ModMul} + 9 * \text{ModExp} + (|\gamma| + 1) * \text{Exp}\) | \(N_m * \text{ModMul} + 7 * \text{ModExp} + 2 * (|\gamma| + 1) * \text{Exp}\) | |
| \(DR\) | \(2 * \text{ModExp} + \text{ModMul} + |\theta| * \text{BiPair}\) | \(2 * \text{ModExp} + \text{ModMul} + |\theta| * \text{BiPair}\) | \(2 * \text{ModExp} + \text{ModMul} + |\theta| * \text{BiPair}\) |

**TABLE V: Computation Cost.**
ciphertext of length $2 \times L(n)$, which is the masked aggregated user’s data, to CSP$_B$. Then, CSP$_B$ sends one ciphertext of $2 \times L(n)$ to CSP$_A$, which is the masked encrypted processing result, and one CP-ABE ciphertext of $(|\gamma| + 1) \times \mathcal{L}$, where $\mathcal{L}$ is the bit length of elements in ABE. Therefore, the total communication among CSPs in the single-user data processing case is: $4 \times L(n) + (|\gamma| + 1) \times \mathcal{L}$.

The communication between CSPs in multiple users data processing is as follows. CSP$_A$ sends $N$ ciphertext (masked of encrypted user’s data) of length $2 \times L(n)$ to CSP$_B$, which is $N \times 2 \times L(n)$. Then, similar to the single-user data processing scenario, CSP$_B$ sends one ciphertext of $2 \times L(n)$ and $(|\gamma| + 1) \times \mathcal{L}$ of the CP-ABE ciphertext to CSP$_A$. The total communication cost among CSPs in multiple users data processing case is: $(N + 1) \times 2 \times L(n) + (|\gamma| + 1) \times \mathcal{L}$.

3) CSP$_A$-to-DRs: In the single and multiple users data, CSP$_A$ sends to DRs one ciphertext of length $2 \times L(n)$ (the encrypted processing result) and one CP-ABE ciphertext of length $(|\gamma| + 1) \times \mathcal{L}$. Thus, the communication between CSP$_A$ and the DRs is: $2 \times L(n) + (|\gamma| + 1) \times \mathcal{L}$.

A comparison between the communication overhead of the SAMA scheme and the addition scheme proposed in [30] is shown in Table VI. Overall, we can observe that the SAMA scheme has lower communication overhead than the Addition scheme in [30] at the user and DR side, while, the communication overhead between CSPs in multiple users case of the SAMA scheme is higher than [30].

| TABLE VI: Communication Overhead. |

| Communication of Single-user Data Processing | Communication of Multiple Users Data Processing | Communication of Addition in [30] |
|---------------------------------------------|-----------------------------------------------|---------------------------------|
| $N \times 2 \times L(n)$ as this is a common step | $(N + 1) \times 2 \times L(n) + (|\gamma| + 1) \times \mathcal{L}$ | $N \times 4 \times L(n)$ |
| CSP$_A$-to-CSP$_B$ | $2 \times L(n) + (|\gamma| + 1) \times \mathcal{L}$ | $N \times 2 \times L(n) + (|\gamma| + 1) \times \mathcal{L}$ |
| CSP$_A$-to-DR | $2 \times L(n) + (|\gamma| + 1) \times \mathcal{L}$ | $N \times 2 \times L(n) + (|\gamma| + 1) \times \mathcal{L}$ |

1) Computational Cost of Data Processing: We evaluate the computational cost for all of the four entities: $U$, CSP$_A$, CSP$_B$ and DR in both single and multiple users data processing scenarios and compare with the related work [30] (multi-user) in terms of different lengths of $n$. In addition, we show the computational cost of single and multiple users processing cases with a variable number of messages and users, respectively.

(a) Operation time of the user with the different lengths of $n$ (b) Operation time of CSP$_A$ with the different lengths of $n$

(c) Operation time of CSP$_B$ with the different lengths of $n$ (d) Operation time of DR with the different lengths of $n$

Fig. 6: Computational cost of the SAMA scheme with the different lengths of $n$.

C. Experimental Results

Here we present the experimental results of SAMA in three different settings: (1) computational cost of the data processing operations, (2) computational cost of the data access operations, and (3) communication overheads within SAMA.

For the computational cost, we have implemented the SAMA scheme to test its computational performances by conducting experiments with Java Pairing-Based Cryptography (jPBC) [40] and Java Realization for Ciphertext-Policy Attribute-Based Encryption (cpabe) [41] libraries on a laptop with Intel Core i7-7660U CPU 2.50GHz and 8GB RAM. We ran each experiment 500 times and took the average values. We set the length of $n$ to 1024 bits, $m$ to 250 bits, and $r$ to 500 bits. We show the computation evaluation for the single-user and multiple users data processing for all entities separately and specifically CSP$_A$ and CSP$_B$ as they perform different sets of computations in each case as described in Section VII-C1. In addition, the efficiency of user-centric access control and communication overhead among the entities are shown in Section VII-C2 and Section VII-C3 respectively.
We tested the performance of SAMAs with a variable number of users: We tested the performance of SAMAs with users and DRs which are resource constrained sides. Also, our scheme performs better computation efficiency compared to the scheme in [30], which supports only multi-user processing.

(ii) Performance of SAMAs single-user data processing with a variable number of messages: We tested the computation of SAMAs single-user data processing case by varying the number of data messages provided by a single-user as shown in Fig. 7. It can be seen from the figure, the operational time increases with the increase of the number of messages. However, only DR’s operation time is independent of the number of messages because it decrypts the processed result once, regardless of the number of messages that are processed at the CSPs.

(iii) Performance of SAMAs multiple-users data processing with a variable number of users: We tested the performance of SAMAs multiple users data processing by varying the number of users \((N_U = 10, 100, 1000, 10000)\) and fixing each user to generate only one message for data processing. As expected, the CSPs have more operation time compared to the user and DR. Moreover, if we compare Fig. 7 and Fig. 8, then the CSP operation time is higher in the multi-user case compared to the single-user data processing. Since VPE-HE supports only single-key homomorphic addition and does not support multi-key homomorphic addition, our multi-user processing computation time is higher than the single-user data processing at the CSP side. In other words, homomorphic data processing is executed over data encrypted only with the same encryption key. Therefore, as the multi-user data processing requires the decryption of all the messages using ssK, and then encrypting the aggregate with ppK, this incurs extra computation time.

2) Efficiency of User-Centric Access Control: We tested the computational efficiency of CP-ABE by varying the number of attributes from two to ten that are involved in the access policy as shown in Fig. 9. The Setup algorithm is relatively constant as it does not depend on the number of attributes. In addition, the decryption algorithm Dec_{ABE} in the test was set to require only one attribute needed to satisfy the access policy tree, therefore, the operation time of Dec_{ABE} is constant. The computational costs of Enc_{ABE} and KGen_{ABE} are linearly increasing with the increase in number of the attributes. Although employing CP-ABE achieves user-centric fine-grained access control, there is an additional computation overhead incurred. However, in our scheme, since the computation of Enc_{ABE} is outsourced to CSP, and Setup, KGen_{ABE} are
outsourced to \( KA \), this extra computation will not burden the resource-constrained devices (wearable) at the user side.

3) Communication Efficiency: The communication overhead among the entities is shown in Fig. 10 and it is evaluated by fixing the key size length \( n = 1024 \) bits and varying the number of messages to be computed. It is evident from Fig. 10a, the User-to-CSP \(_A\) communications at the SAMA scheme reduce the communication overhead by 50% compared to the scheme in [30]. Furthermore, it is essential to note that the scheme in [30] supports only multi-user processing by encrypting data with CSP\(_A\)’s public key. Therefore, in order to support single-user processing, they need to re-encrypt the same data again with the user’s public key as mentioned in [30]. Moreover, if we also compare the single processing communication overhead of our scheme with the scheme in [30] at the user side, our scheme reduces the communication overhead by 75%. At SAMA, a user has to encrypt wearable data only once for single and multi-user processing compared to the scheme in [30], which requires encrypting the user’s data twice to support both single and multiple data processing.

In addition, the scheme in [30] generates two ciphertexts for every data encryption, which increases communication overhead on the user side. While in the SAMA scheme only one ciphertext is generated. Clearly, we reduced the communication overhead significantly at the user side, which suits the resource-constrained devices. These results are consistent with the results obtained in [39], in which it compares the communication overhead of the two HE algorithms: BCP and VP-HE. They found that the communication cost of BCP is about twice that of VP-HE, which was used in [30].

Figure 10b depicts the communication overhead among the cloud servers: \((\text{CSP}_A\text{-to-CSP}_B\text{ and CSP}_B\text{-to-CSP}_A)\). Although our single-user processing achieves better communication efficiency compared to [30], the multi-user processing communication performance is significantly higher than the multi-user scheme of [30]. However, since CSP is not limited in resources, it can afford to support this higher communication overhead for multi-user processing. Moreover, the frequency of multiple-users data processing is relatively less than single-user data processing in most wearable and healthcare use cases that are more personalized. We achieve better communication efficiency with the most frequent single-user data processing. Therefore, our scheme is suitable mainly for the applications that require more frequent single-user than multi-user data processing such as wearables and outsourced personalized healthcare data processing.

The communication overhead between the CSP-to-DR is shown in Fig. 10c. As DR\(_s\) access only the processed result, there is less communication overhead between the CSP\(_s\) and DR. It is clear that our single and multi-user processing performs better than the scheme in [30] which supports only multi-user processing. Therefore, overall our scheme has significantly less total communication overhead compared to [30].

VIII. Conclusion

Wearable users (data owners) increasingly outsource their wearable data to third parties, e.g., cloud-based services, to benefit from personal and sociable analytic models results. As health data are sensitive, they should be protected against outsider as well as insider threats such as those from the data managers (cloud service providers). In addition, this protection should be applied to both raw data and aggregated data, and should allow data owners to manage the rights/privilege in accessing their data. This leads to the need for a solution with these properties: data are managed by a third party, and no trust is required on the third party, efficient single and multiple user(s) data processing, data owners can specify who can access which data items (i.e., user-centric access control), and fine-grained access to both raw data and computed/aggregated data (i.e., computed results).

To achieve these properties systematically, in this paper, we have designed and evaluated a novel flexible data processing and access control scheme, called SAMA, which supports aggregation over encrypted data of single and multiple users with user-centric access control and privacy preservation. SAMA combines the use of Paillier homomorphic encryption and CP-ABE with a user-centric approach. Security analysis through simulations shows that SAMA scheme fulfills the specified set of security and privacy requirements. Experimental results have also demonstrated its efficiency and advantages in terms of communication and computation in comparison with the existing related schemes.
As future work, we plan to extend SAMA to support more operations on encrypted data in order to facilitate more complex analytical processes.

REFERENCES

[1] V. G. Motti, “Wearable interaction,” in Wearable Interaction. Springer, 2020, pp. 81–107.
[2] S. Sharma, K. Chen, and A. Sheth, “Towards practical privacy-preserving analytics for iot and cloud-based healthcare systems,” IEEE Internet Computing, vol. 22, no. 2, pp. 42–51, 2018.
[3] M. M. Islam, S. Mahmud, L. Muhammad, M. R. Islam, S. Nooruddin, and S. I. Ayon, “Wearable technology to assist the patients infected with novel coronavirus (covid-19),” SN Computer Science, vol. 1, no. 6, pp. 1–9, 2020.
[4] A. Alavi, G. K. Bogu, M. Wang, E. S. Rangan, A. W. Brooks, Q. Wang, E. Higgs, A. Celii, T. Mishra, A. A. Metwally et al., “Real-time alerting system for covid-19 using wearable data,” medRxiv, 2021.
[5] S. Safavi and Z. Shukur, “Conceptual privacy framework for health information on wearable device,” PloS One, vol. 9, no. 12, pp. 1–16, 2014.
[6] A. Aloufi, P. Hu, Y. Song, and K. Lauter, “Computing blindfolded on data homomorphically encrypted under multiple keys: A survey,” ACM Computing Surveys (CSUR), vol. 54, no. 9, pp. 1–37, 2021.
[7] H. Shafagh, A. Hithnawi, L. Burkhalter, P. Fischli, and S. Duquennoy, “Secure sharing of partially homomorphic encrypted iot data,” in Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems, 2017, pp. 1–14.
[8] J. Zhou, Z. Cao, X. Dong, and X. Lin, “Security and privacy in cloud-assisted wireless wearable communications: challenges, solutions, and future directions,” IEEE Wireless Communications, vol. 22, no. 2, pp. 136–144, 2015.
[9] J. Liu, X. Huang, and J. K. Liu, “Secure sharing of personal health records in cloud computing: Ciphertext-policy attribute-based signcryption.” Future Generation Computer Systems, vol. 52, pp. 67–76, 2015.
[10] H. Shafagh, L. Burkhalter, S. Ratnasamy, and A. Hithnawi, “Droplet: Decentralized authorization and access control for encrypted data streams,” in 29th USENIX Security Symposium (USENIX Security 20), 2020, pp. 2469–2486.
[11] F. Wang, J. Mickens, N. Zeldovich, and V. Vaikuntanathan, “Sieve: Cryptographically enforced access control for user data in untrusted clouds,” in 13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16), 2016, pp. 611–626.
[12] M. Li, S. Yu, Y. Zheng, K. Ren, and W. Lou, “Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption,” IEEE Transactions on Parallel and Distributed Systems, vol. 24, no. 1, pp. 131–143, 2012.
[13] “Regulation 2016 - general data protection regulation,” (Accessed on 02/03/2022). [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32016R0679
[14] A. Act, “Health insurance portability and accountability act of 1996,” Public Law, vol. 104, p. 191, 1996.
[15] M. Blatt, A. Gusev, Y. Polyakov, and S. Goldwasser, “Secure large-scale genome-wide association studies using homomorphic encryption,” Proceedings of the National Academy of Sciences, vol. 117, no. 21, pp. 11608–11613, 2020.
[16] A. Ara, M. Al-Rodhaan, Y. Tian, and A. Al-Dhelaan, “A secure privacy-preserving data aggregation scheme based on bilinear elgamal cryptosystem for remote health monitoring systems,” IEEE Access, vol. 5, pp. 12601–12617, 2017.
[17] G. Wang, R. Lu, and Y. L. Guan, “Achieve privacy-preserving priority classification on patient health data in remote healthcare system,” IEEE Access, vol. 7, pp. 33 565–33 576, 2019.
[18] Z. Erkin, T. Veugen, T. Toft, and R. L. Lagendijk, “Generating private recommendations efficiently using homomorphic encryption and data packing,” IEEE Transactions on Information Forensics and Security, vol. 10, no. 3, pp. 1053–1066, 2015.
[19] J. Zhou, Z. Cao, X. Dong, and X. Lin, “PPDM: privacy-preserving protocol for dynamic medical text mining and image feature extraction from secure data aggregation in cloud-assisted e-healthcare systems,” IEEE Journal of Selected Topics in Signal Processing, vol. 9, no. 7, pp. 1332–1344, 2015.
[20] Y. Zhang, W. D. Xu, X. Jiang, H. Xiong, and S. Wang, “Foreseen: Fully outsourced secure genome study based on homomorphic encryption,” in BMC Medical Informatics and Decision Making, vol. 15, no. 5, Springer, 2015, pp. 1–11.
[21] X. Liu, K. K. R. Choo, R. H. Deng, R. Lu, and J. Weng, “Efficient and privacy-preserving outsourced calculation of rational numbers,” IEEE Transactions on Dependable and Secure Computing, vol. 15, no. 1, pp. 27–39, 2016.
[22] A. Peter, E. Tews, and S. Katzenbeisser, “Efficiently outsourcing multiparty computation under multiple keys,” IEEE Transactions on Information Forensics and Security, vol. 8, no. 12, pp. 2046–2058, 2013.
[23] J. Feng, L. T. Yang, Q. Zhu, and K. K. R. Choo, “Privacy-preserving tensor decomposition over encrypted data in a federated cloud environment,” IEEE Transactions on Dependable and Secure Computing, vol. 17, no. 4, pp. 857–868, 2018.
[24] L. Xiao, O. Bastani, and L.-L. Yen, “An efficient homomorphic encryption protocol for multi-user systems,” Cryptology ePrint Archive, 2012.
[25] M. Wang, J. Ren, K. Zhang, D. Zhang, Y. Zhang, and X. Shen, “Verifiable public key encryption with keyword search based on homomorphic encryption in multi-user setting,” IEEE Access, vol. 6, pp. 42 445–42 453, 2018.
[26] B. Wang, M. Li, S. S. Chow, and H. Li, “A tale of two clouds: Computing on data encrypted under multiple keys,” in 2014 IEEE Conference on Communications and Network Security, 2014, pp. 337–345.
[27] Z. Erkin, T. Veugen, T. Toft, and R. L. Lagendijk, “Generating private recommendations efficiently using homomorphic encryption and data packing,” in Proceedings of the 16th IEEE Symposium on Computers and Communications (ISCC), IEEE, 2011, pp. 850–855. [Online]. Available: http://gas.dia.unisa.it/projects/jpbc/
[28] J. Wang, “Java realization for ciphertext-policy attribute-based encryption,” in Proceedings of the 16th IEEE Symposium on Computers and Communications (ISCC), IEEE, 2011, pp. 850–855. [Online]. Available: https://github.com/junwei-wang/jpaebc/