Privacy-Preserving Service Composition with Enhanced Flexibility and Efficiency

Kevin Theuermann(✉), Felix Hoerandner, Andreas Abraham, and Dominik Ziegler

Institute of Applied Information Processing and Communications, Graz Technical University, 8010 Graz, Austria
{kevin.theuermann,felix.hoerandner,andreas.abraham}@egiz.gv.at
dominik.ziegler@tugraz.at

Abstract. Service compositions are implemented through the interplay between actors of different organizations. Many composition systems use a middleware, which coordinates the service calls according to specified workflows. These middlewares pose a certain privacy issue, since they may read all the exchanged data. Furthermore, service compositions may require that only selected subsets of data that was initially supplied by the user are disclosed to the receiving actors. Traditional public key encryption only allows encryption for a particular party and lack of the ability to efficiently define more expressive access controls for a one-to-many communication. Besides privacy-preserving requirements, it may be necessary for participants in service compositions to be able to verify which actor has modified or added data during a process to ensure accountability of performed actions. This paper introduces a concept for efficient, privacy-preserving service composition using attribute-based encryption in combination with outsourced decryption as well as collaborative key management. Our concept enables end-to-end confidentiality and integrity in a one-to-many communication using fine-grained access controls, while minimizing the decryption effort for devices with low calculation capacity, which enables to use smartphones at the client side. The feasibility of the proposed solution is demonstrated by an implemented proof-of-concept including a performance evaluation.

Keywords: Confidentiality · Integrity · Privacy · Efficiency

1 Introduction

Service composition is the combination of multiple atomic services to achieve a more complex service. These atomic services can be offered by the public or private sector. The advantages of service composition stem from the services' modularity: Atomic services that can be reused in various use cases, which reduces the complexity of building new process flows or adapting existing flows. Furthermore, some use cases dictate that multiple services need to be integrated, e.g., if the involved data is governed by different organizations. Our work focuses on

S. Gritzalis et al. (Eds.): TrustBus 2020, LNCS 12395, pp. 109–124, 2020.
https://doi.org/10.1007/978-3-030-58986-8_8

service orchestrations where a middleware coordinates requests to sub-services within the process flow. Forwarding the users' requests through a middleware raises privacy concerns, as this middleware is able to read the requests' data content or learns from it. Additionally, in many cases, not all actors of a composite service need access to the whole data-set of an initially transmitted request because they only require selected parts. Finally, end-to-end integrity generally represents a challenge in service compositions, as sub-services within the process flow or the middleware in service orchestrations itself could alter message parts. A fully trusted middleware may evaluate privacy policies and only selectively disclose necessary attributes to the different services (c.f. Ma et al. [1]). This enables fine-grained access controls by the middleware and thus enhances the users' privacy with respect to information exposed to the sub-services. However, this approach requires full trust in a central party (i.e., the middleware), which still learns the users' (sensitive) data. Alternative work aims to reduce risks of a curious middleware by providing end-to-end security through applying public-key cryptography (c.f. Singaravelu and Pu [2]). In this approach, a sender encrypts message fields for each service. Disadvantages of this concept are the required knowledge about recipients and their public keys, poor efficiency, a high number of needed encryption keys and message overhead by having multiple encryptions of the same attribute for different services.

Contribution. In this paper, we **propose a service composition architecture** that tackles the before mentioned issues by leveraging the capabilities of Ciphertext-Policy Attribute-Based Encryption (CP-ABE) [3] in combination with collaborative key management as well as outsourced decryption [4]. Our main contributions are:

- Our proposed architecture **preserves the users' privacy** against both, the different sub-services as well as the middleware, by applying advanced cryptographic mechanisms (c.f. CP-ABE) to our service composition protocol. This end-to-end encryption further **reduces trust requirements** into the middleware, as it only handles ciphertext.
- Additionally, our proposal **enhances flexibility** for service composition. With attribute-based encryption, the sender does not need to know the identity (e.g., public key) of all receiving participants of the service composition. Instead, the sender encrypts message parts for attributes that can be derived from the message's content (e.g. sensitivity), while the middleware can combine and replace sub-services holding sufficient decryption rights to fulfill the use case.
- The applied cryptography enables **efficient one-to-many communication**, as the decryption policy of one message part may be satisfied by multiple receivers. In contrast to other approaches, our architecture is efficient in the sense of computational effort (e.g., outsourced decryption) as well as message size overhead (e.g., no duplicate encryption of the same message part for different receivers).
- This work further supports **end-to-end integrity** and accountability by integrating digital signatures into the service composition protocol.

– To **evaluate** our proposed concept, we **present a case study** that describes a possible use-case within an e-Health context. In this case-study, we define the actors and show in detail the interactions as well as the data flows including cryptographic computations. In addition, we have **implemented and benchmarked a proof-of-concept**, which not only demonstrates the feasibility of our concept but also highlights its efficiency in real-world scenarios.

The remainder of this paper is structured as follows. Sections 2 and 3 provide information about the related work as well as the cryptographic background applied in our proposed concept. Section 4 describes the system model including the involved actors and our defined security objectives. The novel privacy-preserving service composition protocol is introduced by Sect. 5. Finally, Sect. 6 provides an evaluation of an implemented proof-of-concept, followed by a conclusion in Sect. 7.

2 Related Work

Privacy in Service Compositions. Recent work has focused on solving privacy concerns towards the different services that receive and process data. The work of Ma et al. [1] tries to solve privacy concerns by introducing a Privacy-Proxy Service (PPS), which is a fully trusted middleware that receives all the user's data and only discloses the needed data to the services. Even though this work solves privacy concerns regarding data receiving services there are still privacy and trust concerns towards the middleware. In contrast, the work of Khabou et al. [5,6] addresses privacy concerns in service compositions by proposing a framework that defines privacy models and policies. This work introduces services that agree on a privacy model and policy between a client and data processing services via a middleware. Therefore, these concepts raise privacy concerns regarding the middleware. The work of Carminati et al. [7] solves this privacy issue of an untrustred broker in service compositions. However, this concepts uses public keys of actors for encryption, which requires a sender to know the recipients. We want to guarantee encryption, where senders do not need to know anything about recipients at message generation time, to enable a highly flexible composition in the background. Our proposed concept tackles privacy concerns towards a centralized trusted party or a set of trusted parties as well as towards data processing services by utilizing CP-ABE.

Policy Language and Systems. In the past, the OASIS eXtensible Access Control Markup Language (XACML) [8] was specified aiming to increase privacy. The work of Kaehmer et al. [9] defines an Extended Privacy Definition Tool (ExPDT) trying to solve privacy concerns by using a policy language. Several issues emerge with the use of policies to achieve stronger privacy such as the policy enforcement, required knowledge of recipients as well as the inflexibility with respect to changes of the service composition flow by adding, removing or replacing services. Our proposed system tackles these issues by utilizing a novel combination of CP-ABE outsourced decryption and a collaborative key

management system. With the utilization of these cryptographic primitives, our system reflects a privacy-preserving system, which simultaneously provides high flexibility in terms of defining the process flows and involved services.

End-to-End Integrity. The work of Singaravelu et al. [2] focuses on end-to-end integrity within service compositions by applying public-key cryptography. In this concept, different message parts are encrypted for various service providers intended to decrypt them. This approach requires the client to know the included service providers and their corresponding public keys as well as their required data parts. This way, data that is transmitted to multiple receivers are contained in a service request redundantly, since the same data are encrypted for different service providers. Consequently, this approach suffers from issues regarding the required knowledge of involved service providers as well as an increasing message size overhead. We tackle these issues by using Ciphertext-Policy Attribute-Based Encryption in our concept, which does not require the client to have knowledge about involved service providers and their public keys and does not transfer data redundantly.

Cryptographic Primitives. Bethencourt et al. [3] first introduced Ciphertext-Policy Attribute-Based Encryption (CP-ABE), which enables the definition of access control policies through a set of attributes. While this cryptographic primitive enables a client to define complex access rules, it still raises concerns especially regarding the key revocation, key escrow and poor performance, because it is based on expensive bilinear pairings. This inefficiency becomes problematic, especially for client devices with limited computational power such as smartphones. To solve the performance problem in CP-ABE, concepts integrating outsourced decryption [10–12] were proposed. Lin et al. [4] proposed a solution for many of the issues related to CP-ABE by introducing a collaborative key management protocol. This protocol tackles the key escrow and key exposure problem and additionally reduces the decryption overhead of clients significantly through the application of outsourced decryption.

3 Background

Our concept for privacy-preserved service composition relies on cryptographic mechanisms. This section provides a description of applied technologies to understand the proposed solution.

3.1 Attribute-Based Encryption

Sahai and Waters [13] first introduced Attribute-Based Encryption (ABE), which represents an encryption mechanism that goes beyond traditional one-to-one public key encryption types. Instead of using the public key of a party to encrypt a message, a sender only has to know the attributes of the receiver in an ABE system. This way, the sender can encrypt a message, which is only decryptable

by receivers owning the specified attributes. The main advantage over conventional one-to-one encryption is that ABE enables one-to-many encryption. There are two types of ABE: Firstly, in Key-Policy Attribute-Based Encryption (KP-ABE) [14], the trusted authority attaches policies (i.e., access structures) to the issued keys, while the senders specify a set of attributes when encrypting the message. A ciphertext can only be decrypted if the key's access structure is satisfied by the ciphertext's attributes. Secondly, in Ciphertext-Policy Attribute-Based Encryption (CP-ABE) [3], the user defines the access policy and attaches it to the ciphertext during the encryption process. The trusted authority assigns a set of attributes when issuing the key material to users. As we want the client to be able to determine access policies, we use CP-ABE for our concept and give background information below. When encrypting a message, the user specifies an associated access structure over attributes of a given attribute universe U. These access structures are defined by attributes using conjunctions, disjunctions and *(k, n)* threshold gates, which means that k out of n attributes must be present. For example, let us assume an attribute universe $U = \{A, B, C, D, E\}$, a private key PK^1 to the attributes $\{A, B, C\}$ and a second private key PK^2 to the attributes $\{D, E\}$. A ciphertext that was encrypted with the policy $(A \wedge B) \vee (A \wedge E)$ would only enable the PK^1 to decrypt it.

Definition 1 (CP-ABE). We recall the definition of CP-ABE as suggested by Rouselakis and Waters [15]. CP-ABE consists of several algorithms defined as follows:

$\mathsf{Setup}(1^\kappa) \rightarrow (\mathsf{params}, \mathsf{MSK})$**:** The setup algorithm only takes the security parameter $\kappa \in \mathbb{N}$ as input[1]. The output of this function is represented by the public parameters params containing a description about the attribute universe U and a master secret key MSK.

$\mathsf{KeyGen}(\mathsf{MSK}, S) \rightarrow \mathsf{PK}$**:** The key generation algorithm takes the master secret key MSK as well as an attribute set S as input and outputs a private key PK corresponding to the given attribute set S. The generated private key can be used to decrypt a ciphertext, if its corresponding attribute set is an authorized set of S.

$\mathsf{Encryption}(\lambda, \mathsf{params}, m, \mathbb{A}) \rightarrow \mathsf{CT}$**:** The algorithm to encrypt a message requires as input the security parameter λ, the public parameters params, a message m, and a corresponding access structure $\mathbb{A}$ on the attribute universe U and generates a ciphertext CT.

$\mathsf{Decryption}(\mathsf{PK}, \mathsf{CT}) \rightarrow m$**:** The decryption algorithm takes the private key PK and the ciphertext CT. Finally, it outputs the message m in plain text.

A CP-ABE scheme is correct if a decryption private key on the attribute set S correctly decrypts a given ciphertext of an access structure $\mathbb{A}$ according to the decryption algorithm, when the attribute set S is an authorized set of access structure $\mathbb{A}$.

[1] Algorithms of CP-ABE take as input the security parameter κ (unary representation), which represents an input size of the computational problem indicating the complexity of cryptographic algorithms.

3.2 Outsourced Decryption and Collaborative Key Management

A collaborative key management protocol for Ciphertext Policy Attribute-Based Encryption (CKM-CP-ABE) has been introduced by Lin et al. [4] to extend upon CP-ABE. In this model, an actor's private key is split into three key parts: 1) for the key authority (KA), 2) for a re-encryption server (RS), and 3) for the client (CL). Using these key parts, the key authority and re-encryption server can reencrypt and partially decrypt a ciphertext for the client, thereby performing the most expensive calculations. Finally, the client applies her private key part to decrypt the prepared ciphertext, which usually requires less computational effort. Their model also allows for updates of the key material by using attribute group keys, which represent a collection of actors sharing the same attribute. If an attribute has been revoked from an actor, this actor is removed from the respective attribute group. The reencryption algorithm in this concept takes all attribute group keys and a ciphertext as input and updates decryption keys in case of a revocation. Below, we recall the cryptographic definitions from Ziegler et al. [16], which we slightly adapted for readability. We omit the public parameters $\mathsf{params} \leftarrow \{\mathsf{params}_{\mathsf{base}}, \mathsf{params}_{\mathsf{KA}}, \mathsf{params}_{\mathsf{RS}}\}$ as input for the algorithms.

Definition 2 (CKM-CP-ABE). Collaborative key management protocol in CP-ABE (CKM-CP-ABE) consists of the following algorithms:

$\mathsf{BaseSetup}(1^\kappa, S) \rightarrow \mathsf{params}_{\mathsf{base}}$**:** This algorithm takes a security parameter κ and a an attribute set S as input and outputs the base system parameters $\mathsf{params}_{\mathsf{base}}$.

$\mathsf{KASetup}(\mathsf{params}_{\mathsf{base}}) \rightarrow \mathsf{params}_{\mathsf{KA}}$**:** This algorithm takes the base system parameters and outputs the key authority parameters $\mathsf{params}_{\mathsf{KA}}$.

$\mathsf{RESetup}(\mathsf{params}_{\mathsf{base}}) \rightarrow \mathsf{params}_{\mathsf{RS}}$**:** This algorithm takes the base system parameters as input and outputs the reencryption parameters $\mathsf{params}_{\mathsf{RS}}$.

$\mathsf{KeyCreation}(S) \rightarrow \mathsf{IK}^{\mathsf{KA}}$**:** This algorithm takes a client's attribute set S as input and generates the initial key $\mathsf{IK}^{\mathsf{KA}}$.

$\mathsf{KeyUpdate}(\mathsf{IK}^{\mathsf{KA}}) \rightarrow (\mathsf{PK}^{\mathsf{KA}}, \mathsf{PK}^{\mathsf{RS}}, \mathsf{PK}^{\mathsf{CL}})$**:** Given the initial key $\mathsf{IK}^{\mathsf{KA}}$ of a user, this algorithm outputs an updated version of a user's three private key parts for the corresponding actors, $(\mathsf{PK}^{\mathsf{KA}}, \mathsf{PK}^{\mathsf{RS}}, \mathsf{PK}^{\mathsf{CL}})$.

$\mathsf{Encryption}(\mathbb{A}, m) \rightarrow \mathsf{CT}^{\mathsf{init}}$**:** To encrypt a message, this algorithm takes an access structure $\mathbb{A}$ as well as a message m as input. It outputs an initial ciphertext $\mathsf{CT}^{\mathsf{init}}$.

$\mathsf{ReEncryption}(\mathsf{CT}^{\mathsf{init}}, [\mathrm{Key}_{\mathrm{AG}}]) \rightarrow \mathsf{CT}^{\mathsf{re}}$**:** This algorithm, given an initial ciphertext $\mathsf{CT}^{\mathsf{init}}$ and a collection of attribute group keys $[\mathrm{Key}_{\mathrm{AG}}]$, generates a reencrypted ciphertext $\mathsf{CT}^{\mathsf{re}}$.

$\mathsf{PartialDecryption}(\mathsf{CT}^{\mathsf{re}}, \mathsf{PK}^{\mathsf{KA}}, \mathsf{PK}^{\mathsf{RS}}) \rightarrow \mathsf{CT}^{\mathsf{ulti}}$**:** Given a reencrypted ciphertext $\mathsf{CT}^{\mathsf{re}}$ and the private key parts of the key authority $\mathsf{PK}^{\mathsf{KA}}$ and the reencryption server $\mathsf{PK}^{\mathsf{RS}}$, this algorithm partially decrypts $\mathsf{CT}^{\mathsf{re}}$ and creates a final ciphertext $\mathsf{CT}^{\mathsf{ulti}}$.

$\mathsf{Decryption}(\mathsf{CT}^{\mathsf{ulti}}, \mathsf{PK}^{\mathsf{CL}}) \rightarrow m$**:** The decryption algorithm takes the final ciphertext $\mathsf{CT}^{\mathsf{ulti}}$ and the private key part of the client $\mathsf{PK}^{\mathsf{CL}}$ as input. This algorithm outputs the plaintext message m.

4 System Model

This section describes the involved actors including their related trust assumptions as well as the data flow of a composite service in our service composition architecture. Based on this trust model, we derive several security objectives, which serve as motivation for our proposed composition architecture and protocol.

4.1 Actors and Data Flow

The system architecture for our service composition system consists of the actors explained in the following paragraphs.

Client. The client represents a device used by a user, who wants to consume a composite service. The client does not need to know anything about actors participating in a composition to define a confidentiality policy. Hence, it is not required to know recipients at message generation time, which enables a highly flexible service composition in the background. Since smartphones must considered at client side, also devices with limited computational power have to be considered. One-time transmissions of data-sets reduce the communication steps for clients in service compositions. Therefore, clients must be able to protect the privacy of their data in a one-to-many communication.

Middleware. The middleware consists of several engines providing the functionality needed to coordinate the service calls and to handle the cryptographic mechanisms (e.g., reencryption and partial decryption of ciphertexts). As the coordination of service requests is performed by the middleware, it acts as a service registry and defines the workflows of composite services. Therefore, the middleware also provides the functionality to register services of service providers in the composition network by collaborating with the key authority. To protect the privacy of the users' data, it must be impossible for the included engines to read any data contents, which are exchanged in a composition. The middleware is semi-trusted and only receives encrypted messages.

Service Provider. Service providers are actors providing one or more services that can be used for service compositions. Each service provider has to perform an initial authentication against the key authority in order to participate in the service composition network. In the course of this authentication, a service provider can apply for a set of desired attributes. On a successful authentication and validation of authorizations, these attributes are assigned to the respective applicant. To guarantee confidentiality in our system model, service providers must not be able to read data contents, which are not intended for them. Additionally, since an initial data set may be altered by any service provider in a process sequence, every actor in a composition must be able to verify which party has modified data. This is necessary to be able to attribute responsibilities to actors regarding the output of a service.

Key Authority. The key authority (KA) is a semi-trusted component and responsible for the creation and update of decryption keys together with the reencryption server and a client. Furthermore, it is also responsible for the registration of new actors who want to participate in the service composition network. However, the type of authentication itself is not in the focus of this work. During registration, every applicant has to generate an asymmetric key-pair needed for signing. After successful registration, the KA issues a certificate for the public key of an applying actor, which is based on a PKI. The KA is authorized to grant or revoke attributes and is allowed to add attributes to the existing attribute universe. Furthermore, the KA must agree on updating a decryption key of any actor and is primarily responsible for key revocation management. In most of the concepts for ABE [3,4,17], the KA requires full trust, as it is able to generate any decryption key on its own and is, therefore, able to decrypt every ciphertext. To ensure the privacy of exchanged data in our system model, the KA must not be able to read any data contents. Thus, the KA must not be able to generate decryption keys on its own. For this purpose, we apply collaborative key management, where the KA cannot create decryption keys alone and is only able to create a decryption key together with a decryption server and a client. This way, full privacy can be ensured against the KA as well, even if this component is compromized by any adversary, which decreases the required trust. Even if the decryption server and the KA collude, they are not able to fully decrypt a ciphertext, as the client must also be included. When creating a decryption key, neither the KA nor the decryption server gets access to the final decryption key, they only contribute to its generation.

4.2 Security Objectives

This section summarizes our security objectives in the following paragraphs based on our system model and the actors' trust assumptions.

Objective 1: End-to-End Confidentiality. Intermediate actors such as the middleware or the key authority in our introduced system model are not sufficiently trusted to read the users' data. Therefore, the first security objective is end-to-end confidentiality between communicating actors.

Objective 2: End-to-End Integrity. In service compositions, the output of one service may serve as input for another service in a sequence. Hence, an (intentionally) incorrect result of a service may affect subsequent service and may cause a critical impact on a user (e.g., prescribed wrong medication). Attackers could try to change the output of a previous service to manipulate the final outcome of a composite service. For this purpose, every actor must be able to verify which data has been modified or added by which actor during a process to ensure end-to-end integrity. Also, this makes it possible to clarify accountabilities.

Objective 3: Fine-Grained Access Control. The actors (e.g., service providers) have different information requirements and are also only trusted to learn different subsets of the data. Therefore, every actor in our system model

must only be able to read data contents, which are intended for them. Our system model empowers clients to determine authorizations of actors to read desired message parts. This can be achieved by separating a message into several parts and encrypting them into fine-grained access rules. As clients in our system model do not necessarily need to know the actors of a composite service, it must be possible to encrypt message parts without knowing the specific recipients at message generation time.

Objective 4: Secure One-to-Many Communication. Parts of the same message might be required by multiple service providers. By applying one-to-one encryption to achieve confidentiality, such message parts need to be encrypted multiple times, for different recipients, which leads to a transmission overhead. Thus, our fourth security objective is encryption for one-to-many-communication to reduce the communication overhead by only transmitting an encrypted dataset once, where message parts are protected by individual access controls.

5 Composition Architecture and Protocol

This section describes architectural components and introduces our protocol for service compositions, which demonstrates the interplay between actors and achieves our defined security objectives.

5.1 Components

The middleware consists of the following four components:

Orchestration Engine. The orchestration engine (OE) coordinates the sequence of service requests. Information about available services is collected in a service registry. The OE determines the workflow of a composite service depending on the desired use case. Since this middleware receives all data transmitted by actors of a composite service, these data must be end-to-end encrypted.

Reencryption Server Engine. The reencryption server engine (RSE) reencrypts the ciphertext using the reencryption algorithm introduced in Sect. 3. Besides, the RSE is also needed for the generation of the system parameters and transmits its part of the user's private key to the decryption server engine to enable partial decryption. Since the RSE only receives and outputs ciphertext without learning the plain text intermediately, the privacy of sensitive data is guaranteed.

Decryption Server Engine. The decryption server engine (DSE) provides the functionality to partially decrypt ciphertexts. For the partial decryption, the DSE requires two parts of the user's private key: one part of the key authority and another of the reencryption server engine. It performs the computationally expensive operations of the decryption algorithm and outputs a final ciphertext. This final ciphertext requires less computational effort at client-side. The DSE only receives encrypted information and is not able to fully decrypt a ciphertext.

Service Registration Engine. This Service Registration Engine (SRE) is necessary to provide a registration mechanism for service providers applying to participate in the service composition network. The key authority informs the SRE after successful registration of a service provider about the approved attributes and transmits the issued certificate for the signing key material. The SRE validates the certificate and adds the respective services to the service registry.

5.2 Protocol for Privacy-Preserving Service Composition

This section describes a novel privacy-preserving service composition protocol offering end-to-end confidentiality and integrity. In this protocol, a user initially sends a request to the orchestration engine that forwards this request in sequence to multiple service providers $(SP_1, ..., SP_n)$, which operate on the received data and modify the data for the next actor. This process continues until the service composition is finally completed, and (optionally) a response for the user is generated. Below, we describe the protocol's steps as an iteration from a sender i to a receiver $i+1$, which becomes the next sender and starts the iteration again.

① The sender i splits the message m_i into multiple message parts $m_{i,j}$, if different requirements regarding the confidentiality for distinct message fields are necessary. The sender encrypts the message parts $m_{i,j}$ for an appropriate attribute set $\mathbb{A}_{i,j}$ which produces a ciphertext $\mathsf{CT}^{\mathsf{init}}_{i,j} \leftarrow \{\mathsf{Encryption}(\mathbb{A}_{i,j}, m_{i,j}) : \forall j\}$. Next, a session key $SymKey$ is generated for the creation of a HMAC for each message part $HMAC(m_{i,j}, SymKey)$, which are added to the message parts and signed with the private signature key SK_i of the pre-registered key-pair of the sender $\sigma_{i,j} \leftarrow \mathsf{Sign}(\mathsf{SK}_i, m_{i,j}, HMAC_{i,j})\forall j$. The HMAC avoids that the signature of the individual message parts gives any oracle about the plain text, which is especially important if data inputs can only embrace limited values. The sender also encrypts the session key $\mathsf{CT}_{\mathrm{SymKey}} \leftarrow \{\mathsf{Encryption}(\mathbb{A}, SymKey)\}$. Also, another signature is generated for the whole initial data set $\sigma_{initData} \leftarrow \mathsf{Sign}(\mathsf{SK}_i, m_i)$. $\mathsf{CT}^{\mathsf{init}}_i$, $\mathsf{CT}_{\mathrm{SymKey}}$ and the signatures $\sigma_{initData}$, $\sigma_{i,j}$ are transmitted to the orchestration engine.

② The orchestration engine receives the service request including all parameters and forwards $\mathsf{CT}^{\mathsf{init}}_{i,j}$ together with information about the next actor $i+1$ to the reencryption server engine to perform a reencryption of the ciphertext.

③ The reencryption server engine receives the initial ciphertexts $\mathsf{CT}^{\mathsf{init}}_{i,j}$ and performs the reencryption algorithm. With multiple message parts $m_{i,j}$, this reencryption process is performed for each individual part, resulting in the reencrypted ciphertexts $\mathsf{CT}^{\mathsf{re}}_{i,j} \leftarrow \{\mathsf{ReEncryption}(\mathsf{CT}^{\mathsf{init}}_{i,j})\forall j\}$. Next, the reencryption server engine verifies, if the next actor of the composite service is still authorized to possess her attribute set, or if any attribute has been revoked for this actor. In case of revocation, the reencryption server engine updates the corresponding set of attribute group G keys. For reencryption, we used the attribute group-based algorithm of Hur [18]. Thereafter, the reencryption

server engine returns the reencrypted ciphertext $\mathsf{CT}_i^{\mathsf{re}}$ to the orchestration engine.

④ The orchestration engine receives the reencrypted ciphertext $\mathsf{CT}_i^{\mathsf{re}}$ and forwards it to the decryption server engine.

⑤ The decryption server engine receives the decryption query with $\mathsf{CT}_i^{\mathsf{re}}$ from the orchestration engine. First, it retrieves two of three private key parts of the receiver, i.e., $\mathsf{PK}_{i+1}^{\mathsf{KA}}$ and $\mathsf{PK}_{i+1}^{\mathsf{RS}}$ from the key authority and the reencryption server engine, respectively. With those key parts, the decryption server engine partially decrypts the individual ciphertext parts $\mathsf{CT}_{i,j}^{\mathsf{ulti}} \leftarrow \mathsf{PartialDecryption}(\mathsf{CT}_{i,j}^{\mathsf{re}}, \mathsf{PK}_{i+1}^{\mathsf{KA}}, \mathsf{PK}_{i+1}^{\mathsf{RS}})$. The reencryption server engine outputs the final ciphertext $\mathsf{CT}_i^{\mathsf{ulti}}$, which is returned to the orchestration engine.

⑥ The orchestration engine forwards the final ciphertext $\mathsf{CT}_i^{\mathsf{ulti}}$ together with all the received signature values to the next actor according to the process sequence.

⑦ The next actor receives the partially decrypted ciphertext $\mathsf{CT}_i^{\mathsf{ulti}}$ and applies her private key part $\mathsf{PK}_i^{\mathsf{CL}}$ to finally decrypt the message parts $m_{i,j} \leftarrow \mathsf{Decryption}(\mathsf{CT}_{i,j}^{\mathsf{ulti}}, \mathsf{PK}_i^{\mathsf{CL}})$. If the user has sufficient attributes, she is able to successfully perform this decryption. Next, she verifies the signature value of each message part $\mathsf{Verify}(\mathsf{VK}_i, \sigma_{i,j}, m_{i,j})$ as well as the signature of the whole message $\mathsf{Verify}(\mathsf{VK}_i, \sigma_i, m_i)$. This way, she can verify changes of message parts. Afterwards, she performs her business logic on the plaintext, giving m_{i+1}. This actor may now take on the role of the sender and repeat the process with the next actor, which is coordinated by the orchestration engine. If the actor changes or modifies any data of a message part $m_{i+1,j}$, she has to sign this message part with her pre-registered private signature key SK_{i+1}, giving $\sigma_{i+1,j} \leftarrow \mathsf{Sign}(\mathsf{SK}_{i+1}, m_{i+1,j})$. Furthermore, she also has to generate a signature for the whole data set to enable a verification against integrity for the following actor $\sigma_{i+1} \leftarrow \mathsf{Sign}(\mathsf{SK}_{i+1}, m_i)$.

6 Evaluation

This section illustrates a case study for a telemedical service, presents a proof-of-concept implementation, evaluates the performance, and discusses various aspects.

6.1 Case Study

To demonstrate the feasibility of our proposed concept, let us assume an example use-case representing a telemedical service that offers an alert system, as shown in Figure 1. This composite service consists of three web services:

Ⓐ The first service validates a heart rate against abnormalities and verifies motion-data to detect unusual movements. If the heart rate is abnormal and the movement is unusual (e.g. patient fell down), it adds a validation result indicating a case of emergency. This service is provided by a health service provider.

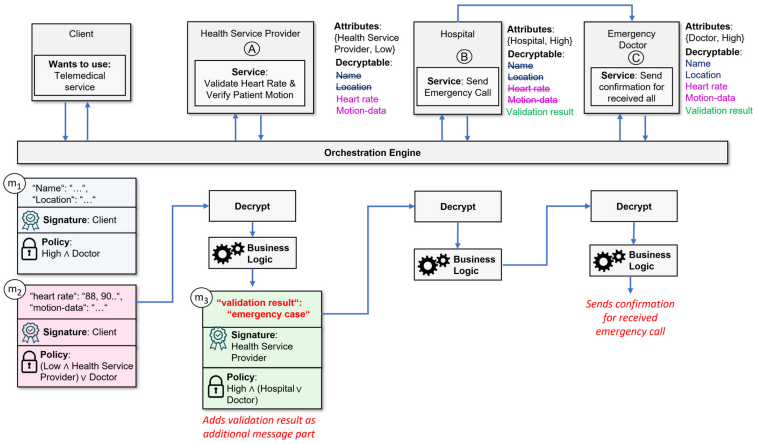

Fig. 1. Use-case of a medical drug prescription service

Ⓑ The second service verifies the validation result and transmits an emergency call including all information to any available doctor in case of emergency. This service is operated by a hospital.
Ⓒ The final service sends a confirmation to the client, when the emergency call has been received. This service is operated by an emergency doctor.

The attribute universe $\mathbb{U}$ for this scenario includes the following attributes: $\mathbb{U} = \{\text{Low, High, Health Service Provider, Hospital, Doctor}\}$, whereby the attributes 'Low' and 'High' indicate a security level. The initial message transmitted by the client consists of the name, location, heart rate and motion-data of the patient, which are transmitted to the client device by sensors. The data set is split into two message parts and encrypted to different confidentiality policies as given by Fig. 1. Service A extends the initial message by adding the validation result based on the heart rate and motion-data. This services has to ensure authenticity, integrity and confidentiality of the added message field. Thus, this service provider signs the respective data field and encrypts it. In this use-case, any service provider encrypts a received message part to the same access structure, which was required to decrypt it.

6.2 Implementation and Performance

We have implemented the introduced use-case with Java-based web services as well as an Android app. These actors are based on the collaborative key management protocol for CP-ABE introduced by Lin et al. [4]. We have implemented this cryptographic scheme by using the IAIK Java Cryptography Extension (IAIK-JCE)[2] as well as the IAIK ECCelerate elliptic curve library (See footnote 2),

[2] https://jce.iaik.tugraz.at.

which provides cryptographic functionality for bilinear pairings. For our evaluation, we choose a security parameter of $\lambda = 256$ bits, providing long-term security according to NIST's recommendation [20].

Table 1. Execution times in milliseconds (ms) ("–" denotes unnecessary operations)

Sender → Receiver	Setup	Key Creation	**Message Part**	Encryption	ReEncryption	Partial Decryption	Decryption
Client ↓ HSP	26	47	1	18	95	140	–
			2	24	101	102	15
HSP ↓ Hospital			1	–	–	–	14
			2	24	111	106	13
			3	19	104	118	–
Hospital ↓ Doctor			1	18	115	135	15
			2	26	109	116	13
			3	29	106	142	18

Message Format. To encode the messages, we have chosen JSON Web Tokens (JWT). The sender (i.e., user or service provider) first structures the data as JSON documents and then attaches a signature generated with its private signing key. If an actor modifies the payload of a JWT or adds a new message part, it has to sign the token again, which enables to trace who has edited data. Finally, all the JWTs are encrypted to a desired access structure $\mathbb{A}$. The use of JWT allows for lightweight communication, which offers suitable performance for our use case.

Test Setup. The collaborative key management protocol algorithms are performed by three web services running on a Lenovo Thinkpad T460s, which has an Intel i5-6200U dual-core processor and 12 GB RAM. Since we also want to consider smartphones at client-side, we use a HTC U11, which has a Qualcomm Snapdragon 835 quad-core processor with 2.45 GHz each and 4GB RAM.

Performance for Case Study. The smartphone starts the process by initially encrypting the two generated JWT with different symmetric 256-bit AES session-keys. Afterwards, these keys were encrypted to different confidentiality policies as defined in Sect. 6.1. One web service represents the orchestration middleware and takes over the service call coordination and performs the reencryption algorithm. The second web service performs the partial decryption algorithm, which is normally performed by a powerful decryption server. To provide reliable results, we have conducted the test 50 times and used the median of these performance values. Message parts that are not decrypted by an actor do not need to be encrypted, reencrypted or partially decrypted for the next actor in the sequence. These unnecessary operations are denoted as "–" in Table 1. The test results demonstrate that our choice of cryptographic mechanisms provides practically-efficient execution time for service compositions. Especially, the needed execution time for decryption at client-side demonstrates a perfectly suitable approach to be used on devices with limited computational power such

as smartphones. The execution time of the partial decryption can be further decreased by using of powerful hardware for the decryption server, rather than our Lenovo Thinkpad T460s.

General Performance of CKM-CP-ABE. According to Ambrosin et al. [19], a reasonable scenario in ABE comprises about 30 attributes, which can lead to highly complex policies. Ziegler et al. [16] performed a comprehensive evaluation of CP-ABE with outsourced decryption. They investigate the execution times for a varying number of attributes as well as the impact of different security levels with security parameters of $\lambda = 80$ bits to $\lambda = 512$ bits. Their results show, that applying CKM-CP-ABE to a service orchestration system enables practical efficiency.

6.3 Discussion

Fine-Grained Access Control. By applying ciphertext-policy attribute-based encryption to our concept, senders are to define fine-grained access controls to message parts. During encryption, the sender specifies a confidentiality policy based on attributes. Only receivers who hold attributes that satisfy this policy are able to decrypt the message parts.

Flexibility. The use of attribute-based encryption enables actors to encrypt data without requiring any specific knowledge about the intended recipients. This improves flexibility regarding the composition in the background, because the service registry can change workflows by simply replacing a service and does not need to inform actors in the service composition system about any alterations.

Reducing Trust Requirements. By employing a collaborative key management protocol, our proposed system avoids a common problem of ABE systems: Even the key authority in our concept is not able to create decryption keys on its own in order to fully decrypt a ciphertext, which ensures privacy against this party and represents a major advantage compared to other ABE systems.

Efficient One-to-Many Encryption. Our concept also supports efficient as well as secure one-to-many communication, as multiple services may hold sufficient attributes to decrypt the same message parts. Such one-to-many communication reduces the communication overhead in comparison to systems that have to encrypt the same message multiple times for different receivers.

Performance. The integration of outsourced decryption in combination with a collaborative key management protocol represents a suitable mechanism to ensure sufficient performance on smartphones. Although we did not perform the partial decryption on a powerful server, the algorithm' execution time was already satisfying and can be further reduced by using more powerful hardware.

7 Conclusion

In this paper, we have introduced an architecture and protocol for efficient, flexible, and privacy-preserving service composition. The proposed concept enables end-to-end encryption and fine-grained access controls in a one-to-many communication by combining ciphertext-policy attribute-based encryption with outsourced decryption and a collaborative key management protocol. Such attribute-based encryption also improves flexibility, as senders do not need to know the final recipients, which allows middlewares to change or replace the services within the process flow on demand. While secure one-to-many communication reduces the transmission overhead, little computational effort is required of clients (e.g., smartphones) as outsourced decryption enables to outsource large parts of the decryption efforts to powerful servers. Additionally, our protocol offers end-to-end integrity for exchanged messages, which allows to trace modifications to the message parts and therefore provides accountability. The feasibility of this concept is highlighted by a proof-of-concept implementation, which demonstrates practical efficiency of the used cryptographic mechanisms.

References

1. Ma, Z., Manglery, J., Wagner, C., Bleier, T.: Enhance data privacy in service compositions through a privacy proxy. In: Sixth International Conference on Availability, Reliability and Security, pp. 615–620 (2011). https://doi.org/10.1109/ARES.2011.94
2. Singaravelu, L., Pu, C.: Fine-grain, end-to-end security for web service compositions. In: IEEE International Conference on Services Computing (SCC 2007), pp. 212–219 (2007). https://doi.org/10.1109/SCC.2007.61
3. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: IEEE Symposium on Security and Privacy (SP 2007), pp. 321–334 (2007). https://doi.org/10.1109/SP.2007.11
4. Lin, G., Hong, H., Sun, Z.: A collaborative key management protocol in ciphertext policy attribute-based encryption for cloud data sharing. IEEE Access **5**, 9464–9475 (2017). https://doi.org/10.1109/ACCESS.2017.2707126
5. Khabou, I., Rouached, M., Abid, M., Bouaziz, R., Enhancing web services compositions with privacy capabilities. In: Proceedings of the 17th International Conference on Information Integration and Web-based Applications & Services, iiWAS 2015, Brussels, Belgium, pp. 57:1–57:9 (2015). https://doi.org/10.1145/2837185.2837240
6. Khabou, I., Rouached, M., Bouaziz, R., Abid, M.: Towards privacy-aware web services compositions. In: 2016 IEEE International Conference on Computer and Information Technology, CIT 2016, Nadi, Fiji, December 8–10, pp. 367–374 (2016). https://doi.org/10.1109/CIT.2016.26
7. Carminati, B., Ferrari, E., Tran, N.H.: Secure web service composition with untrusted broker. In: IEEE International Conference on Web Services, pp. 137–144 (2014)
8. Organization for the Advancement of Structured Information Standards (OASIS): Extensible Access Control Markup Language (XACML), Identity, v.1.1. (2006)

9. Kaehmer, M., Gilliot, M., Mueller, G.: Automating Privacy Compliance with ExPDT. In: 2008 10th IEEE Conference on E-Commerce Technology and the Fifth IEEE Conference on Enterprise Computing, ECommerce and E-Services, pp. 87–94 (2008). https://doi.org/10.1109/CECandEEE.2008.122
10. Zuo, C., Shao, J., Wei, G., Xie, M., Ji, M.: CCAsecure ABE with outsourced decryption for fog computing. Future Gener. Comput. Syst. **78**, 730–738 (2016). https://doi.org/10.1016/j.future.2016.10.028
11. Wang, Z., Liu, W.: CP-ABE with outsourced decryption and directionally hidden policy. Secur. Commun. Netw. **9**(14), 2387–2396 (2016)
12. Green, M., Hohenberger, S., Waters, B.: Outsourcing the decryption of ABE ciphertexts, pp. 34–34 (2011)
13. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005). https://doi.org/10.1007/11426639_27
14. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: Proceedings of the ACM Conference on Computer and Communications Security, pp. 89–98 (2006). https://doi.org/10.1145/1180405.1180418
15. Rouselakis, Y., Waters, B.: New constructions and proof methods for large universe attribute-based encryption. In: Proceedings of the ACM Conference on Computer and Communications Security, pp. 463–474 (2013). https://doi.org/10.1145/2508859.2516672
16. Ziegler, D., Sabongui, J., Palfinger, G.: Fine-grained access control in industrial Internet of Things. In: Dhillon, G., Karlsson, F., Hedström, K., Zúquete, A. (eds.) SEC 2019. IAICT, vol. 562, pp. 91–104. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-22312-0_7
17. Oualha, N., Nguyen, K.T.: Lightweight attribute-based encryption for the Internet of Things. In: 2016 25th International Conference on Computer Communication and Networks (ICCCN), pp. 1–6 (2016). https://doi.org/10.1109/ICCCN.2016.7568538
18. Hur., J.: Improving security and efficiency in attribute-based data sharing. IEEE Trans. Knowl. Data Eng. **25**, 2271–2282 (2013). https://doi.org/10.1109/TKDE.2011.78
19. Ambrosinet, M., et al.: On the feasibility of attribute-based encryption on Internet of Things Devices. IEEE Micro **36**, 25–35 (2016). https://doi.org/10.1109/MM.2016.101
20. National Institute of Standards & Technology: Recommendation for Key Management, Part 1: General (Rev 4). SP 800-57 (2016)