Compliance Checking of Cloud Providers: Design and Implementation

MASOUD BARATI, Carleton University, Canada
KWABENA ADU-DUODU, Newcastle University, UK
OMER RANA, Cardiff University, UK
GAGANGEET SINGH AUJLA, Durham University, UK
RAJIV RANJAN, Newcastle University, UK

The recognition of capabilities supplied by cloud systems is presently growing up. Collecting or sharing healthcare data and sensitive information especially during Covid-19 pandemic has motivated organizations and enterprises to leverage the upsides coming from cloud-based applications. However, the privacy of electronic data in such applications remains a significant challenge for cloud vendors to adapt their solutions with existing privacy legislation standards such as general data protection regulation (GDPR). This paper, first, proposes a formal model and verification for data usage requests of providers in a cloud composite service using a model checking tool. A cloud pharmacy scenario is presented to illustrate the connectivity of providers in the composite service and the stream of their requests for both collection and movement of patient data. A set of verification is, then, undertaken over the pharmacy service in accordance with three significant GDPR obligations, namely user consent, data access and data transfer. Following that, the paper designs and implements a cloud container virtualization based on the verified formal model realising GDPR requirements. The container makes use of some enforcement smart contracts to only proceed the providers' requests, which are compliant with GDPR. Finally, several experiments are provided to investigate the performance of our approach in terms of time, memory and cost.

CCS Concepts: • **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network rlatex downleliability.

Additional Key Words and Phrases: data privacy, cloud services, transition systems, blockchain, container, general data protection regulation

1 INTRODUCTION

Cloud users have the right to control their own information, whether private, professional, or public. The users often use cloud services without any knowledge of the cloud servers' physical location and the processing operations carried out on their personal data. In public clouds, where their infrastructures and solutions are provisioned for open use, the risk of disclosing confidential information through third party access to sensitive personal data is very high. Hence, data privacy concerns have always become a main challenge and a big threat for businesses in public cloud-based systems despite many laws have been published for protecting the sensitive information relevant to cloud users [1]. General data protection regulations (GDPR) is one of such laws which recently came into effect in order to clearly set out the ways

Authors' addresses: Masoud Barati, Masoud.Barati@carleton.ca, Carleton University, Canada; Kwabena Adu-Duodu, Newcastle University, UK, larst@affiliation.org; Omer Rana, Cardiff University, UK, ranaof@cardiff.ac.uk; GaganGeet Singh Aujla, Durham University, UK, gagangeet.s.aujla@durham.ac.uk; Rajiv Ranjan, Newcastle University, UK, rranjans@gmail.com.

2769-6472/2023/2-ART $15.00
https://doi.org/10.1145/3585538

under which the privacy rights of users are completely preserved. GDPR enforces the actors collecting and processing personal data to comply with its obligations and carries hefty penalties for non-compliance demeanor [4].

The integration of GDPR in cloud has led to assigning explicit responsibilities for cloud providers with the roles of data controllers and processors so as to securely keep sensitive information of cloud users (data subjects) and give them the right to access their under-processing data any time [6]. For example, under GDPR, a provider as a data processor is responsible for implementing safe guards or encryption mechanisms before maintaining personal data in its local cloud storage. In order to automatically verify the compliance of GDPR obligations over the activities of cloud providers on user data, Blockchain and smart contracts technologies have presently been used to implement GDPR rules as programming codes and flag data breaches in a secure, automated and transparent way [8]. Blockchain as a public decentralized ledger has also been exploited in cloud container-based virtualization to improve data accountability, which is a key GDPR principle, by recording any action (e.g., read, write etc.) taken by a provider on personal data [13].

Making use of GDPR and Blockchain in cloud has lately given rise to emerging several privacy-aware solutions enhancing customers' satisfaction and trust to use cloud services. An instances of such solutions is an information management framework that represents how Blockchain can realize GDPR requirements for state-of-the-art cloud-based solutions [14]. However, the framework did not examine the GDPR concerns that must be monitored for both data access and transfer requests. A cloud-based data management platform which is compliant with GDPR and uses blockchain technology was proposed in [15]. The platform enabled both cloud providers and their users to manipulate and process their data while data provenance mechanism is provided through Blockchains. Though, the verification of GDPR obligations was only limited to the user consent's requirement over the framework. In [16, 17], the combination of GDPR and Blockchain within the business process of cloud providers has been developed so as to enhance the data protection of cloud users. Nonetheless, preventative strategies for limiting the access of providers to personal information were not investigated in their proposed approach. In [18], the authors by using Blockchain presented a GDPR compliant decentralized and trusted data sharing and tracking model for cloud. The approach keeps sensitive data off-chain and submits non-sensitive identifiable information in Blockchain so as to track personal information movement among cloud providers. The approach, however, lacks an automatic verification technique for detecting GDPR violations in accordance with data transfer obligations. Moreover, the authors presented their model theoretically without any investigation on cloud testbeds or Blockchain test networks.

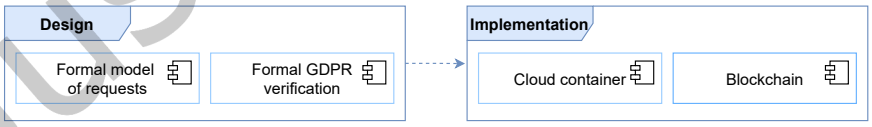

Fig. 1. Overview of our approach

In order to address the shortcomings highlighted in the aforementioned approaches, this paper presents a privacy-preserving solution for monitoring data usage requests issued by cloud providers and filtering/ rejecting those, which are not compliant with GDPR requirements. Figure 1 shows an overview of our solution that encompasses: (i) design an abstract model for verifying GDPR-compliance of the requests in a formal way, and (ii) technical implementation of the model in practice using container and Blockchain. The main contributions of the paper are summarised below:

- Defining a formal representation for data usage requests of providers on cloud users data in order to facilitate verification of envisaged requests at design time;
- Developing GDPR standard obligations, including user consent, data access and data transfer, as guards on the proposed data usage requests model for specifying compliance/ non-compliance patterns;
- Presenting a formal behavioral description for data usage requests using timed automata. It provides a foundation for checking the GDPR compliance of cloud requests with the aid of a model checking tool (e.g., Uppaal);
- Extending cloud-based containers' architecture by adding a monitoring layer that realises our presented abstract model and formal verification. The architecture leverages both GDPR standards and Blockchain technology for managing, filtering and logging cloud access and transfer requests;
- Implementing a preventative mechanism through smart contracts under which the requests of service providers that do not follow GDPR requirements are automatically flagged and prohibited to proceed;
- Proposing a classification for non-compliance requests using a postponed verification that is practically implemented by a smart contract. It enables data controllers and trusted legal parties to retrieve the violation level of data processors or their subcontractors and from a Blockchain network in a transparent way;
- Providing some experiments from our implementations that demonstrate the potential scalability of the proposed approach in terms of time, memory, used gas etc.

The remainder of the paper is structured as follows: Section 2 reviews some related work. Section 3 presents a scenario describing data usage requests within a cloud pharmacy composite service. Section 4 defines formalism and GDPR-compliance verification of cloud providers' requests using finite state machines. Section 5 designs a container-based architecture for implementing a control version mechanism for data collection and transfer requests. Section 6 provides experimental results of our monitoring approach in both design time and run time, and finally Section 7 describes conclusions and future works.

2 RELATED WORK

According to [5], cloud computing poses a strict challenge to create symmetry between data controller and processor in context of data protection regulations due to its complex service chain. There are various existing proposals that considered privacy and compliance (specifically GDPR) as a priority while providing cloud services. For example, a two phase framework supporting heterogeneous privacy for nodes' personal data was proposed in [20]. The nodes through one-shot noise perturbation reached heterogeneous privacy protection over various data servers. Moreover, providing that data servers have fixed budgets, an efficient incentive strategy was presented for optimising computation accuracy. However, the compliance of the framework with regards to existing data privacy regulations was not studied. A fairness and transparent framework named as ATMOSPHERE for trustworthy cloud ecosystem was proposed in [2]. This framework was built over Lemonade platform and computes a trust score based on measure and monitor policy. However, this work does not clarify how it considers trustworthiness across different cloud providers. Also, they do not discuss any specific GDPR regulations for establishing fairness and transparency. Another work [10] considers PKI-based trust model to incorporate authentication and authorization to ensure some of the GDPR requirements (privacy, confidentiality, integrity and non-repudiation) in fog-to-cloud systems. But, it do not consider a full fledged end to end solution concerning GDPR for cloud systems. In another work [3], the authors proposed a semantically rich Ontology to represent the GDPR mandates for the cloud providers. Although this work identify the different roles

and their corresponding obligations, but the practical implementation and technical consideration of this work are not validated.

The existing proposals also suggests that Blockchain technology provides a useful mechanism to verify if a particular security policy has been realised in practice. [21, 22]. It enables the production of an audit trail for cloud providers through a fully distributed, secure and consensus-based approach [23]. A framework for supporting cloud users in designing and deploying multi-cloud systems was presented in [25]. Although the framework made use of GDPR rules for ensuring data privacy of cloud users, it lacked a Blockchain-based technique to automatically verify such rules on processing activities carried out by providers on user data. However, an automatic mechanism for detection of the possible privacy or GDPR violations should be embedded in the cloud architecture by design. For instance, an automatic way for tracking and enforcing data sharing agreements between a customer and cloud providers using smart contracts and Blockchain was proposed in [26]. In this approach, the violation of the shared agreements by the providers were detected by a number of voters listed in a voting contract. Similarly, the integration of Blockchain and GDPR resulted in the design of a privacy-aware architecture for cloud ecosystems – promoting access control [7, 8]. For instance, MeDShare [24] system has been proposed to manage medical data sharing in an untrusted cloud environment. This system enables data provenance, auditing and control mechanisms using a Blockchain, to record all the actions that are performed over the data when it is transmitted from one entity to another. Here, smart contracts and access control mechanisms are applied to track data movement, identifying parties that violate data permissions, and revoking data access for such violators.

Some of these proposals suggested that container-based monitoring can provide a real-time mechanism to record and log the data events in cloud ecosystem inline with the data privacy regulations [13]. Container-based monitoring can also enforce controls and policies defined by organisations (such as GDPR) [27, 28]. An automatic scenario for privacy-aware software architecture was developed in [9] to study the impact of GDPR. This architecture is based on virtualization technologies alongside adoption of authorization and encryption mechanism. But, it does not provide any provision to automatically verify the compliance neither did it consider Blockchain for audit trail of the events. In [29], a number of GDPR rules were translated into smart contracts in order to automatically verify legal compliance for operations executed by providers on data of cloud customers. Although the foregoing approaches make use of GDPR and Blockchain for improving data privacy, none of them provides enforcement and preventative mechanisms for the data access and transfer requests of cloud providers. Moreover, they cannot be directly utilised alongside existing container technologies – e.g. Docker, Kubernetes. A public Blockchain-based architecture was used to provide an evidence of compliance to GDPR in [11]. This architecture utilizes off chain mechanism to handle the challenge of scalability and verifies the information external to the Blockchain. A case study for smart home was developed to the cross-boundary compliance. However, this architecture does not provide an automated mechanism for detection and prevention from GDPR violations. Moreover, it has not been validated against popular cloud virtualization architectures. It was suggested in [13], that a unified architecture that considers container-based monitoring to record and log it further on the Blockchain for verification can handle the challenges of the existing data privacy solutions in cloud systems. A cloud-based order system was formally verified via Uppaal in accordance with a GDPR obligations under which processing activities violating the obligations were detected at design time [16]. However, the implementation of such verification through practical technologies/ tools (e.g., Blockchain and container) was not discussed. An Ontology for GDPR in integration with Blockchain was developed in [12] to realize and automated and real-time compliance. This solution ensures that any operation is executed only if it complies with the GDPR (privacy rules). In case of a valid transaction, the personal data is stored in off chain storage in an automatic fashion. Also, an access control mechanism

is used to enforce data privacy when a multi-party data sharing is required. However, this framework was only validated over an Ethereum network, and it was not validated in a realistic cloud environment. Moreover, the implementation of the framework through practical cloud technologies (e.g., Container) are not discussed.

3 A CLOUD-BASED PHARMACY SCENARIO

Imagine that a cloud customer uses an online pharmacy to submit an order, make a payment and get medicines through a shipping service. Figure 2 represents the online pharmacy. As seen, there exists six cloud services within the composite service offered by the online pharmacy and each provider of such services requires some personal data for the aims of prescription's processes and medicine delivery.

The requested personal data items can be classified into customer's identification, bank account details, address and healthcare information (e.g., general practitioner diagnosis). The cloud-based pharmacy utilizes a Russia-based Infrastructure as a Service (IaaS) provider (Cloud4U) to manage its website and mobile application. The pharmacy's application involves "social plugins" from a social network (FriendFace), which is designed to automatically forward personal data (e.g., user address) from the online pharmacy to FriendFace once the application has been activated by the customer. It uses such a data for making a patient's profile, being useful for advert targeting purposes.

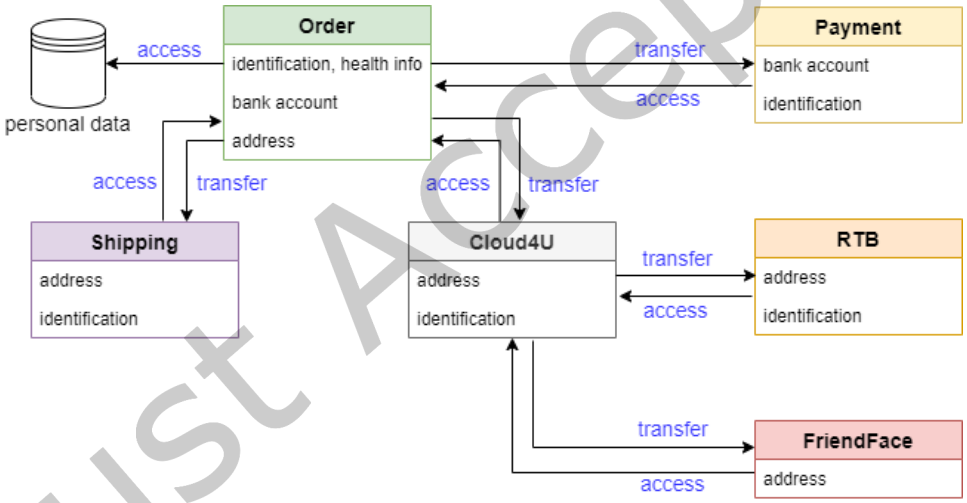

Fig. 2. Interactions within a cloud-based pharmacy

The pharmacy contains a real-time bidding (RTB) system of a prominent online advertiser and intermediary (Froogle) to offer advertising inventory space on its application. When a user accesses the pharmacy's app, RTB cookies and tracking pixels are placed on the user's device so as to enable the publish of some personal data to advertising companies. The data involves user's address and identification (e.g., name, ID etc.), which are received from Cloud4U. Technically, the Froogle enables the broadcast of personal data on behalf of the pharmacy.

The online pharmacy contains payment and shipping service providers to manage the payment and send prescriptions to customers, respectively. The payment service provider receives the patient's identification

and bank account details to run the payment process. The shipping service provider gets the patient's identification and address to pack the order and deliver it.

We consider two typical operations requiring to be checked for GDPR compliance. Firstly, data access that needs the provision of personal data by the customer to the pharmacy for the order, payment and shipping services. Secondly, data transfer, that moves personal data from the online pharmacy to other providers (sub-contractors).

4 GDPR-BASED DATA PROCESSING MODEL

Cloud providers normally collect and manipulate their users' personal data in order to offer them a composite service realizing the users' requirements. The operations of providers on personal data can be classified as *access*, *transfer* and *use*. Precisely, the *access* operation refers to the receipt of personal data by an actor for the purpose of processing. The *use* operation, however, includes any type of data processing activity (e.g., profiling, automated preparation/adjustment, etc.) after accessing the data. Such data may be used on the provider host locally or *transferred* to another provider(s) for further processing.

With increasing the number of cloud-based services, numerous personal data items, varying from name and phone number to trade union information, are collected and processed by providers. However, the audit trail of providers with respect to the manipulation of such multitudinous data items has become a very complex and time consuming process. To address this, a set of general data categories (e.g., identification, address, health etc.) can be designated for these tremendous detailed items so as to facilitate their verification. Given this, we can readily track what data categories are requested by providers and what are their sensitivity levels in GDPR. As an instance, the healthcare data category is sensitive in GDPR standard and its protection by both data controllers and processors is very significant.

The process of cloud providers on the personal data categories of their users in a cloud composite service can formally be defined as follows.

Definition 1. The processing model of cloud providers on users personal data is represented by a tuple of $\mathcal{P}_s = \langle P_s, S_s, U_s, A_s, D_s, R_s \rangle$, where P_s is a set of cloud providers involving in a composite service s, S_s is a set of cloud services offered by the providers through s, U_s is a set of users using the composite service, A_s is a set of operations on users data, and D_s is a set of collected or processed users data categories. $R_s \subseteq P_s \times S_s \times U_s \times A_s \times D_s$ is a relation set denoting what provider for which service executes what operation on which user data category.

Regarding the definition, we can express the requests of cloud providers on user data within a composite service s through the relation set R_s.

Example 1. Consider a composite service s for the online pharmacy presented in Sect. 3. The providers involving in s are order service provider p_o, payment service provider p_p, shipping service provider p_s, Cloud4U service provider p_c, RTB service provider p_r and FriendFace service provider p_f. For an user u_s, a subset of relation set can be:

$$\{\langle p_o, order, u_s, access, identification\rangle, \langle p_p, payment, u_s, access, bank\ account\rangle,$$
$$\langle p_c, RTB, u_s, transfer, address\rangle, \langle p_f, friendface, u_s, access, address\rangle,$$
$$\langle p_r, RTB, u_s, access, identification\rangle, \langle p_s, shipping, u_s, access, address\rangle\} \subset R_s.$$

4.1 User Consent

GDPR enforces cloud providers to clearly identify their purposes of data processing for end users. Moreover, it encourages users to explicitly clarify their positive or negative consents (votes) over the purposes before any data processing by providers. In fact, the users' votes should be kept as an evidence for future verification.

Given Def. 1, the purposes of data processing determined in a composite service s can be expressed by the relation set $\mathcal{R}_s \subseteq R_s$. A formal representation for giving user vote to the relation set (purposes) can be defined.

Definition 2. Let $\mathcal{P}_s$ be the processing model of providers on user data in a composite service s and $\mathcal{R}_s \subseteq R_s$ be the relation set showing the purposes of data processing of providers on users' personal data in s. The positive or negative consents of a user $u_s \in U_s$ to such purposes can be expressed by a boolean function:

$$\mathcal{V} : \mathcal{R}_s \rightarrow \{\top, \bot\}.$$

Let $r_s = \langle p_s, s_s, u_s, a_s, d_s \rangle \in \mathcal{R}_s$ be a purpose of data processing determined by $p_s \in P_s$ for service $s_s \in S_s$ that operation $a_s \in A_s$ will be executed on data category $d_s \in D_s$. A positive consent is given to the purpose by u_s if $\mathcal{V}(r_s) = \top$.

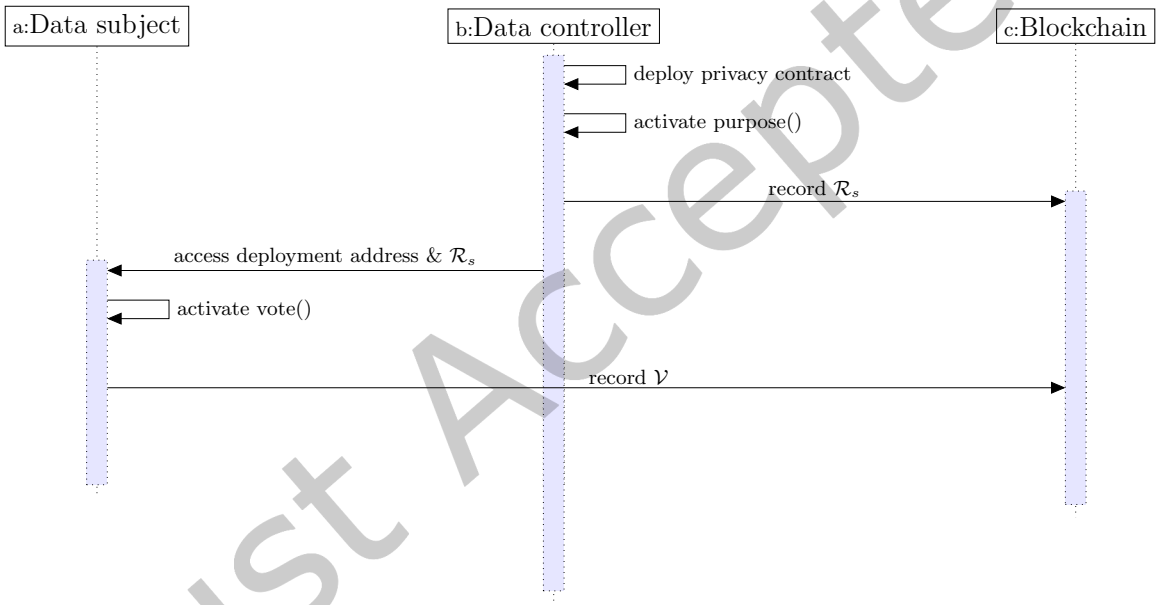

Fig. 3. A protocol for recording user consent

4.1.1 Implementation of User Consent. As a technical solution, one of the transparent and verifiable tools for keeping the user votes for providers' purposes is Blockchain. Figure 3 represents a sequence diagram depicting the interactions between user (data subject), a provider (actor) with the role of data controller and a Blockchain for receiving user consent. As seen, a smart contract, referred to as `privacy`, is implemented and deployed through the portal/ interface of cloud composite service, where data controller by activating a function, called as `purpose()`, can send the information making data processing purposes set $\mathcal{R}_s$ into Blockchain. The contract contains another function, called as `vote()`, that is executed by data subjects for submitting their votes $\mathcal{V}$ based on the purposes set $\mathcal{R}_s$ to the Blockchain. Before activating

the function, the data controller provides data subjects with the deployment address of privacy contract and $\mathcal{R}_s$.

Due to the immutability feature of Blockchain and improving data privacy, an anonymized version of users and providers identifications are kept in a Blockchain. Moreover, the values assigned to personal data items (category) are not sent into the Blockchain. Several techniques are available for data anonymization [19] such as hashing, permutation among others, each of which can be used for mapping the user's and provider's identifications into the anonymized ones. For instance, the values of p_s and u_s in $\mathcal{R}_s$, submitted to Blockchain, refer to the hashed/ permuted version of provider IP and user ID, respectively.

4.2 Data Access Regulation

Cloud providers with the roles of both controllers or processors must implement an encryption mechanism for protecting the personal data of their users before any data access or usage (Art. 32(1)(a) of GDPR). Moreover, the providers should already receive explicit positive consents from the users when they want to access or use personal data (Recital (32), (43) of GDPR).

The following definition expresses the encryption status of a cloud provider on user personal data by a formal representation.

Definition 3. Let $\mathcal{P}_s$ be the processing model of providers on users' data in a composite service and R_s be a relation set showing the requests of providers to personal data. The encryption status of data by providers through the access requests, can be a boolean function:

$$\mathcal{E} : R_s \rightarrow \{\top, \bot\}.$$

Given an access request $r_s = \langle p_s, s_s, u_s, a_s, d_s \rangle \in R_s$, the provider $p_s \in P_s$ encrypts $d_s \in D_s$ if: $\mathcal{E}(r_s) = \top$.

The GDPR compliance of data access operation is subject to both user consent and data encryption.

Definition 4. Assuming that $r_s = \langle p_s, s_s, u_s, a_s, d_s \rangle \in R_s \cap \mathcal{R}_s$ is a data access request. The provider p_s is allowed to access to d_s if:

$$\mathcal{V}(r_s) = \top \ \wedge \ \mathcal{E}(r_s) = \top.$$

4.3 Data Transfer Regulation

GDPR sets a number of obligations for transferring data among actors (e.g., cloud providers). In addition to user consent, the location of data receiver, namely inside or outside Europe, and the certifications of Binding Corporate Rules (BCR) for non-European ones are really significant (Art. 44–47 of GDPR). The BCR is a code of conduct accepted by a group of multinational institutes intending to transfer personal data internationally over different jurisdictions [31]. Given such GDPR concerns, the following definition formally presents the compliance of data transfer.

Definition 5. Let $r_s = \langle p_s, s_s, u_s, a_s, d_s \rangle \in R_s \cap \mathcal{R}_s$ be a processing model relation of $\mathcal{P}_s$, where a_s is a data transfer operation. Moreover, let $\rho_s \in P_s$, $EU(r_s)$, and $BCR(r_s)$ be, respectively, the data receiver of d_s, a boolean value expressing whether ρ_s is inside Europe ($EU(r_s) = \top$) or not ($EU(r_s) = \bot$), and a boolean value denoting whether ρ_s holds a BCR certification ($BCR(r_s)=\top$) or not ($BCR(r_s)=\bot$). The transfer of d_s from p_s to ρ_s is GDPR compliant if:

$$\mathcal{V}(r_s) = \top \ \wedge \ EU(r_s) = \top \ \wedge \ BCR(r_s) = \top.$$

4.4 Verification of Regulations

Cloud providers through a composite service can send their requests for data processing operations (e.g., access, transfer, etc), where the format of each request is formally denoted by a relation belonging to R_s.

As earlier mentioned, the GDPR obligations for data access and transfer requests are expressed using a number of boolean responses, received from both user and provider. For instance, the GDPR issues associated with *data access* request $r_s \in R_s$ are data encryption $\mathcal{E}(r_s)$ and user consent $\mathcal{V}(r_s)$, which forms the attributes of the request. Formally, a set of GDPR attributes for R_s is denoted by $Att_s = \{att_1, \cdots, att_n\}$ and each GDPR attribute att_i can be assigned by a boolean value.

The behavior of providers involving in a composite service s for submitting their requests can be abstracted by a timed automaton. Such an abstraction facilitates the GDPR compliance verification of providers with the aid of model checking tools.

Providers' behavior is an abstraction that shows the order of data processing requests demanded by providers within a composite service in the execution time. Formally providers' behavior of composite service s is a tuple $\mathcal{B}_s = \langle B_s, b_{0_s}, F_s, C, R_s, Att_s, \gamma_s, \delta_s \rangle$, where B_s is the finite set of states showing the progress of demanded requests by providers; b_{0_s} is the initial state where the first request is sent from; $F_s \subseteq B_s$ is the set of final states where the sequence of requests is successfully terminated; C is a set of clocks; $R_s \subseteq P_s \times S_s \times U_s \times A_s \times D_s$ is the set of requests; Att_s is a set of GDPR attributes associated with the requests; $\gamma_s : R_s \rightarrow 2^{Att_s}$ is a function assigning a subset of attributes to a request; $\delta_s \subseteq B_s \times R_s \times G(C) \times B_s$ is the transition relation, where $G(C)$ the set of constraints over C.

The notation $\langle b_s, r_s, g, b'_s \rangle \in \delta_s$ denotes when the request r_s is submitted from state b_s by provider p_s with the clock constraint g, the submission of the request progresses $\mathcal{B}_s$ to a successor state b'_s.

A desired target behavior as a GDPR-compliance specification can be derived from providers' behavior under which the providers' requests and their orders are completely those claimed from the purposes of data processing. Moreover, the realization of a request is subject to the compliance of its attributes with GDPR.

Target behavior formally is a tuple $\mathcal{B}_t = \langle B_t, b_{0_t}, F_t, C, R_s, Att_s, \gamma_t, \delta_t, \partial_t \rangle$, where R_s is the set of providers' requests claimed through the purposes of data processing and ∂_t is a set of guards over R_s based on their GDPR attributes Att_s. Such guards are expressed as boolean functions under which the GDPR compliance of requests hold based on Definitions 4 and 5.

Upon the submission of a request by a provider behavior, the target behavior only permits the progress of the request if its attributes are GDPR compliance.

4.4.1 Implementation and Verification in Uppaal. The interactions between the providers' behavior and target behavior for sending and checking requests can be simulated in Uppaal [30]. The choice of this model checker is that it provides an environment for modeling and verification of timed finite state machines.

A binary synchronization channel for a request $r_s \in R_s$ is declared as "`chan` r_s" in Uppaal. The handshaking between the providers' behavior and target behavior through the channel of r_s is represented by "r_s`!`" on the transition of sending automaton (providers' behavior) and "r_s`?`" on the transition of receiving one (target behavior). The GDPR attributes of r_s with their boolean values appear on the related transition in the providers' behavior. Similarly, such attributes with their GDPR-compliance values appear as guards on the target behavior's state (i.e., the destination of taken transition). A clock is declared as "`clock x`" in the declaration part of the timed transition system. Moreover, a clock constraint (e.g., $l_1 \leq x \leq l_2$) appears on the transition of automaton. In Uppaal simulation, transitions can be taken in a manual mode (i.e. step-by-step) to validate the interactions or an automatic mode (i.e. a continuous way) to quickly identify possible deadlocks.

Figures 4 and 5 show the implementation of providers' behavior and target behavior in Uppaal. As seen, the format of a request in providers' behavior is "$p_s\_s_s\_u_s\_a_s\_d_s$!" and it is "$p_s\_s_s\_u_s\_a_s\_d_s$?" in the target behavior. For the aim of simplicity, we consider a request with multiple data categories as only one transition. The clock constraints over the transitions in the providers' behavior represent the deadlines at which the request and its GDPR attributes are completed and submitted to the system. Likewise, the time constraints in the target behavior emphasize on the process and check of requests after their submission deadlines determined in the providers' behavior. To handle such constraints, two different clocks (i.e., x and y) are declared in our implementation. Moreover, once a transition has been taken in the providers' behavior, the clock is reset. On each state of target behavior[1], an invariant as a guard appears in order to check the GDPR compliance of received request. The satisfaction of such guards allows the acceptance and progress of requests released by the providers' behavior.

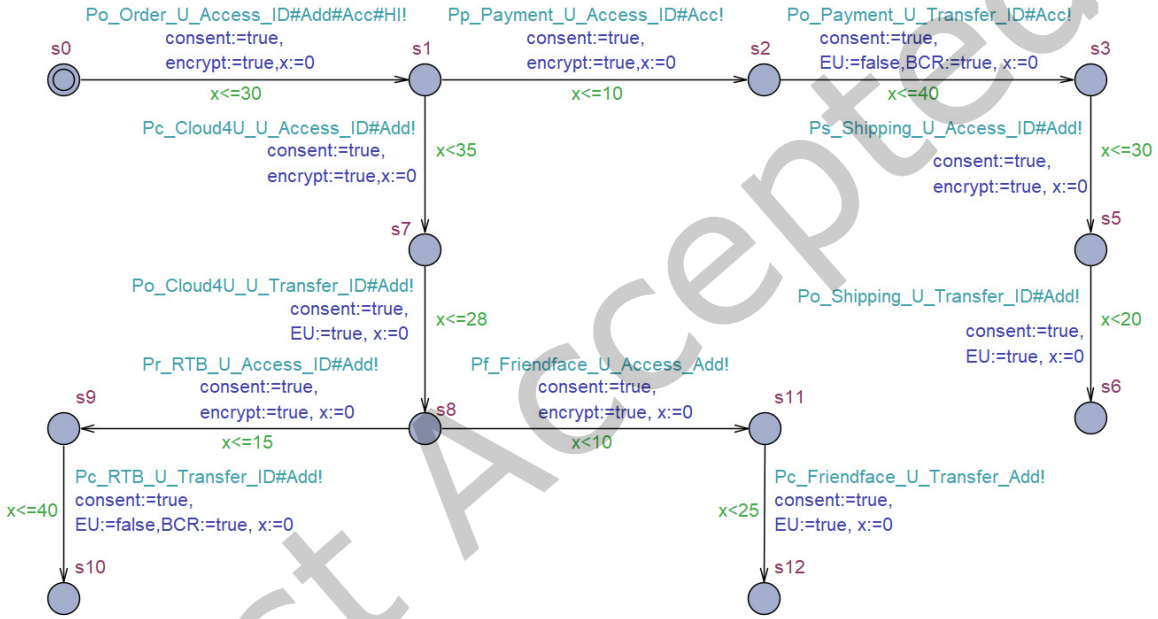

Fig. 4. Providers' behavior of online pharmacy in Uppaal

After the implementation of providers' behavior and target behavior, several verification can be undertaken to detect non-GDPR compliance requests. In case, there exists a deadlock during the execution of interactions between the target and providers' behavior, the following state formula can be verified to detect which requests of providers violates GDPR:

$$\texttt{E<>}\ \mathcal{B}_s.b_s\ \texttt{and not deadlock} \tag{1}$$

The formula should be checked on all states in the providers' behavior $\mathcal{B}_s$ in order to flag all possible GDPR violations under consideration for this work. Its dissatisfaction on a particular state b_s denotes the non-GDPR compliance of the request that its transition directly reaches b_s.

[1]The initial state does not involve any invariant.

Example 2. The formula 1 was checked on all the states of providers' behavior and it was satisfied. We assume that the payment provider P_p is a non-European actor who does not hold a BCR certification (i.e., `BCR:=false`). In this case, Formula 1 is not satisfied from state `s2` in the providers' behavior and the transition "`Po_Payment_U_Transfer_ID#Acc!`" cannot be taken, since the guard (`BCR==true`) appearing in the state `t3` of target behavior is violated.

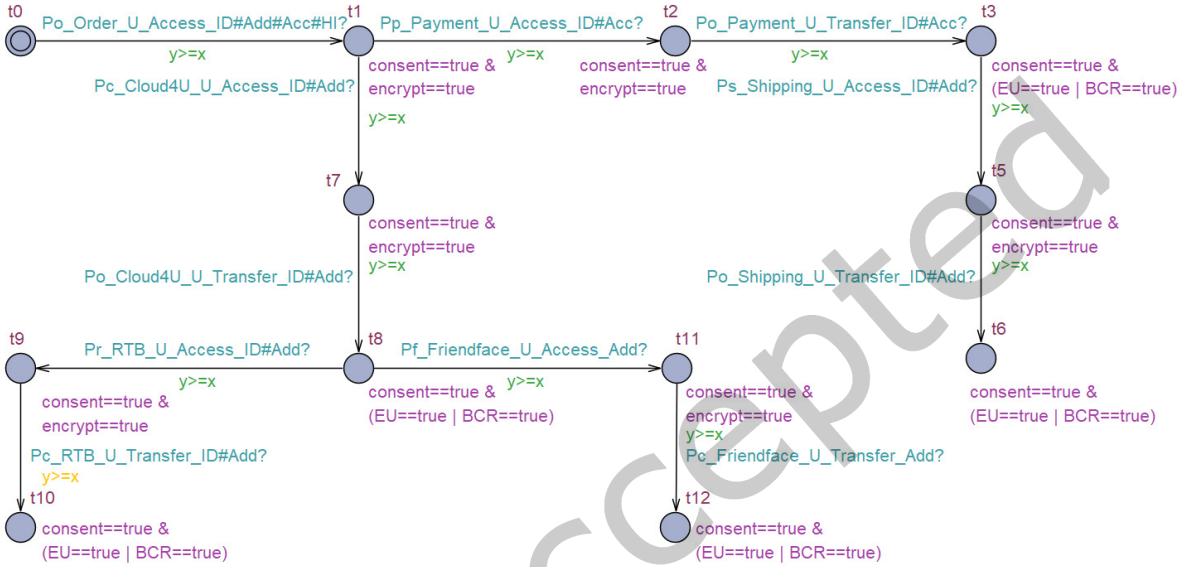

Fig. 5. Target behavior of online pharmacy in Uppaal

5 GDPR-COMPLIANCE IMPLEMENTATION

This section presents an implementation conforming to our proposed GDPR-based processing model. Given a sequence of requests releasing from cloud providers, a container-based framework is implemented to check the GDPR compliance of providers' behavior. The container monitors, filters and accepts the requests in accordance with the guards exposed by the target behavior. In other words, the core of our designed container will fully emulate the target behavior in terms of both data access and data transfer requests.

5.1 Container Architecture

The container-based architecture for data accountability of cloud providers is represented in Fig. 6, where the container hosts the personal data of cloud users. The architecture has appended a new layer, called *privacy monitoring*, to the basic cloud containers' architecture in order to provide the audit trail of cloud providers. Containerization of the privacy monitoring tool breaks the software's dependency on underlying host devices to ensure compatibility with multiple cloud providers. The key components of the layer are described below.

Gateway server is the entry-point for data processing requests (e.g., access, transfer, etc.) of providers P_s belonging to a composite service s that are released into the layer. Formally, each request contains

the items making the relation $r_s = \langle p_s, s_s, a_s, d_s \rangle \in R_s$. Each released request can be modeled with a transition in the providers' behavior (i.e., r_s!).

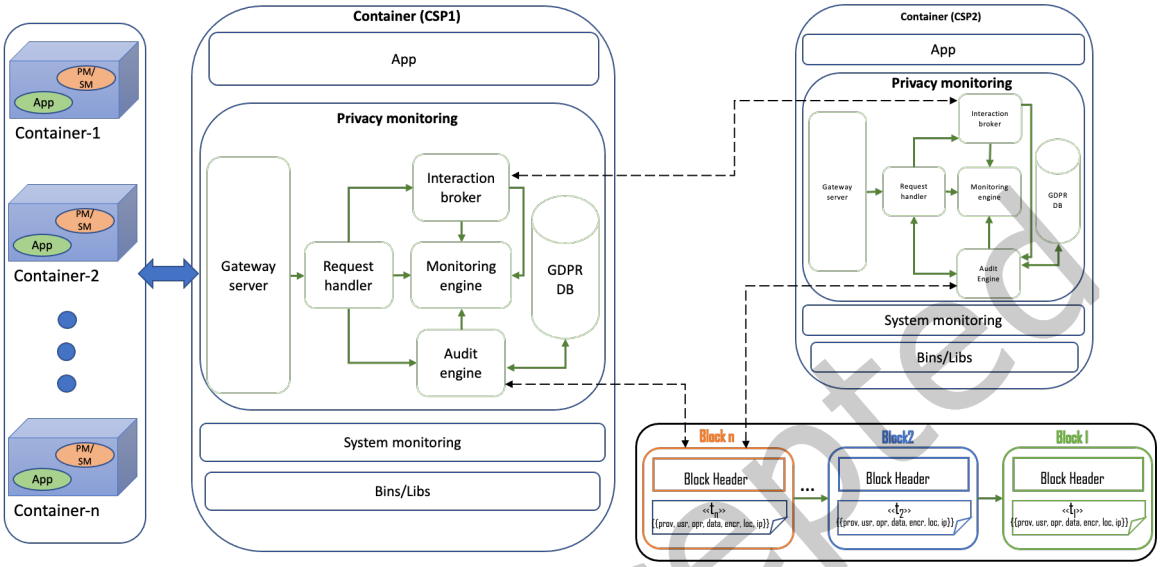

Fig. 6. A container-based architecture

The gateway provides the connectivity between the providers and the internal components of the privacy monitoring layer. It is implemented using the restlet framework which supports major web standards like HTTP (HyperText Transfer Protocol), XML (Extensible Markup Language), JSON (JavaScript Object Notation), SMTP (Simple Mail Transfer Protocol), WADL (Web Application Description Language) and Atom and benefits from its simplicity and scalability [32]. The requests that arrive at the gateway are passed on to the request handler component. Notably, the communications with the gateway server is based on HTTP protocol.

Request handler is responsible for analysing and parsing the requests. This component manages the parsing of various requests for access and transfer operations by exposing a set of API service endpoints for the various operations. It interacts with the other components of the privacy monitoring layer to process the request and send a feedback to the provider via the gateway. The request handler provides the audit engine component with the information making the relation $\langle p_s, s_s, u, a_s, d_s \rangle$ set based on the released request "r_s!". The request handler also receive the GDPR attributes (e.g., $\mathcal{E}(r_s)$, $EU(r_s)$ etc.) associated with the request from providers to complete the transitions of providers' behavior.

Interaction broker communicates with the interaction brokers of external third-party service providers, involving in a composite service, to receive or send the requests from or to the external service providers.

For synchronous events that need feedback from third party providers to realize a request, the transaction is put on hold at the request handler of the primary service provider until a feedback is issued whether a data access or data transfer is permitted or not. Asynchronous requests that do not need the feedback from the third parties are realized within the primary service provider and updated later on getting feedback from the third-party providers.

Audit engine deploys an enforcement contract, called *compliance*, for issuing a grant access or deny access

to the requested personal data items. In fact, the audit engine through the contract checks whether the requests are GDPR compliant or not. In the case of non-compliance, the request is denied. This component can simulate the states in the target behavior, involving the GDPR guards.

Through the contract, the information related to the request together with its GDPR status (compliance or non-compliance) is sent into a Blockchain to provide a basis for the further verification of cloud providers. Moreover, a copy of such information along with some additional data (e.g., original users ID, providers IP etc.) are stored in an secure local storage, called as *GDPR database*.

Monitoring engine tracks all activities that happen within the layer and sets a fault tolerance mechanism when any component in the layer encounters with a failure.

5.1.1 Implementation Follows Design. Given the implementation of the container's architecture, we can discuss about the conformity or map between our proposed components in the architecture and the automaton-based models presented for the target and providers behaviors in the design time.

Assuming that $\langle b_s, r_s, g, b'_s \rangle \in \delta_s$ is a transition in the providers' behavior $\mathcal{B}_s$ and $\langle b_t, r_s, g, b'_t \rangle \in \delta_t$ is its equivalent transition relation in the target behavior $\mathcal{B}_t$, and $Att \subseteq Att_s$ is the GDPR attributes' set relevant to r_s. The state b_s denotes that r_s is made by provider p_s and is ready to be picked up from the gateway server. The state b'_s shows the receipt of (r_s, Att) by the request handler. Regarding the target behavior, b_t is a state in which the request handler listens to forward (r_s, Att) to the audit engine. Finally, b'_t is a state through which a preventative smart contract as a guard is executed by the audit engine in order to verify the GDPR compliance of Att.

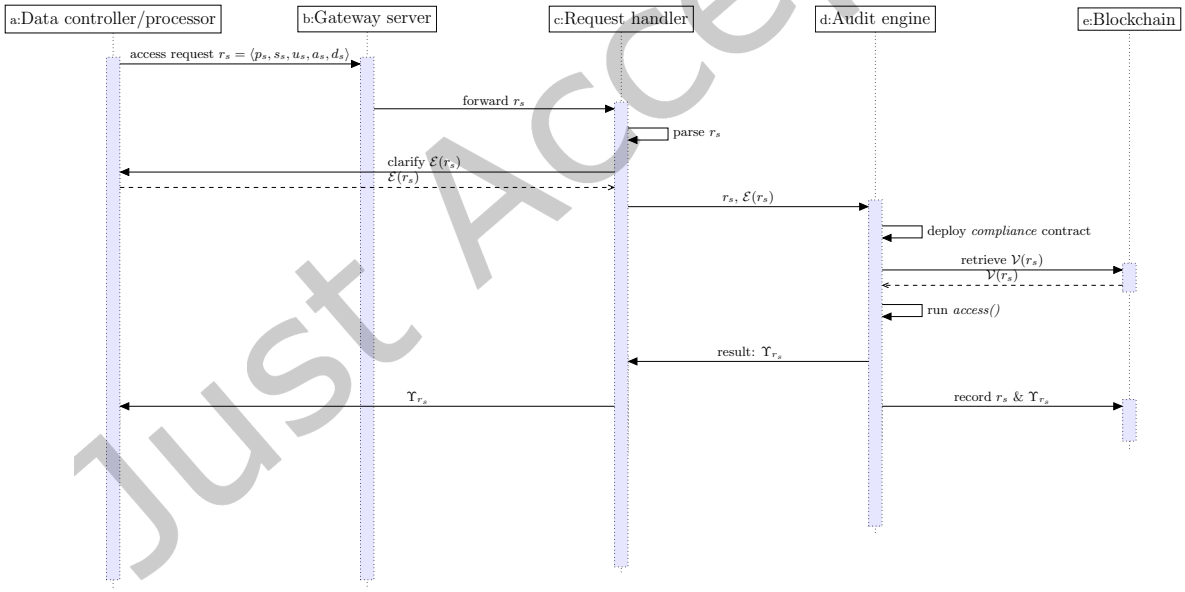

Fig. 7. A protocol for data access

5.1.2 Realization of Data Access. For data access requests, an enforcement smart contract, called *compliance* here, can be implemented in practice and deployed by the audit engine component in the architecture so as to get the access permission only to the providers following the GDPR requirements (i.e., positive

consent and data protection). Such a smart contract can be a real implementation of the guards appeared on the target behavior's states, which are the destination of data access requests.

The protocol depicted in Fig. 7 is proposed to show how a data access request is handled by the components existing in the privacy monitoring layer. First of all, the request $r_s = \langle p_s, s_s, u, a_s, d_s \rangle$, where a_s is data access operation, is delivered by a provider (data controller/ processor) to the request handler. The request is, then, being parsed by the handler and the encryption status of the provider p_s for protecting d_s is demanded. Upon the receipt of $\mathcal{E}_{p_s}(r_s)$ by the handler, both the request and encryption status are delivered to the audit engine. These information together with the user vote $\mathcal{V}(r_s)$, already recorded in Blockchain, make the input of `access()` function, which belongs to the *compliance* contract and is activated by the audit engine. Finally, an allow or deny access to d_s by p_s is sent to the provider via the request handler. A copy of the request and the permission result are, also, recorded in the Blockchain for future verification. Notice that both p_s and u_s are the anonymized versions of original provider IP and user ID, respectively.

Recording the access request and its GDPR compliance status (allow ($\Upsilon_{r_s} = \top$) or deny ($\Upsilon_{r_s} = \bot$) result) in the Blockchain provides a transparent evidence for users to be informed what providers are processing their data. Such records can realize the right of access in GDPR under which data subjects are allowed to track the providers accessing/ manipulating their personal data (Art. 15 of GDPR). In case of a deny access, keeping the record in Blockchain can be used for data controllers or any trusted data protection officer to automatically flag non-GDPR compliance providers.

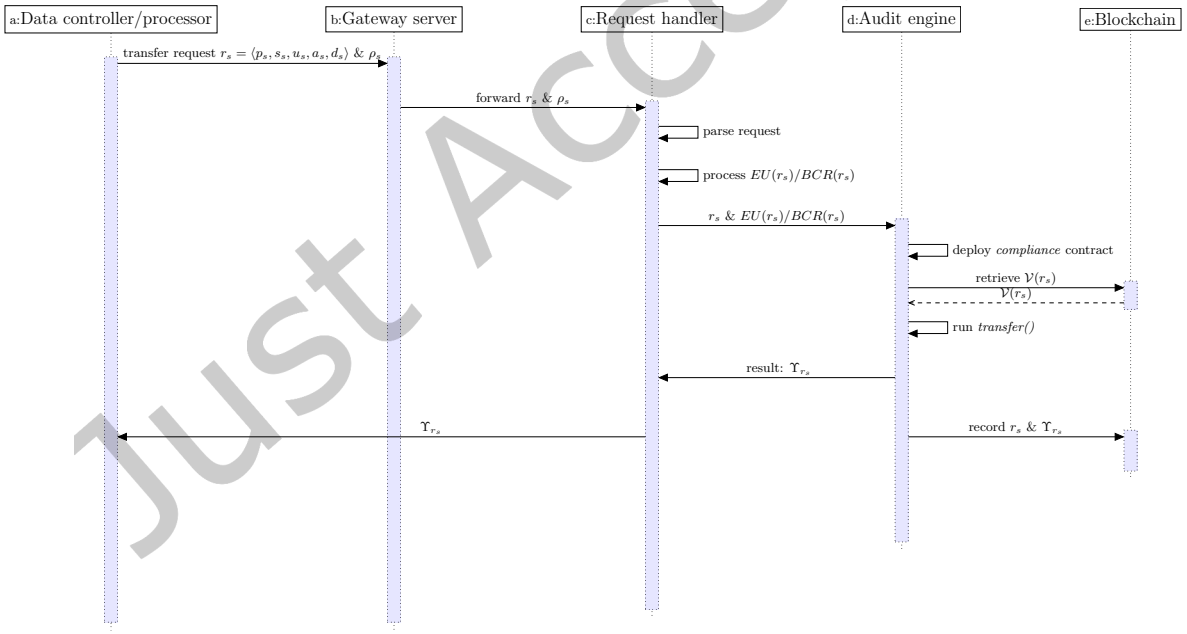

Fig. 8. A protocol for data transfer

5.1.3 Realization of Data Transfer. For the implementation of GDPR-oriented data transfer using Blockchain, a new function, called as `transfer()`, can be implemented inside the *compliance* contract deployed by the audit engine so as to run an enforcement and preventative mechanism for data transfer requests. The sequence diagram demonstrated in Fig. 8 represents a protocol for data transfer. As seen, upon the receipt of data transfer request r_s by the request handler, the data receiver (ρ_s) IP is processed to learn whether it is inside Europe or not. In case of non-European data receiver, its BCR status is verified by the request handler. Following that, both $BCR(r_s)$ and $EU(r_s)$ are sent to the audit engine that are two inputs of `transfer()` function. The other input is user consent retrieved from the Blockchain. Finally, the audit engine sends a feedback, namely accept ($\Upsilon_{r_s} = \top$) or reject ($\Upsilon_{r_s} = \bot$), to the request handler. A copy of the request together with the feedback are also recorded in the Blockchain.

When a data transfer request is successful, the interaction broker monitors the data categories $\mathcal{D}_s \subseteq D_s$ which have been transferred from a cloud provider to another provider(s). The broker submits the monitored data $\mathcal{D}_s$ to the audit engine in order to be recorded in a Blockchain. To this end, a new function, called as `track()`, can be implemented within the *compliance* contract to directly send $\mathcal{D}_s$ into the Blockchain for the verification purposes.

5.1.4 A Classification for Violated Requests. A violation, can be defined as the non-compliance activities of actors (cloud providers) on users' data with regards to the GDPR regulations for access and transfer operations. User consent is a key concern in GDPR that must be given to a purpose of data processing. Hence, the data access or transfer requests that are submitted by cloud providers should already have received the positive vote from data subject. If we consider a higher priority for the consent obligation compared to the other GDPR obligations aforementioned for access and transfer requests (e.g., $EU(r_s)$ etc.), it can be defined a classification for detected violations. So, we can classify the requests violating GDPR requirements into three classes: *high risked violation*, *medium risked violation* and *low risked violation*. The high risked violations, denoted by V_h, are the requests that did not get a positive consent and also are non-GDPR compliance even if we ignore the consent obligation. The medium risked violations, denoted by V_m, refer to the requests that did not receive positive consent even though the other GDPR conditions (e.g., $\mathcal{E}_{p_s}(r_s)$) are satisfied. Finally, the low risked violations, denoted by V_l, are the requests that got positive consents while they do not comply with GDPR.

In order to detect such classified violations in an automatic way, the compliance smart contract is extended using a new function, called as `breachClass()`, for flagging violators based on their category. In this end, Algorithm 1 is proposed to demonstrate how the function identifies the category of non-compliance access and transfer requests (i.e., V_h, V_m and V_l sets). Such classified violations' sets facilitate the audit trail of cloud providers for legal trusted third parties in a more transparent and automatic way. In addition, the main data controller(s) can be informed about the activities of their subcontractors and their illegal transfer/ access requests to customer data.

6 EXPERIMENTAL RESULTS

The evaluation of our approach is conducted using three studies. The first study is carried out at the design step and utilizes simulation tests to analyze the average time and memory usage for GDPR-compliance verification with Uppaal. The remaining two studies focus on analysis of post-implementation. The second study assesses the average processing time under different scales of requests. It is implemented by Ropsten test network that has free Ethers obtained from available faucets [33]. The third study focuses on analyzing the costs for deployment of the smart contracts and running the various contract functions. Also the CPU usage, memory usage and throughput are observed here to determine how they are affected or changed by changing the number of requests. This study makes use of Ganache to provide a fast local

Algorithm 1 Identifying Violation classes

$V_h, V_m, V_l \leftarrow \emptyset$
for *all* $r_s \in R_s$ **do**
  case $access$:
    if $(\mathcal{V}(r_s) = \bot \ \& \ \mathcal{E}_{p_s}(r_s) = \bot)$ **then**
      $V_h \leftarrow V_h \cup \{r_s\}$
    else if $(\mathcal{V}(r_s) = \bot \ \& \ \mathcal{E}_{p_s}(r_s) = \top)$ **then**
      $V_m \leftarrow V_m \cup \{r_s\}$
    else if $(\mathcal{V}(r_s) = \top \ \& \ \mathcal{E}_{p_s}(r_s) = \bot)$ **then**
      $V_l \leftarrow V_l \cup \{r_s\}$
  case $transfer$:
    if $(\mathcal{V}(r_s) = \bot \ \& \ (EU(r_s) = \bot \ \& \ BCR(r_s) = \bot))$ **then**
      $V_h \leftarrow V_h \cup \{r_s\}$
    else if $(\mathcal{V}(r_s) = \bot \ \& \ (EU(r_s) = \top \mid BCR(r_s) = \top))$ **then**
      $V_m \leftarrow V_m \cup \{r_s\}$
    else if $(\mathcal{V}(r_s) = \top \ \& \ (EU(r_s) = \bot \ \& \ BCR(r_s) = \bot))$ **then**
      $V_l \leftarrow V_l \cup \{r_s\}$
return V_h, V_m, V_l

Blockchain with auto mining for creating blocks. Ganache is a fast and customizable Blockchain emulator which allows calls to the Blockchain without the overheads of running an actual Ethereum node [34].

6.1 Study 1: Evaluation of Time and Memory for Design Time Verification

This experiment investigates the average time and memory used for verifying GDPR-compliance constraints by changing the number of data access and transfer requests. The number of requests varies from 10 to 50 and the average results were calculated after five times simulations and verification in Uppaal. For each simulation test, the requests were randomly selected from data access and transfer and a clock constraint was assigned to each request—appearing on a transition. The state formula presented in Eq. 1 was tested on each state of the automaton modelling providers behavior. Table 1 demonstrates the average required time and memory for verifying the state formula on the transition systems. As seen, when the number of transitions (or states) rises, the used time and memory increases slightly. The amount of time elapsed for the verification is low and this may motivate cloud developers to check the GDPR-compliance of their abstracted systems in a pretty short time.

6.2 Study 2: Evaluation of Processing Time for GDPR Verification

This experimental setup uses test sets by changing the number of requests (concurrent compliance verification checks) ranging from 20 to 200 to calculate the processing time of the system developed. The Ropsten test network was used to emulate, as closely as possible, a realistic workflow. Such a test network utilizes proof-of-work (PoW) consensus algorithm for validating transactions which makes it the best like-for-like representation of the Ethereum Mainnet. Free ethers for test purposes were obtained online via Ropsten faucets. The total processing time ($time_t$) is calculated as:

$$time_t = time_s + time_m \tag{2}$$

Table 1. The used memory and time for verification

Transitions	Time (seconds)	Memory (KB)
10	9.02	8312
15	13.05	8344
20	17.06	8548
25	22.07	8804
30	28.09	8870
35	33.1	9140
40	39.12	9152
45	43.13	9167
50	47.15	9186

The notation $time_s$ represents the processing time at the server (excluding the verification request submission to the Ethereum network). This is calculated as the time interval between which a request reaches the gateway server and the time at which a result is derived. The transaction mining time ($time_m$) for each compliance verification request is obtained directly from the network by submitting the requests to the Blockchain and checking the transaction details from the Ropsten network.

The overall monitoring framework can be decoupled into two main parts: The API server for receiving and processing verification requests and the Blockchain part for carrying out the compliance verification and logging. The server, which is based on the Restful framework, is scalable and supports multi threading. The values for the server processing time ($time_s$) were measured in milliseconds. Ropsten does not support concurrent handling of transactions from a node, thus each access/transfer request must be processed sequentially. This creates a bottleneck for the overall compliance monitoring tool at the Blockchain.

The results of this experiment are given in Table 2. By increasing the number of requests from 20 to 200, the processing time at server ($time_s$) rises marginally from 1 to 5 seconds owing to the API server's multi-threading feature and scalability. However, the mining time ($time_m$) grows significantly up between any two successive requests increment due to the linear processing of the Blockchain transactions from a node.

Table 2. Total Processing Time

Number of Requests	$time_s$ (secs)	$time_m$ (secs)	$time_t$ (secs)
20	1.69	364	365.69
40	2.44	561	563.44
60	2.76	840	842.76
80	2.97	1142	1144.97
100	3.36	1419	1422.36
120	3.84	1880	1883.84
140	4.21	2004	2008.21
160	4.57	2393	2397.57
180	4.71	2626	2630.71
200	5.06	2943	2948.06

6.3 Study 3: Evaluations of cost, CPU, Memory and Throughput

This section assesses consumed gas for the proposed smart contracts and then investigates the throughput, CPU and Memory Usages. Ganache CLI was used to simulate a full client behaviour. It is lightweight and can be easily deployed as a private Blockchain network running within a container. Ganache provides some free accounts by default, each instantiated with a fixed amount of ether (no need for faucets) and transaction calls to the test Blockchain are created instantly. The testchain events can also be observed via its GUI [34].

Gas refers to the computational effort, or fee, required to execute a contract or run a transaction on the Ethereum network. It is normally priced in *wei* which is a small fraction of the cryptocurrency ether. The transaction costs for the smart contracts developed as part of this compliance monitoring tool are assessed here to gauge their optimality. Table 3 shows the amount of gas used for running the functions involving in the *privacy* and *compliance* smart contracts together with the costs for their deployment. A higher gas value was computed for the privacy contract owing to the fact that it has more functions (opcodes) than the compliance contract. The privacy contract also required a higher amount of data storage for information such as providers' service details and customer votes.

Table 3. Gas fees for various operations

Transactions/ Contracts	Gas used (wei)
Compliance contract	1522190
Privacy contract	1735263
Purpose()	299560
Retrieve()	106214
Vote()	271807
Access()	63803
Transfer()	66467
Breach()	136891

The rest of this section investigates the throughput (tps), CPU and Memory usages. Although these metrics are influenced by the host device used for testing, they provide a comparable assessment for the compliance monitoring tool when the host device's specifications are known. In this study, the evaluations were carried out on an M1 MacBook Air, 2020 running MacOS BigSur v11.3.1 OS with 8 core CPU (4 performance and 4 efficiency cores) and 8GB memory.

Throughput is measured as the number of successful requests per second starting from the first request's invocation time. For this metric, we compare the value recorded for an arbitrary number of 250 requests for two different scenarios: (i) All service provider requests (for data access/transfer) are GDPR compliant, and (ii) None of the requests are GDPR compliant with regards to user consent obligation. An average throughput of 18.5 transactions per second was recorded for 250 requests when none of the requests are compliant with a lower value of 16.5 tps for the GDPR compliant requests. The higher recorded value is for the scenario where the requests are not GDPR compliant, since after detection of a no user consent for the request, all subsequent compliance checks are ignored. In such a case, the request is cancelled and a failure result is sent back to the requester. This creates lower overheads in terms of processing time for the requests. Thus, the higher throughput as compared to the case where all requests are GDPR compliant and successfully terminate.

The evaluation of scalability metric is essential, as it gives a measure of our monitoring tool's efficiency, quality and competitiveness. The scalability of our proposed approach is measured by deriving the

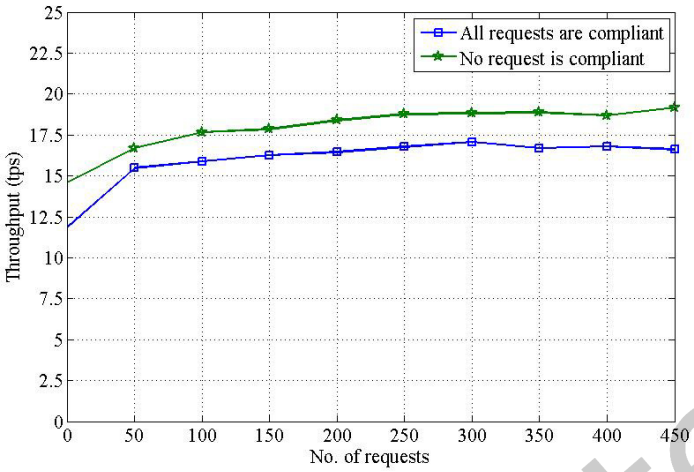

Fig. 9. Throughput of our presented approach

throughputs with incremental number of requests to a maximum of 450. In Figure 9, we observe that the rate of throughput rises sharply up to 15.5 tps when the number of compliant requests reaches 50. In addition, the rate starts to taper off till it gets a maximum of 17 tps at 300 and before starting to gradually drop for higher numbers of GDPR-compliance requests. The throughput rates totally vary from 12 to maximum of 15.5 tps. Given the trend for the non-compliance requests, it is observed that the throughput rates are higher than the ones for the scenario with the compliance requests. The rate of throughput also appears to still have an increase after the 450 requests cut off due to the early termination of requests prior to their completion. So, more requests are handled within a given time interval. The behaviours observed for the two scenarios above are influenced by the scalability of the RESTful API service for receiving compliance monitoring requests and handling them and also the auto mining feature of Ganache that allows fast compliance verification through the Blockchain.

6.3.1 CPU and Memory. The variation in CPU usage for the verification of GDPR-compliance requests is depicted in Fig. 10. The CPU usage for a time period of 1200 seconds is measured using the JNA-based Operating System and Hardware Information (OSHI) library for JAVA. This library provides a cross-platform implementation to retrieve system information [35]. The process to collect this metric runs independently of the monitoring tool to prevent it from influencing the processing time of requests. As seen, between 0 and 500 seconds, the CPU usage is collected without any compliance verification processes being run. The CPU usage is recorded to be between 5% and 10% except the occasional blip at 200 which can be described as some other system running process that affected the CPU usage. At time 500 seconds, the monitoring tool starts receiving requests until 800 seconds. The abrupt increase in CPU shown here can be attributed to two main factors; Firstly, the RestLet service was developed with scalability in mind and thus immediately runs the internal components required to satisfy the API compliance requests. In addition, the high number of concurrent compliance requests received at the monitoring tool further influenced the CPU usage. The CPU usage has an increase to an average of 26% within this time window before dropping back down to the normal 5% average CPU usage between 800 and 1200 seconds when there are no more verification checks running.

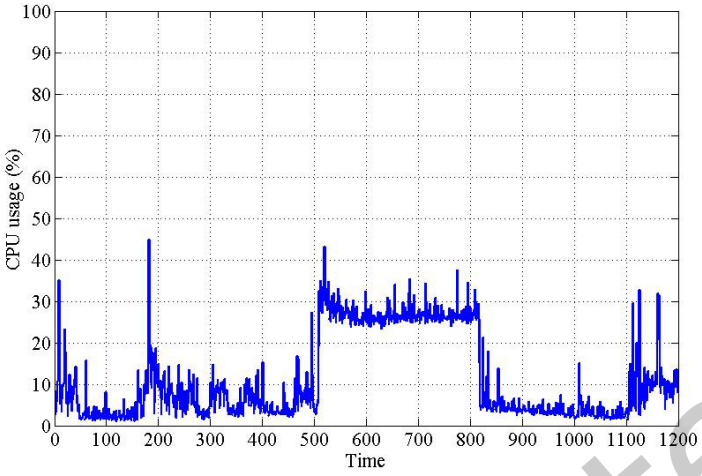

Fig. 10. CPU usage

The effects on memory usage were also accessed using the OSHI library for the same time period and requests described above. The memory usage was found to be affected only minimally when the requests are run for the 300 seconds with the usage staying between 70% and 80% throughout the test scenario.

7 CONCLUSION

The paper formally presented a GDPR-supported model for both data access and transfer requests using timed transition systems. The behavior of such requests within a composite service was abstracted by a timed automaton and its GDPR-compliance with respect to a specification was tested in Uppaal. The pharmacy scenario has been used to demonstrate the complexities involved in the flow of sensitive personal data processing by different cloud service providers in the backend of a cloud pharmacy application. However, our proposed approach can also be used for other scenarios where the protection of users' data and the system's compliance are quite significant. In order to practically implement the model on real cloud environment, we made use of Blockchain and trusted container technologies. Through the former, enforcement smart contracts have been implemented based on the guards appeared in the specification's states to prohibit non-compliance requests and log a copy of each request in a Blockchain for further verification. The latter (i.e., trusted container) enabled a secure mechanism for hosting personal data and deploying the smart contracts. The architecture of such a container has been developed to incorporate a new privacy monitoring layer for managing, checking, filtering and logging cloud requests. We evaluated our proposed approach in both design and implementation stages. There is a direct relation between the number of data access/ transfer requests and the amount of time and memory used for their verification. Moreover, the GDPR-compliance requests got lower rates of throughput compared with those whose providers did not receive user consent. This is due to the verification of all GDPR conditions (e.g., encryption, consent, BCR etc.) for the compliance requests within our container-based framework. We focused on GDPR compliance verification with attention to trust, security, and privacy with the adoption of containerization technology and the benefits that Blockchain provides (e.g., transparency, traceability, etc). However, two limitations can be identified in our solution. First, the processing time results in Table 2 show a scalability issue with using the Ethereum Blockchain which uses the PoW consensus algorithm.

Another limitation of Ethereum is the Maximal Extractable Value (MEV) which is the highest value a miner can extract from changing the position of transactions when creating a block. This is, however, mitigated by Ethereum's move from PoW to proof-of stake consensus algorithm – mitigating associated attacks in this regard.

The future work focuses on developing our designed container's architecture in IoT and edge computing. Furthermore, we plan to propose a graph-based model for connectivity of cloud providers in a composite service–communicating with each other to handle customer data. The deployment of our proposed smart contracts in a more scalable Blockchain network (e.g., Polkadot) remains another challenge for future investigation.

REFERENCES

[1] E. Fosch-Villaronga and C. Millard, Cloud robotics law and regulation: Challenges in the governance of complex and dynamic cyber–physical ecosystems, Robotics and Autonomous Systems, vol. 119, pp. 77–91, 2019.

[2] A., Nuno, L. Balby, F. Figueiredo, N. Lourenco, W. Meira, and W. Santos, Fairness and transparency of machine learning for trustworthy cloud services, In 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W), pp. 188–193, 2018.

[3] L. Elluri, KP. Joshi, A knowledge representation of cloud data controls for EU GDPR compliance, In IEEE World Congress on Services (SERVICES), pp. 45–46, 2018.

[4] M. Virvou and E. Mougiakou, Based on GDPR privacy in UML: Case of e-learning program, In Proc. of the 8th International Conference on Information, Intelligence, Systems & Applications, Larnaca, Cyprus, 2017.

[5] R. Ducato, Cloud computing for s-health and the data protection challenge: Getting ready for the General Data Protection Regulation, In IEEE International Smart Cities Conference (ISC2), pp. 1–4, 2016.

[6] B. Russo, L. Valle, G. Bonzagni, D. Locatello, M. Pancaldi, and D. Tosi, Cloud computing and the new EU general data protection regulation, IEEE Cloud Computing, vol. 5, no. 6, pp. 58–68, 2018.

[7] M. Barati, O. Rana, G. Theodorakopoulos, and P. Burnap, Privacy-aware cloud ecosystems and GDPR compliance, In IEEE 7th International Conference on Future Internet of Things and Cloud, Istanbul, Turkey, pp. 117–124, 2019.

[8] M. Barati and O. Rana, Privacy-aware cloud ecosystems: Architecture and performance, Concurrency and Computation: Practice and Experience, 2020. DOI: 10.1002/cpe.5852

[9] K. Tom, J. Lambrecht, and C. Horn, A privacy-aware distributed software architecture for automation services in compliance with GDPR, In IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA), pp. 1067–1070, 2018.

[10] C. Shirley and J. Jensen, Towards a secure and gdpr-compliant fog-to-cloud platform, In IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion), pp. 296–301, 2018.

[11] W. Zhou, A. B. Williams and D. Perouli, Dependable public ledger for policy compliance, a Blockchain based approach, In 39th International Conference on Distributed Computing Systems (ICDCS), pp. 1891–1900, 2019.

[12] M. Abhishek and K. P. Joshi, Automating GDPR Compliance using Policy Integrated Blockchain, In IEEE 6th International Conference on Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing,(HPSC) and IEEE International Conference on Intelligent Data and Security (IDS), pp. 86–93, 2020.

[13] G.S. Aujla, M. Barati, O. Rana, S. Dustdar, A. Noor, J.T. Llanos, M. Carr and R. Ranjan, COM-PACE: Compliance-Aware Cloud Application Engineering Using Blockchain, IEEE Internet Computing, vol. 24, no. 5, pp. 45–53, 2020.

[14] L. Campanile, M. Iacono, F. Marulli and M. Mastroianni, Designing a GDPR compliant blockchain-based IoV distributed information tracking system, Information Processing and Management, vol. 58, 2021.

[15] N. B. Truong, K. Sun, G. M. Lee and Y. Guo, GDPR-compliant personal data management: A Blockchain-based solution, IEEE Trans. on Information Forensics & Security, vol. 15, pp. 1746–1761, 2020.

[16] M. Barati, G. Theodorakopoulos and O. Rana, Automating GDPR compliance verification for cloud-hosted services, IEEE International Symposium on Networks, Computers and Communications, Montreal, Canada, 2020.

[17] M. Barati and O. Rana, Design and verification of privacy patterns for business process models, In: S. Patnaik et. al. eds., Blockchain Technology and Innovations in Business Processes, Springer, 2021.

[18] M. H. Onik, C.-S. Kim, N.-Y. Lee and J. Yang, Privacy-aware blockchain for personal data sharing and tracking, Open Computer Science, vol. 9, pp. 80–91, 2019.

[19] S. Murthy, A. Abu Bakar, F. Abdul Rahim and R. Ramli, A Comparative study of data anonymization techniques, In Proc. of the 5th IEEE International Conference on Big Data Security on Cloud (BigDataSecurity), High Performance and Smart Computing (HPSC) and Intelligent Data and Security (IDS), Washington, USA, 2019, pp.306–309.
[20] X. Wang, J. He, P. Cheng and J. Chen, Privacy Preserving Collaborative Computing: Heterogeneous Privacy Guarantee and Efficient Incentive Mechanism, IEEE Transactions On Signal Processing, vol. 67, pp. 221–233, 2019.
[21] E. Gaetani, L. Aniello, R. Baldoni, F. Lombardi, A. Margheri, and V. Sassone, Blockchain-based Database to Ensure Data Integrity in Cloud Computing Environments, In Proc. of the First Italian Conference on Cybersecurity, Venice, Italy, 2017, pp. 146–155.
[22] N. Al-Zaben, M. M. H. Onik, J. Yang, N-Y. Lee, and C-S. Kim, General data protection regulation complied Blockchain architecture for personally identifiable information management, In Proc. of the International Conference on Computing, Electronics & Communications Engineering, Southend, UK, 2018, pp. 77–82.
[23] Y. Zhang, S. Wu, B. Jin, and J. Du, A Blockchain-based Process Provenance for Cloud Forensics, In Proc. of the 3rd IEEE International Conference on Computer and Communications, Chengdu, China, 2017, pp. 2470–2473.
[24] Q. Xia, E. B. Sifah, K. O. Asamoah, J. Gao, X. Du and M. Guizani, MeDShare: Trust-Less Medical Data Sharing Among Cloud Service Providers via Blockchain, IEEE Access, vol. 5, pp. 14757–14767, 2017.
[25] E. Rios, E. Iturbe, X. Larrucea, M. Rak, W. Mallouli, J. Dominiak, V. Muntés, P. Matthews, and L. Gonzalez, Service level agreement-based GDPR compliance and security assurance in (multi) Cloud-based systems, IET Software, vol. 13, no.19, pp. 213–222, 2019.
[26] H. Desai, K. Liu, M. Kantarcioglu, and L. Kagal, Enforceable Data Sharing Agreements Using Smart Contracts, arXiv:1804.10645v1[*cs.CY*], 2018.
[27] N. Kumar, G. S. Aujla, S. Garg, K. Kaur, R. Ranjan and S. K. Garg, Renewable Energy-Based Multi-Indexed Job Classification and Container Management Scheme for Sustainability of Cloud Data Centers, IEEE Transactions on Industrial Informatics, vol. 15, no. 5, pp. 2947–2957, 2019.
[28] M. U. Wasim, A. A. Z. A. Ibrahim, P. Bouvry and T. Limba, Law as a service (LaaS): Enabling legal protection over a blockchain network, In Proc. of the 14th International Conference on Smart Cities: Improving Quality of Life Using ICT & IoT (HONET-ICT), Irbid, 2017, pp. 110–114.
[29] M. Barati and O. Rana, Tracking GDPR Compliance in Cloud-based Service Delivery, IEEE Transactions on Services Computing, 2020. DOI: 10.1109/TSC.2020.2999559
[30] G. Behrmann, A. David, and K. G. Larsen, A tutorial on Uppaal, in Formal Methods for the Design of Real-Time Systems, Lecture Notes in Computer Science, M. Bernardo, F. Corradini (eds), Springer, Vol. 3826, 2004, pp. 200–236.
[31] M. Corrales, P. Jurcys and G. Kousiouris, Smart contracts and smart disclosure: Coding a GDPR compliance framework, SSRN Electronic Journal, 2018.
[32] L. Richardson and S. Ruby, RESTful web services - web services for the real world, O'Reilly Media, Inc. 2007.
[33] Ropsten 2021. Ropsten testnet pow chain. *https://github.com/ethereum/ropsten*.
[34] Ganache 2021. Nethereum Documentation. *https://docs.nethereum.com/en/latest/ethereum-and-clients/ganache-cli/*.
[35] OSHI 2021. Operating System & Hardware Information. *https://github.com/oshi/oshi*.