PrSChain: A Blockchain Based Privacy Preserving Approach for Data Service Composition

Rofaida Khemaissia[1*], Makhlouf Derdour[2], Mohamed Amine Ferrag[3], Mohammed Mounir Bouhamed[4]
Email : khemaissia.rofaida@univ-tebessa.dz, derdour.makhlouf@univ-oeb.dz, ferrag.mohamedamine@univ-guelma.dz, bouhamed-mohammedmounir@univ-eloued.dz
[1]Laboratory of Mathematics, Informatics and Systems (LAMIS), Echahid Cheikh Larbi Tebessi University, Tebessa, 12002, Algeria
[2]Larbi Ben M'Hidi University Oum El Bouaghi, Algeria
[3]8 Mai 1945 University Guelma, Algeria
[4]Echahid Hamma Lakhdar University, El-Oued, Algeria

Keywords: service composition, security, data privacy, data integrity, blockchain, hyperledger fabric, IPFS

Received: August 07, 2023

The main goal of a Data Service Composition is combining multiple data services to provide for a user's query a new service which uses data from multiple service providers that are incorporated in the composition. In this situation, the data privacy and especially of the service providers can be breached when their critical data can be seen by another party. Therefore, keeping the data privacy during the composition process is crucial by every work in the context of the service composition. Recent approaches rely on a central mediator that can be trusted or not to ensuring the privacy of the service providers during the query execution. The most recent approaches found problems in case of untrusted mediator where they enforce restrictions like k-protection that can affect the efficiency of their works. Therefore, we propose PrSChain which preserves the privacy of all service providers during service composition and execution using BlockChain technology. We used a permissioned BlockChain that acts as trusted mediator where it enables users to access to the BC if a valid certificate is given. We use Hyperledger Fabric to implement our solution where it stores sensitive data about the composition plan. In addition, the intermediate query results are saved in IPFS that acts as offchain storage. As a proof of concept, we have tested PrSChain on a real-world medical dataset to show its feasibility and efficiency for maintaining privacy in a secure and trusted manner.

Povzetek: Razvita je nova metoda kompozicije podatkovnih storitev, ki ohranja zasebnost podatkov s tehnologijo BlockChain.

1 Introduction

Protecting data privacy represents a big challenge for researchers [1,2,3,4,5] in the security and privacy fields, especially with a large amount of data diffused using internet services. In Service Oriented Architecture (SOA), providing a service for user queries usually requires sharing data from multiple data service providers during service composition [6]. Traditionally, for keeping the data privacy, the composition process is not done directly between service providers but it requires a third party called mediator [7, 8] that rewrites the query, creates and execute the composition plan by selecting the most suitable services to be composed to achieve the desired result. The mediator carries out all these functionalities that could be trusted or not. Selecting a non-trustable mediator may lead to the disclosure of critical information [9], which is the primary concern of the recent works such as [26], [9] and [10].

Several approaches aim to secure data service composition such as differential privacy [11], secret sharing [12], and k-anonymity [13] , among others, have been proposed to deal with data leakage and preserve privacy during service composition. Most of these works [9,10] abandon mediators to provide privacy. Recent works have adopted the k-anonymity solution, which involves k individuals to release data. K-anonymity trades between using a big k to ensure privacy and the time of releasing data. Therefore, it affects system scalability [14]. Recent works like [10] use an extension of k-anonymity called k-protection which has the same issues as the former to ensure scalability.

The most recent works such as [9] and [10] support sharing the service composition plan with all service providers but this strategy enforces these works to incorporate additional techniques like k-protection to keep the data privacy of the service providers. Also, in case of distrust mediator, the recent works encrypt the exchanged data between the participants in order to hide it from the mediator.

In recent years, Blockchain technology proof its efficiency to keep data privacy and ensure the trust and security. Especially, when a permissioned BC is incorporated like Fabric Hyperledger where every participant must have a valid certificate to grant access permission. This technology is used in more than just cryptocurrencies, such as Bitcoin. It is also widely used to preserve privacy in different fields, such as healthcare and IOT [16, 17], data service sharing [37], network countermeasure selection [38], as well smart cities [34].

The present work invests in Blockchain (BC) [15] technology to ensure trust security while preserving the data privacy of the Service Providers (SPs) during service composition. We believe that BC can give solutions to problems that have been found by recent works.

This paper proposes PrSChain which proposes a novel solution that presents a method of integrating permissioned blockchain in data service composition. It contributes by building a privacy-preserving system based on Hyperledger Fabric permissioned BC [19] for data service composition. The system omits the use of a mediator as a third party. This approach invests in the decentralization of BCs to generate, create and execute the composition plan using smart contracts without any central mediator.

We have summarized the contributions of our paper as follows:

- ✓ Building a decentralized privacy preserving system in service composition process based on Blockchain which is a new, trust and secure technology that does not require any third party to prevent the harmful effect that is associated to the central entity.
- ✓ Use a permissioned BC that enables the service composition when it encompasses the process of authentication, authorization, access control, operation logs, identity management, Plan generation and execution. All of these processes that early mentioned are implemented using smart contracts.
- ✓ Replace the mediator with a coordinator entity that has only limited responsibilities. It has no information about the service providers and the real composition plan which is generated by BC peers.
- ✓ We propose that the generated composition plan must be hidden from the service providers in order to prevent using additional techniques like k-protection.
- ✓ Using IPFS as offchain storage in order to prevent storing the intermediate exchanged data on the BC to ensure system scalability. The IPFS data access is given only to legal participants from BC system.

This paper is structured as follows: section 2 describes the background and related works where we give definitions about blockchain and service composition along with approaches about keeping the privacy with and without blockchain. Section 3 describes our proposal to keep the privacy of the data service providers and secure the service composition process using blockchain where we describe our high-level architecture and the main steps to generate and execute the composition plan under privacy preserving. Section 4 presents the implementation of our proposal. The evaluation results are depicted in section 5 where the discussion part is given by section 6. Section 7 concludes the paper with future directions.

1 Background and related works

1.1 Blockchain

The idea of BC was invented by [20]. However, the term BC was invented by [15], when he used BCs as an immutable ledger to register transactions of the cryptocurrency Bitcoin. The database is distributed as a distributed ledger between a group of nodes. They create and append new blocks of data after a consensus between them. A block contains a bunch of chronologically ordered transactions, their Merkle tree and the hash of the previous block. The data is stored in the block body, and the entire block is hashed and stored as a block header. Each block contains the hash of the previous block, which creates interconnected blocks. BCs are classified into different types depending on requirements and access control such as private and public ones .
New BC domains need a large amount of data; therefore, two important notions are born which are the onchain and offchain. The first one contains critical application data such as access permissions and data hash for further integrity verification. The second one stores the big data used by the application to prevent unscalable BC processing but these data must be used with the onchain information in order to be useful. Several recent works support merging onchain and offchain storage such that in [18, 22, 35].

1.2 Service composition

A composite service [36] is a service that is composed of two or more services to accomplish a user's query. Service composition has five main steps, as shown in Figure 1 [24]. It starts when a user requests a service and the mediator treats the query and discovers services that could be composed. A matchmaking algorithm is used to select the required services. A plan composition is generated to accomplish the query. Srivastava et al [23] modeled a composition plan as a Direct Acyclic Graph (DAG). Each node corresponds to a service participant, whereas edges represent the dependencies between participants that could be parents or children or both. The execution could be run parallel when there is no dependency between services. Finally, the result is sent to the user requester.

1.3 Related works

In this section, we are to bifurcate the related works into two categories. The first one belongs to privacy in service composition. The second one is the investigation of the Blockchain based service composition.

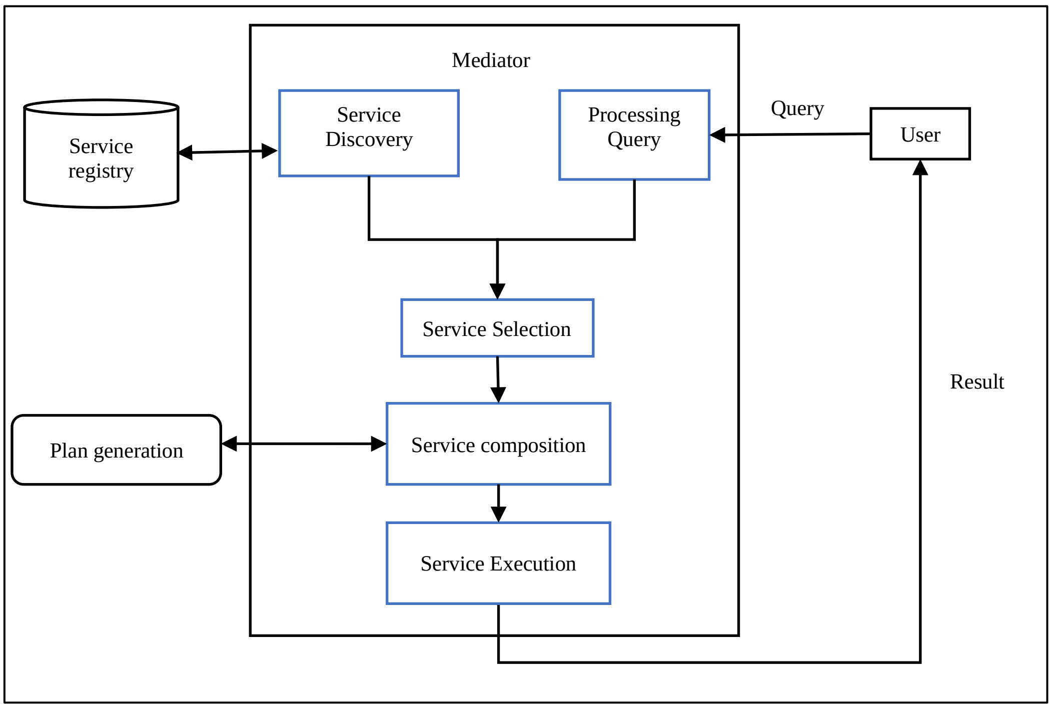

Figure 1: Traditional service composition process [24]

1.3.1 Privacy in service composition

Clarke [25] has seen that the comprehensive interpretation of privacy is about the integrity of the individual and the right to control their data about themselves, and it should be kept from other individuals and organizations. Wherein the privacy in service composition brings the exact definition of protecting sensitive information. Furthermore, herein could infer the critical data, hence we say that the privacy is breached. Thus, to reach the aim of preserving privacy within data services, many methods were put forward.

Many recent studies try to solve the problem of preserving the user's privacy in service composition. It emphasizes that the user's credentials must not reveal to an unauthorized service provider or a mediator as a third party. Bahramgi et al. [9] proposed a practical multi-source data integration approach that would tackle privacy issues raised by the execution of a composition that prevents the leakage between service providers or inferring data about users. They apply the K-protection which is an extension of K-anonymity for protecting critical data between service providers. They used OPES to encrypt numerical data between service providers in order to hide it from the mediator. In the same endeavor by using K-protection, Tiwary et al. [10] proposed a new method for preserving privacy in service composition that relies on a mediator, where the mediator is untrustworthy. They used a secret sharing string to authenticate the service providers between the parent service provider and the child service provider in the composition process. In this approach, the authors put forward that the mediator creates and shares the plan with all the service providers for participating. The selection process is based on the mediator's mapping table. The mediator executed the query with the hashed of the critical data as input. In this approach, the mediator is blind. The authors did not mention how the mediator treats the final results.

On the other hand, Tbahriti et al. [26] put forward the issue of privacy preservation within the service composition by proposing a privacy model that allows a service to define the privacy policy specifying the set of privacy practices applicable to any collected data and a set of privacy requirements specifying the set of privacy conditions that a third party must meet to consume the service's data. They proposed a negotiation mechanism to define a composition plan to reach compatibility of concerned services. The mediator in [26] is trustworthy. It could see the exchanged attributes between the service consumer and the service producer (inputs and outputs).

In a broader sense, all the solutions mentioned above are considered important. However, we point out several issues that belong to privacy leakage. Wherein [9] has mentioned that re-executing the same query from DS1 to DS2 might lead to a privacy breach, through the use of the K-anonymity, the service provider DS2 might infer the critical attribute of DS1. Although [10] proposed using an in-memory table, this later comprised the value of the attribute that belongs to a service provider and the convenient hash value (SHA-256). Owing to the non-scalability of K-anonymity and the distrust of a third party, the mediator creates and generates the composition may not be a good decision because it may tamper with the quality of service, on the one hand, and the exchanged flow of data, on the other hand. We propose to use the BC technology as a trust and decentralized party to handle this issue and to ensure security, privacy and scalability at the same time, in addition to treatment with query re-execute we prefer that child service provider is not aware who is the parent service provider, in this case, the child cannot infer any information about the data owner.

1.3.2 Blockchain based decentralized service composition

Decentralization is an attractive feature of Blockchain, which many researchers in service composition have investigated. The authors [27,28] believe that incorporating Blockchain technology settles centralization problems. In contrast, the decentralization in the service composition will reduce the costs and time, as well as the Blockchain security, scalability, and transparency features. Furthermore, concerning the immutability and traceability features, Yu et al. [27] have put forward how to select the optimal service composition in a transparent and decentralized environment, using particle swarm optimization (PSO) as a consensus to deal with complex tasks in cloud manufacturing. In the same direction, Radmanesh et al. [28] proposed a decentralized platform using blockchain technology for cloud manufacturing service composition, where the miners are rewarded when they propose an optimal service composition solution, as well as they are in charge of building the transactions and blocks by using the proposed consensus proof of optimality. Similar to [28], other works such as [31] and [32] integrate BC technology in cloud manufacturing service composition where the focus is not in data privacy. Due to the diversified internet of things services, Al Ridhawi et al. [29] described the integration of Blockchain in a decentralized cloud solution that uses Software Defined Networks (SDNs), fog computing, and Blockchain to compose complex services without requiring any intermediary. The composition process adopts a reinforcement learning technique to construct secure and reliable paths. Moreover, Blockchain has proved its efficiency in many fields, especially when it ensures trust between the parties of the same framework. Viriyasitavat et al. [30] investigated the Blockchain in an automated business process management to select and compose services in an open business environment, wherein [30] emphasized that using Blockchain will reduce costs in multiple aspects.

Although the merits of integrating Blockchain in service composition ensure security and decentralization, privacy remains a significant concern that should be recognized. As a consequence, the privacy of the user might be leaked.

To the best of our knowledge, we are the first to investigate the Blockchain to preserve privacy for data service providers in service composition. In addition, our proposal ensures the privacy of all sensitive data about the composition and how execute it along with protecting the intermediate exchanged data between service providers. Table 1 gives several comparison points between our work and other recent works that have strong relation with our approach. Our proposal comes to address the main shortcoming mentioned above about the mediator, service providers and service composition. We believe that the recent works such as [9] and [10] did not give full privacy protection even by using K-protection because they support sharing service composition plan with all service providers. We propose a BC based privacy preserving in service composition where the composition process is decentralized. In contrast to previous works where the mediator plays a crucial role, our proposal replaces it with a coordinator which has very low confidence and limited responsibilities. The trust property of the mediator has made problems in the related works where some of them suppose it a trust entity and the others regard it as distrust such as works of [9] and [10]. Thus, our work eliminates the centralized mediator and uses a blockchain network in order to prevent the trust problems. We believe that the service composition plan must not be shared across all service providers because this strategy has some drawbacks where a service provider can associate data to its parents. Since our proposal support permissioned BC, every entity must follow authentication and authorization steps in order to interacts with the BC. Therefore, the access control is assured by the BC where every smart contract invocation is checked if its requester is illegible to use it or not. The intermediate subquery results are not exchanged via coordinator and it is stored in off-chain storage where the access to it is only to legitimate service providers who decrypt the data address.

Table 1: Summary table of the related works

Approach	Tbahriti & al [26]	Bahramgi & al[9]	Tiwary et al [10]	Our Method
BC based	No	No	No	Yes
Decentralization	No	No	No	Yes
Data integrity	No	No	Yes	Yes
Mutual authentication	No	No	Yes	No
Access to intermediate data	Plaintext	Encrypt numerical data	Hash	Encrypt only the pointer
Service providers aware of other service providers in service composition	No	Yes	Yes	No
Privacy between service provider	No	Yes	Yes	Yes
Trust on mediator	Very high	Low	Low	Distrust

2 Proposed approach of blockchain based privacy preserving in service composition

The privacy preserving in service composition process requires maintaining the privacy of all service providers that have participated for accomplishing a user's query. The inclusive idea of the proposed design is that platform relates to service composition that relies on Blockchain which is different most from traditional service composition privacy preserving approach. Thus, we proposed PrSChain (see figure 2) is a secure and privacy preserving platform that aims to execute a user's query which needs data services composition in order to obtain the answer, where the coordinator is distrust entity processes the query and it is in charge of sharing the process result with the permissioned Blockchain. This latter plays a role of a trust system, it adopts the entire functionalities of a traditional and a distrust mediator, mainly generating the composition plan and replies only a limited plan to the coordinator without providing the necessary information (ex. output and input attributes between service providers), where the coordinator in its turn notifies a set of service providers to start execution, thus the information about services provider' sub-query is hidden from the coordinator. Every selected SP requests from the HLF Blockchain system its subquery along with information about input IPFS data and its children public keys. Then, it requests from IPFS all previously results of all its parents in order to execute its subquery and save the results into IPFS. The hash returned by this latter is encrypted with all children public keys and saved into the HLF system. After all service providers have finished the execution, the coordinator gets all final results from the IPFS using requested hash from the HLF system and finally it joins all results and return the final response to the user.

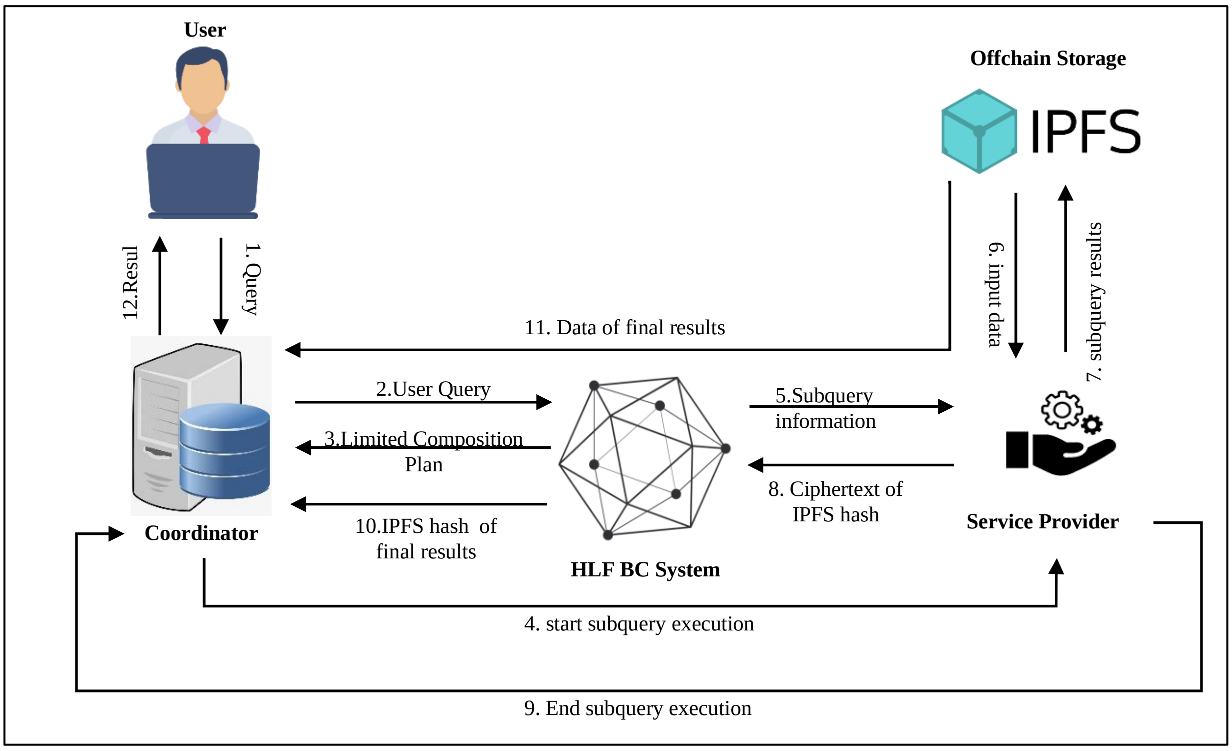
User
Offchain Storage
IPFS
12.Resul
1. Query
6. input data
7. subquery results
11. Data of final results
2.User Query
3.Limited Composition Plan
5.Subquery information
8. Ciphertext of IPFS hash
10.IPFS hash of final results
Coordinator
HLF BC System
Service Provider
4. start subquery execution
9. End subquery execution

Figure 2: PrSChain: High-level system architecture of the proposed protocol for a BC-based preserving privacy of Data Service composition.

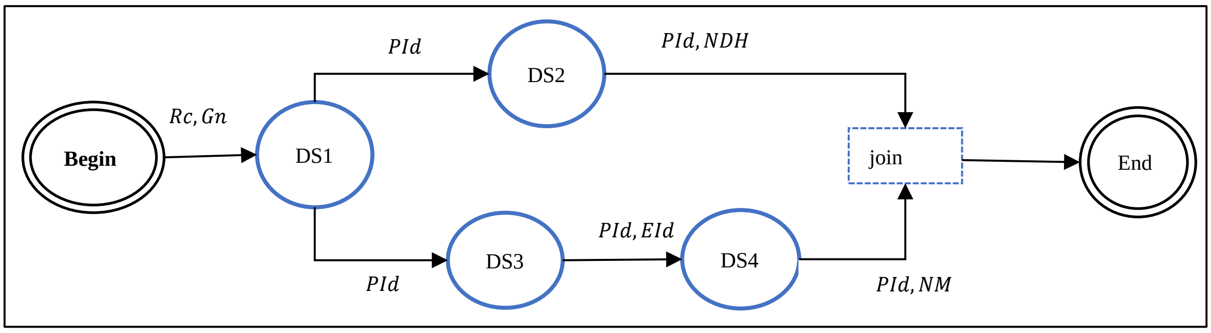

Figure 3: An example of a data service composition.

Example 1: The next example provides a typical service composition based on a real scenario, which will be used along this paper to help readers understand how using the BC to preserve the privacy of involved partners during service composition. In the next sections, we provide examples about the generation of a service composition and how execute it in secure manner and preserve the privacy of the service providers using Blockchain technology.

The example is taken from the medical domain where the data are taken from the original database of diabetes medications [21]. We assume that a user (ex. Doctor) writes the query “What is the number of days in hospital and number of medicaments given to Afro-American male patients that have been supervised by doctors in Cardiology and they have changed the diabetes medication”. We assume that there are four data service providers (DS1, DS2, DS3, & DS4) where every data service DS provides medical records about patients with several medical attributes that given as follows:

- DS1: Patient Identifier (PId), its Gender (Gn) and its Race (Rc).
- DS2: Patient Identifier (PId), Number of days in the hospital (NDH) and an indication if there is a change in his diabetes medication or not (ChD).
- DS3: Patient Identifier (PId), Encounter Identifier (EId) and Medical Specialty (MS).
- DS4: Encounter Identifier (EId) and Number of Taken Medicaments (NM).

We have ensured that a service composition plan can be generated using all service providers. Figure 3 presents the result of the composition plan generation for the previous example where every service provider is responsible for executing a sub-query. A coordinator is in charge of supervising the execution of all subqueries and joins the intermediate results to get the final query result. The sub-queries are executed by the following order:

1. DS1 executes “Select all patient identifiers of Afro-American male patients” .
2. DS2 executes “Select all times in hospital for input patient identifiers that they have changed the diabetes medication”.
3. DS3 executes “Select all encounter identifiers of the input patient identifiers where the medical specialty is Cardiology ”.
4. DS3 executes “Select all number of taken medicaments of the input encounter identifiers”.

All PrSChain steps are given by the figure 4. Algorithm 1 describes the coordinator’s role (steps 1, 3, 4, 10 and 11 in figure 4) in detail in data service composition; and Algorithm 2 describes the part that the service providers play (steps 5, 6, 7, 8 and 9 in figure 4) where Algorithm 3 illustrates the steps followed by blockchain system (step 2 in figure 4). The detailed steps followed by PrSChain components during the Service Composition execution under BC based privacy preserving are described in detail in the following subsections.

2.1 Processing and sharing the query by the coordinator

In this proposed platform, the coordinator is considered a unique entity that manages the interactions between other components under the client/server architecture. In regards to the step 1 in figure 4, among the coordinator responsibilities user‘s query processing , even though it is a distrust entity, it processes the query and it shares the output results (Steps 5 and 6 of Algo 1) with the BC in order to assist the blockchain to generate the composition plan.

Algorithm 1 Coordinator’s Steps of Execution
Input: $user\_query$
Output: $query\_results$

```
1   var $comp\_plan$ is the generated service composition plan without attribute names;
2   var $query\_KG$ is the result of query processing of user query;
3   var $completed\_SP$ is a set contains the service providers that complete the execution of the subqueries;
4   var $final\_results$ is a set that contains query execution results from the service providers that do not have any child;
5   Process $user\_query$ and store it in $query\_KG$;
6   Send $query\_KG$ to the blockchain by invoking specified smart contract;
7   Get the generated composition plan from the BC via smart contract and stores it in $comp\_plan$;
8   Initialize $completed\_SP$ with the service providers (in $comp\_plan$) that do not have any parent;
9   While (the size of $completed\_SP$ does not equal to the number of the SPs in $comp\_plan$) do
10          Take a service provider (Say $S$) from $comp\_plan$;
11          if (all parents of $S$ are in $completed\_SP$) then
12          Send a notification to S to execute its subquery;
13        if (execution completed is received from S) then
14              add S to $completed\_SP$;
15          else
16              abort;
17  Get final results from the blockchain via smart contracts invocation and store them in $final\_results$;
18  Process the final results and store the result in $query\_results$;
19  Return $query\_results$;
```

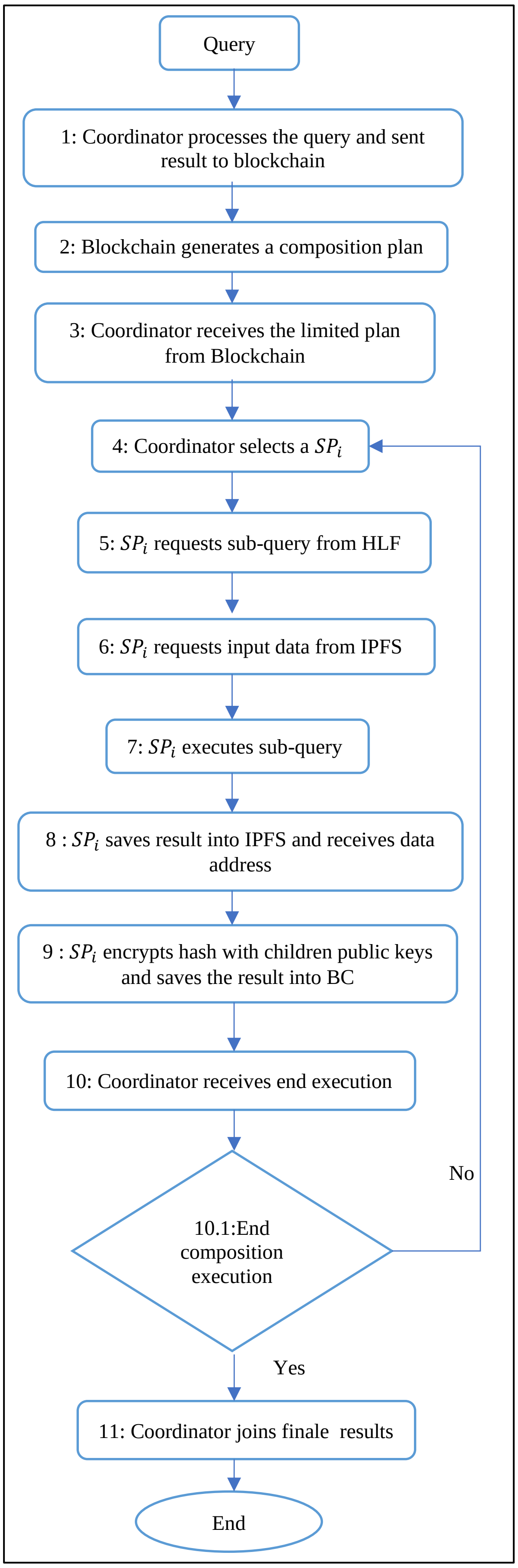
Query
1: Coordinator processes the query and sent result to blockchain
2: Blockchain generates a composition plan
3: Coordinator receives the limited plan from Blockchain
4: Coordinator selects a SP_i
5: SP_i requests sub-query from HLF
6: SP_i requests input data from IPFS
7: SP_i executes sub-query
8 : SP_i saves result into IPFS and receives data address
9 : SP_i encrypts hash with children public keys and saves the result into BC
10: Coordinator receives end execution
10.1:End composition execution
No
Yes
11: Coordinator joins finale results
End

Figure 4: Steps in privacy preserving in service composition

```
Algorithm 2 Service provider's Steps of Execution
Input: $sub\_query$
Output: $sub\_query\_results$
1   We refer the current service provider as $SP$;
2   var $SQ_{SP}$ is the subquery associated with $SP$;
3   var $PK_{SP}$ is the public key associated with $SP$;
4   var $PK\_Ch_{SP}$ is the set of public keys of $SP$ children;
5   var $Addr_{res}$ is the offchain address of the subquery execution result;
6   var $addr\_Pr_{SP}$ is the set of all ciphertexts of the offchain addresses of the query results related to $SP$ parents;
7   var $data\_Res_{SP}$ is the set of all subquery results of $SP$ parents;
8   if (Coordinator asks for executing the associated sub query) then
9       Get $SQ_{SP}$ from the blockchain via smart contracts;
10      Get $PK\_Ch_{SP}$, $SK\_Pr_{SP}$ and $adr\_Pr_{SP}$ from the blockchain via smart contracts;
11      for (address $add$: $addr\_Pr_{SP}$) do
12          Decrypt $add$ by using $PK_{SP}$;
13          Get the data from the offchain using the decryption of $add$;
14          Add the data to $data\_Res_{SP}$;
15      End
16      Execute $SQ_{SP}$ using all data in $data\_Res_{SP}$ and store it in $sub\_query\_results$;
17      Store $sub\_query\_results$ in the offchain and get $Addr_{res}$;
18      for (public key $pk$: $PK\_Ch_{SP}$) do
19          Encrypt $Addr_{res}$ with the $pk$;
20          Save the cyphertext in the blockchain via smart contract;
21      End
22  return $sub\_query\_results$;
```

2.2 Blockchain generates service composition and maintains its integrity

After query processing, the coordinator sends the result to BC, where the peers in their turn execute the smart contract to generate the composition plan [step 2 in figure 4] and [steps 5, 6 and 7 in Algo 3], they use on-chain data about service providers such as their registered services and their input and output attributes. Afterward, the peers apply the selection of optimal plan, in our contribution we have not focused on how select the best services, hence we adopted the idea of [23]. Therefore, the plan generation is executed in decentralized processing without the interfere of the coordinator in order to ensure transparency. Figure 3 presents the result of the composition plan generation for the running example. In the end of the plan generation, the BC store it [step 8 in Algo 3] in order to keep both privacy and integrity.

The process is ongoing and the BC replies an anonymized composition plan [step 3 in figure 4] and [step 9 in Algo 3] to the coordinator in order to orchestrate the execution phase, and because the BC has a copy of the composition plan, the coordinator could not modify it, in addition it is blind and basically it is not aware about service composition critical information like SP subqueries.

Algorithm 3 Blockchain's Steps of Execution
Input: kg_query
Output: $comp\_plan$

```
1   var $comp\_plan$ is the generated service composition plan;
2   var $service\_providers$ is the set of all registered service providers;
3   var $sub\_queries$ is the set of all generated subqueries;
4   if (Coordinator send a user query) then
5       Store the query in $query$;
6       Generate $comp\_plan$ using $query$ and all data provided by $service\_providers$;
7       Generate $sub\_queries$ using $comp\_plan$;
8       Store $comp\_plan$ and $sub\_queries$ in the blockchain using smart contracts;
9       Send $comp\_plan$ to Coordinator without the attribute names;
10  if (a service provider $SP$ asks for its subquery $SQ_{SP}$) then
11      Query the blockchain for $SQ_{SP}$ using smart contracts;
12      Send the subquery $SQ_{SP}$ to $SP$;
13  if (a service provider $SP$ asks for $PK\_Ch_{SP}$ the set of public keys of all its children) then
14      Query the world state for $PK\_Ch_{SP}$ using smart contracts;
15      Send $PK\_Ch_{SP}$ to $SP$;
16  if (a service provider $SP$ asks for $addr\_Pr_{SP}$ the set of ciphertexts of the results of its parents) then
17      Query the world state for $addr\_Pr_{SP}$ using smart contracts;
18      Send $addr\_Pr_{SP}$ to $SP$;
19  if (a cyphertext of the address of the result of a service provider $SP$ is received) then
20      Store the cyphertext in the blockchain using smart contracts;
21  return $comp\_plan$;
```

Example 2. Figure 5 illustrates an anonymized service composition plan derived from figure 3 which has presented the service composition related to the example 1. It contains only necessary information to help the coordinator to organize the plan execution and ensure authentication between coordinator and SPs.

2.3 Service composition execution by the coordinator

After receiving the anonymized composition plan from the BC (step 7 in Algo1), the coordinator firstly notifies all service providers (step 4 in figure 4) that act only as parents to start the subquery execution (step 8 in Algo1). Secondly, after receiving notifications about ending subqueries execution, the coordinator notifies only service providers that their parents have executed their subqueries (step 11 and 12 in Algo1). The process is continued until finishing all subqueries execution. In the end, the coordinator joins all results to get the finale results and return it to the requester (steps 17, 18 and 19 in Algo 1). In contracts to traditional mediator, the coordinator has limited responsibilities and it only organizes the subqueries execution without knowing its content and its results. It sees only the finale results without any correspondence to service providers. Therefore, the service provider privacy is protected from the distrusted coordinator.

2.4 Service provider executes sub-query while maintain privacy

The main goal of every service provider is executing its subquery and preserving data privacy. In contracts to recent works that rely on K-protection to protect SPs' data privacy via mediator, our work uses BC as trust mediator along with the next important restrictions:

- Service provider does not have any information about the service composition.
- Service providers do not know each other's where the authentication between them is guaranteed by BC.
- Every SP has a restrict access only to its subquery and data needed to execute it.

Therefore, by using the latter restrictions, the K-protection is not required where each SP cannot guess who owns any data incorporated during subquery execution.

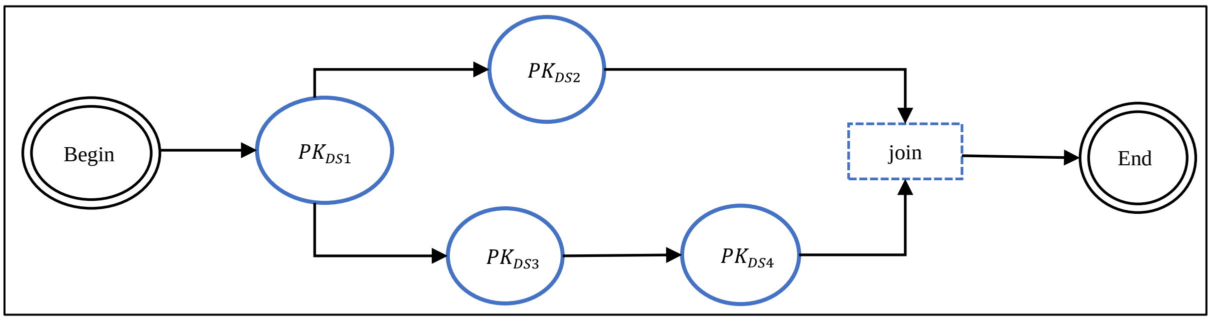

Figure 5: Anonymized service composition plan.

In the next, we provide detailed descriptions of steps followed by every SP in order to accomplish its participation during executing the service composition:

1. **Service provider requests subquery from BC (step 5 in fig4)**: After receiving notification from the coordinator to start execution (Step 8 of Algo 2), it requests its subquery and all its related data from the BC (Steps 9 and 10 of Algo 2) to get important information related to execute it such as:
   - Query content which contains input and output attributes (Steps 10, 11 and 12 of Algo 3).
   - Its children (SPs that are connected to it with input attributes) public keys (Steps 13, 14 and 15 of Algo 3).
   - Ciphertexts of IPFS addresses of data inputs related to the subquery (Steps 16, 17 and 18 of Algo 3).
2. **Service provider gets subquery input data from IPFS (step 6 in fig4) :** If the service provider has parents in the service composition, then its subquery needs data from SP parents in order to complete execution. The SP decrypts all IPFS addresses using its secret key (Step 12 of Algo 2) . After that, it requests data directly from IPFS (Steps 13 and 14 of Algo 2).
3. **Service provider executes subquery and stores results on IPFS (Steps 7 and 8 in fig4) :** It uses all decrypted parent data in order to execute the subquery and stores the results on IPFS and gets a data address (Steps 16 and 17 of Algo2).
4. **Service provider encrypts IPFS address and saves it on Blockchain (step 9 in fig4) :** The result address must be encrypted for protecting its privacy and allowing access for only SP children. Therefore, the SP uses every child public key to encrypt the result address, and the ciphertext is saved on BC to allow SP's children to access the desire result i.e., data of their parent (Steps 18, 19 and 20 of Algo 2, Steps 19 and 20 of Algo 3) .

Example 3: After the service composition generation given by figure 3, the BC creates a query plan which is given by figure 6 where each service provider is assigned to a subquery. As it mentioned before, to ensure SP privacy, the query plan is hidden from both the coordinator and all SPs in order to hide any critical data about the SPs' sub-queries and their results. Figure 6 presents how the BC stores the query plan where every subquery information contains the followings:

- ✓ The subquery input and output attributes.
- ✓ The SP' ID which will execute the subquery.
- ✓ The unique subquery ID which will be shared only with its SP.
- ✓ Children' public keys.
- ✓ Ciphertexts of IPFS addresses of the subquery' result.
- ✓ The hash of the subquery result in order to further integrity checking.

For example, information about subquery $Q1$ contains : SP1 as its service provider, Rc (race) and Gn (gender) as its inputs, Pid (patient identifier) as its output, PK_{SP2} and PK_{SP3} as SP1 children public keys, $PK_{SP2}(addr1)$ and $PK_{SP3}(addr1)$ as its ciphertexts of the IPFS address of its result.

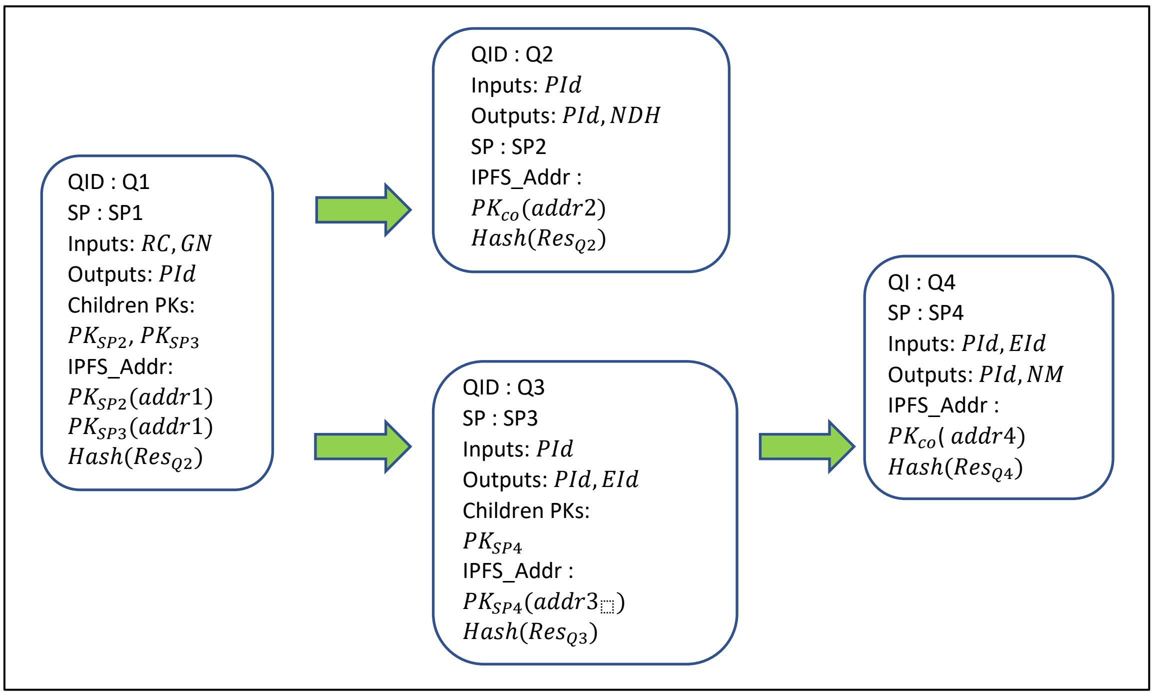

Figure 6: An example of a query plan created from the service composition given by figure 3

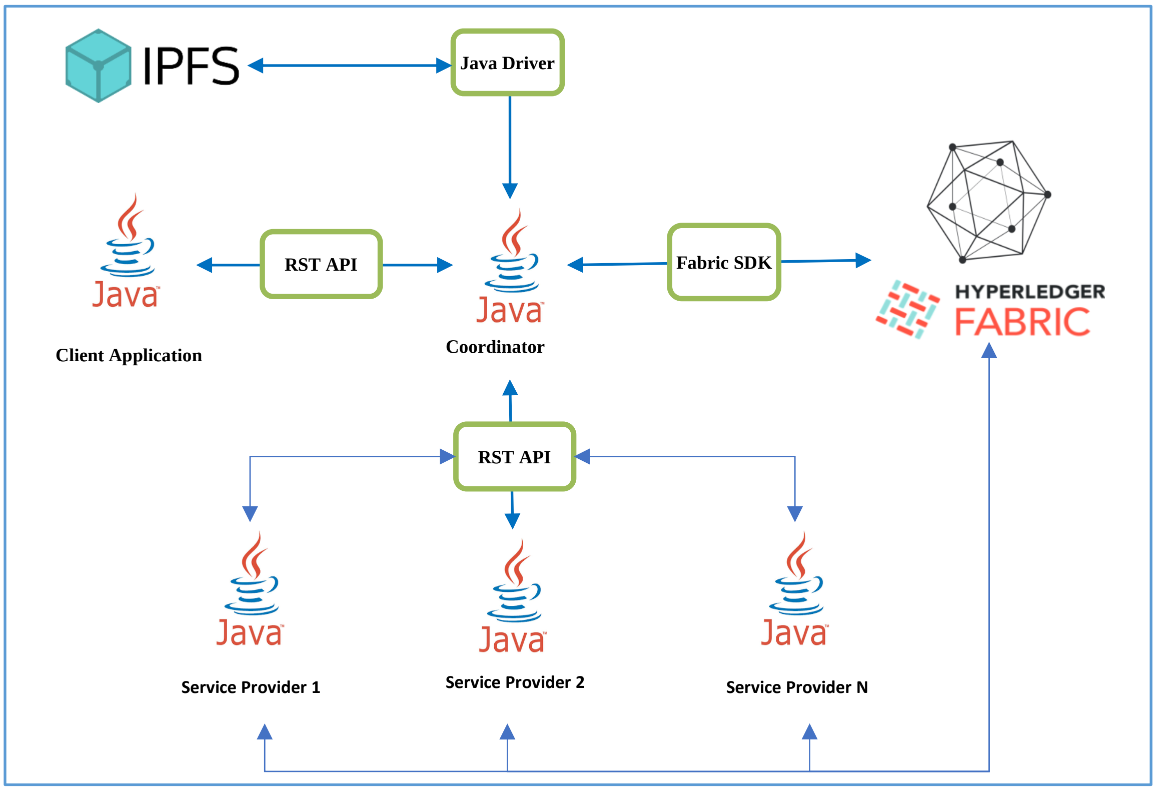

Figure 7: Implementation architecture of PrSChain.

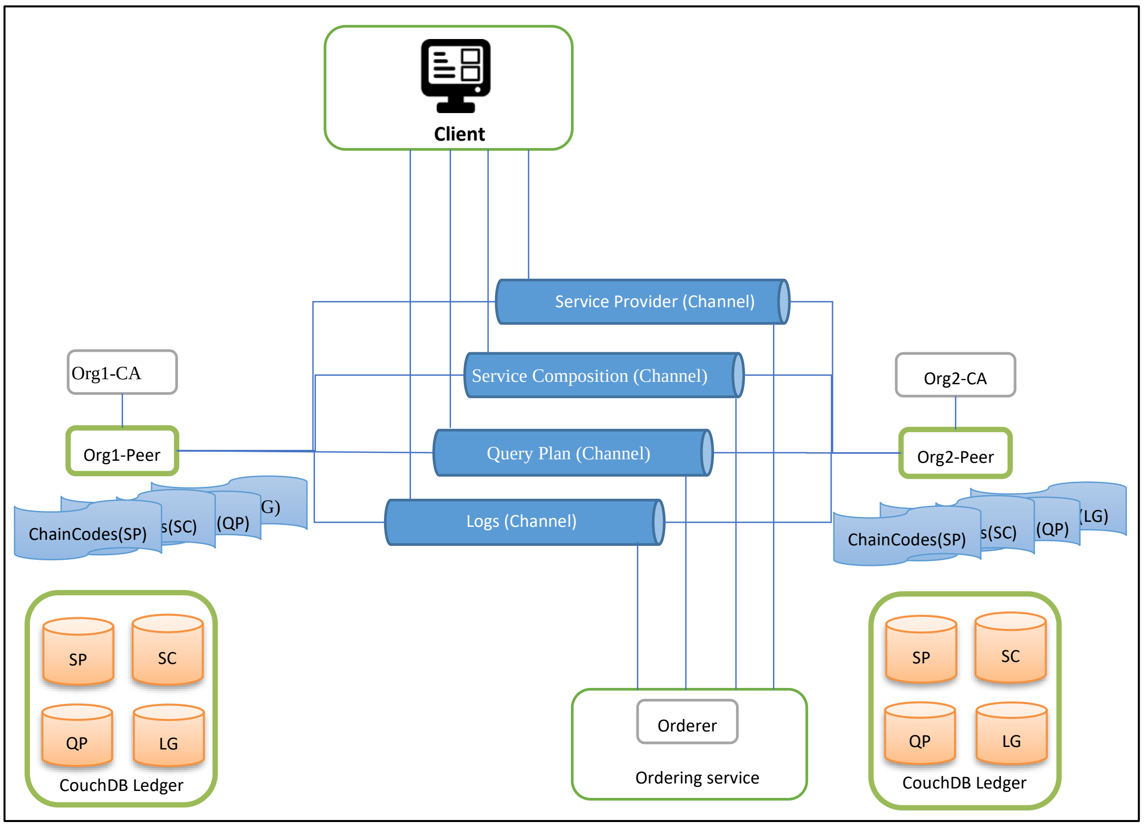

Figure 8: The hyperledger fabric network used by PrSChain.

3 Implementation

The proposed approach is implemented under Eclipse using various Java APIs such as JSON, and Fabric SDK. The architecture of the framework is illustrated in Figure 7, where the main components are:

- ✓ The Fabric Hyperledger Blockchain: is used as permissioned blockchain that allows only legitimate service providers having valid certificates to interact with PrSChain. One of the most important features of fabric Hyperledger is using certificate authorities to generate and control member certificates. The network is configured for two organizations, one peer node for each one. The fabric network uses CouchDB as a world state database and one ordering service. It was built with one certificate authority for each organization. Four channels are created for service providers, service composition, query plan, and logs named respectively "Service Providers", "Service Composition", "Query Plan", and "Logs". Four Fabric smart contracts are deployed using the Go language (one for each channel). The Hyperledger Fabric network used by the proposed method is given in Figure 8, where every channel is associated with its ledger and smart contract.
- ✓ Interplanetary File System (IPFS) https://ipfs.io/ : is a distributed file storage, where every file added to IPFS is given a unique address derived from a hash of the file's content. Our proposal uses IPFS as offchain storage in order to save temporary data about execution of subqueries. Our main goal is ensuring scalability of the blockchain ledger where we keep only sensitive data such as service composition and its subqueries.

3.1 Fabric chaincodes and distributed ledgers

Every peer in HLF has its local database (ledger), which contains all transactions executed by the network via HLF chaincodes. Thus, each peer can have several installed chaincodes for one HLF channel. The distributed ledgers in HLF are updated using smart contracts in demand by external blockchain users. Our work proposes using four distributed ledgers, each associated with one smart contract and several peers. These ledgers store critical data about the implementation components, such as service providers, service composition, sub-queries, and operation logs.

3.1.1 Service provider chaincode

The SP chaincode defines functions that are executed by HLF peers for managing the service providers that are registered by the BC. This chaincode is installed on a channel identified by the same name, "Service Provider", and is associated with a local ledger that saves information about service providers. The SP chaincode uses the Golang structure, illustrated by Listing 1 where some GO functions are given by table 2 along with restricted access.

```
type ServiceProvider struct {
      SpID string `json:"SpId"`
      SpName string `json:"SpName"`
      SpAddress string `json:"SpAddress"`
      SpServices [] ServiceList `json:"SpServices"` }
type ServiceList struct {
     ServiceID string `json:"ServiceID"`
     ServicePK string `json:"ServicePK"`
     Available string `json:"Available"`
     InputAttributes [] AttributesType `json:"InputAttributes"`
   OutputAttributes[] AttributesType `json:"OutputAttributes"` }
type Attributes Type struct {
      AttrName string `json:"AttrName"`
      AttrType string `json:"AttrType"` }
```

Listing 1: The golang structure used by the service provider chaincode

3.1.2 Query plan chaincode

The QP chaincode defines functions that executed by HLF peers for managing the query plan generated from service composition. This chaincode is installed on a channel identified by the same name "Query Plan". The QP chaincode uses the Golang structure which is illustrated by the *listing 2* where some GO functions are given by table 3 along with restricted access.

Table 2: Some smart contract functions that are implemented by the service provider chaincode

Function	Description	Restricted Access
CreateServicePr	Create new service provider using the description given by the invocation parameters	Administrator
GetMatchServices	Get a list of services that match the query attributes given by the invocation.	GenQueryPlan chaincode
updateSP	Updating existed service provider with new information.	Service Provider

```
type QueryPlan struct {
      QueryID string `json:"QueryID"`
      SpID string `json:"SpId"`
      ServiceID string `json:"ServiceID"`
      ServicePK string `json:"ServicePK"`
      SpAddress string `json:"SpAddress"`
      ChildrenPKs [] string `json:"ChildrenPKs"`
      Address_Encryption [] AddressEncryptionType
                    `json:"Address_Encryption"`
      InputValues [] InputValuesType `json:"InputValues"`
      HashResult string `json:"HashResult"` }
type AddressEncryptionType struct {
      Child_PK string `json:"Child_PK"`
      Address_Enc string `json:"Address_Enc"` }
type InputValuesType struct {
      Attribute string `json:"Attribute"`
      Values [] string `json:"Values"` }
```

Listing 2: The golang structure used by the query plan chaincode

Table 3: Some smart contract functions that are implemented by the query plan chaincode

Function	Description	Restricted Access
GenQueryPlan	Generate the query plan that contains all the sub-queries associated to SPs.	Coordinator
SetAddrEnc_Hash	Save the cipher texts (using children PKs) of the address of the sub query results on the BC along with the hash of the result.	Service provider
GetQueryPlan	Get the sub query associated with SP which invokes this function.	Service provider
GetAddressEnc	Get all cipher texts created by all parents of the SP which invokes this function.	Service provider

3.1.3 Service composition chaincode

It manages the anonymized service composition and it is installed on a channel identified by the same name “Service Composition” where it is associated with a local ledger that saves information about the private service composition. The SC chaincode uses the Golang structure which is illustrated by the listing 3 where some GO functions are given by table 4 along with restricted access.

```
type ServiceComposition struct {
    SpID string `json:"SpId"`
    SpAddress string `json:"SpAddress"`
    ServiceID string `json:"ServiceID"`
    ChildrenPKs [] string `json:"ChildrenPKs"`
    ServicePK string `json:"ServicePK"`
}
```

Listing 3: The golang structure used by the service composition chaincode

Table 4: Some smart contracts functions that are implemented by the service composition chaincode

Function	Description	Restricted Access
GetPrivateSC	Get the private service composition which is generated from the query plan.	Coordinator

4 Evaluation

4.1 Experiment configuration

To validate the functionality and test the performance of our approach, a number of experiments have been performed. Experiments are performed on a machine with an Intel Core i7 processor running with a 1.8 GHz clock speed, 16 GB memory, 128 GB SSD and 1 TB for storage. In regards to the implementation architecture, the coordinator is implemented as JAVA REST application that uses the Tomcat 9 as a resource server. All service providers and the client are depicted as JAVA applications that communicate with coordinator using REST API. The SPs and the coordinator interacted directly with IPFS to get sub-queries results. They also have interactions with Fabric network using Fabric SDK. Our implementation uses several Java API in different processes such as: IPFS API, Fabric SDK…etc.

4.2 Dataset

The proposed work is evaluated using the dataset [21] which contains 101767 records, where every patient record has several attributes. Some of them are described by the running example in the first section. Our strategy consists of sampling the whole data set into four subsets, each using different attributes in the condition that two or more subsets share some attributes. Each service provider is associated with one subset to create the service composition given by Figure 3, where the maximum subset size is 101767 records. In the first experiment, we start with 20k records and we add 20k in each experiment until to reach the whole dataset.
The experimental data is available in UCI Machine Learning Repository at https://archive.ics.uci.edu/ml/datasets/Diabetes%20130-US%20hospitals%20for%20years%201999-2008.

4.3 Experiment results

To check the performance of our work, the query given by the example 1 is executed in each experiment. We record the execution time and memory consumption in each experiment.

Figures 9, 10, 11, and 12 show the execution times (in milliseconds) associated with SP1, SP2, SP3, and SP4 during executing the subqueries, and interacting with the Fabric network.

The sub-query execution times are related to sub-query content, IPFS, and Fabric network interactions where the overall value is their sum. The subquery execution dramatically impacts the overall sub-query execution time. This is due to sub-query content, such as the number of variables, constants, and filters. We can observe that SP1 and SP4 take less time than SP2 and SP3 after the execution on 12k records because, in return to the service composition, we found that SP1 and SP4 both have only two specified values as inputs, which act as filters that can reduce the subquery search space. In addition, the number of SP parents can also make a difference during the execution because the SP collects all input data from its parents and process them to achieve the sub-query execution.

The performance evaluation related to the BC communication presents values close to each other, with few differences due to the internal execution of the Fabric network. There are two interaction types between SPs and Fabric networks: request the sub-queries and save ciphertexts of IPFS addresses along with hash results. The SPs always execute the former, which is related to the number of SP children.

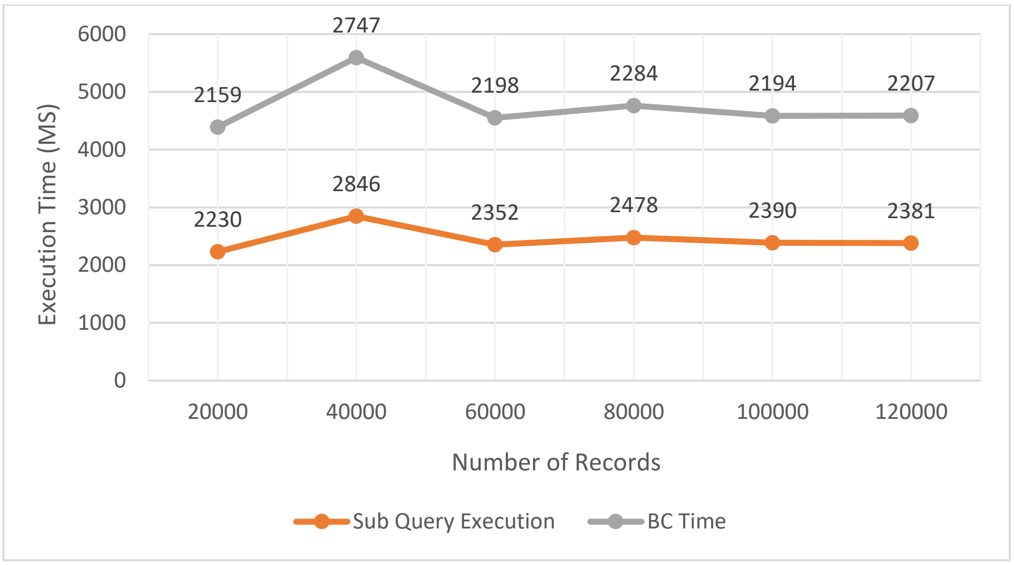

Figure 9: Performance evaluation for service provider 1 (elapsed times vs number of records)

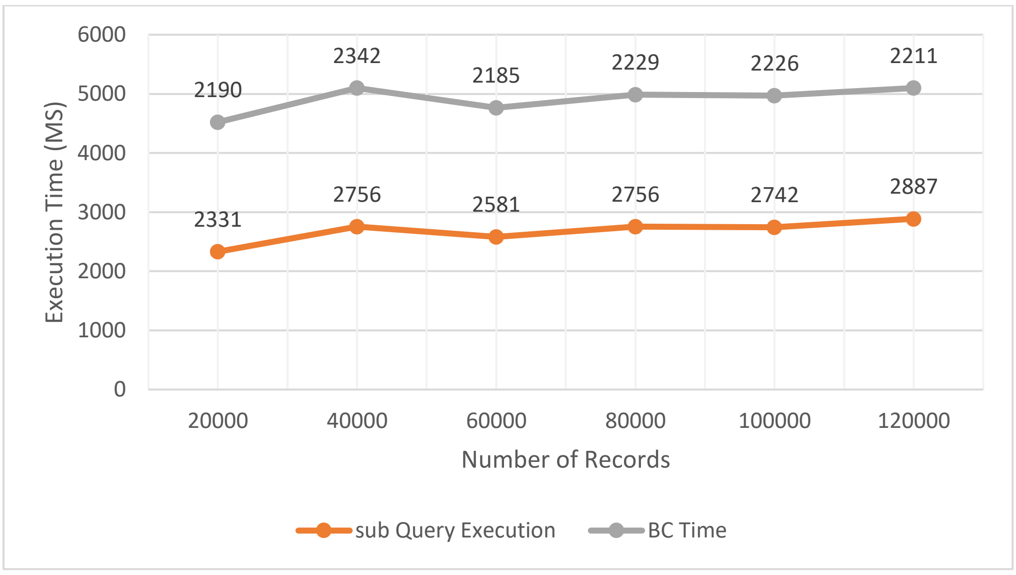

Figure 10: performance evaluation for service provider 2 (elapsed times vs number of records)

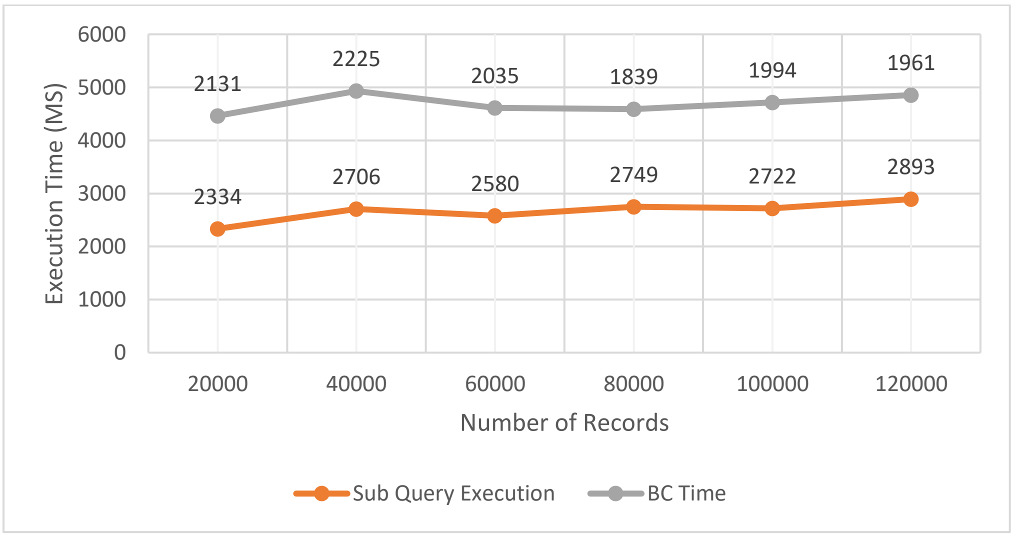

Figure 11: Performance evaluation for service provider 3 (elapsed times vs number of records)

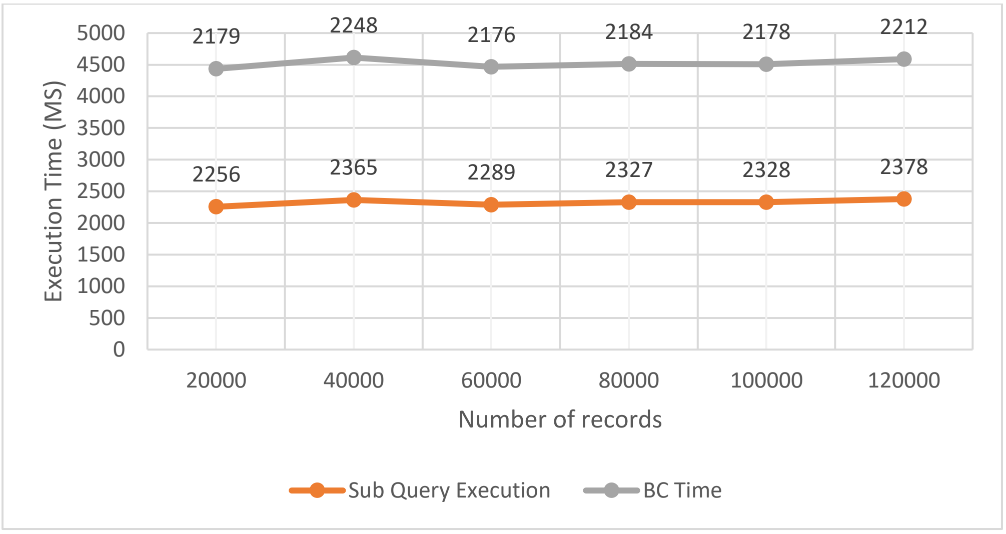

Figure 12: Performance evaluation for service provider 4 (elapsed times vs number of records)

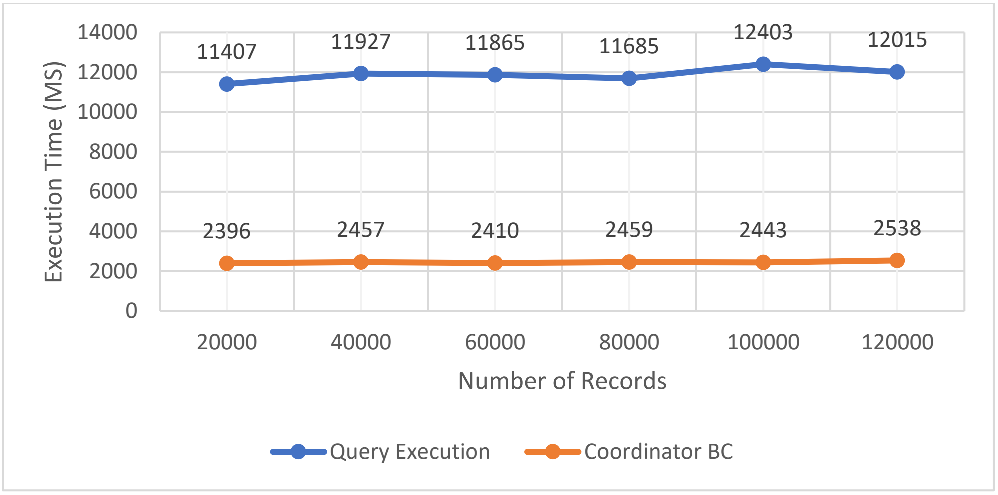

Figure 13: Performance evaluation for the query execution and the coordinator (elapsed times vs number of records)

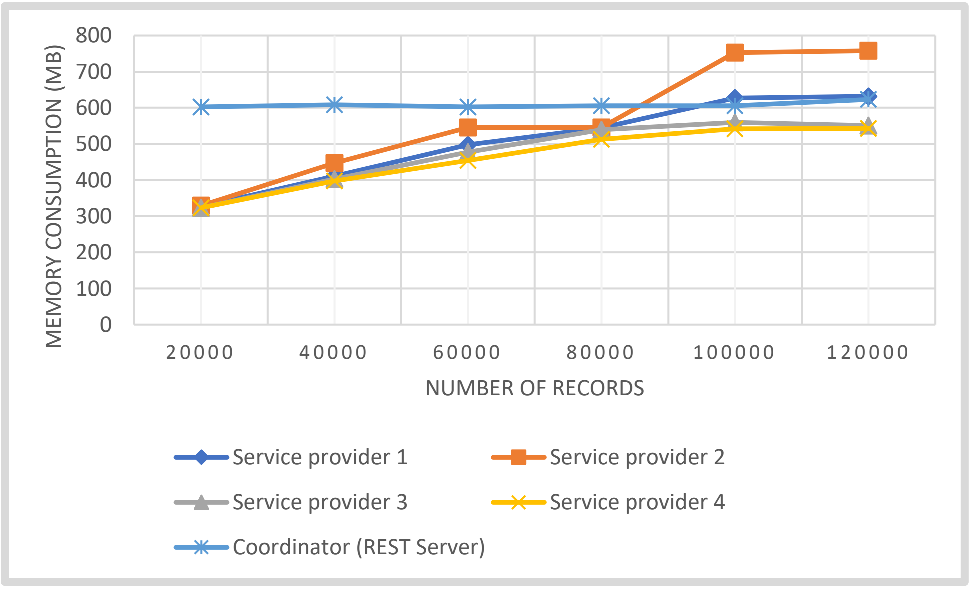

Figure 14: Memory consumption evaluations for the service providers and the coordinator (memory used vs number of records)

Figure 13 shows the overall execution time of the query by all SPs, where there are few changes between experiments. We can conclude that the data size has little effect on the overall execution. This latter can be affected by the number of SPs that participated in the service composition and specifically with its structure; thus, the optimization selection techniques play a centric role. We can observe that the sum of all sub-query execution times is greater than the overall value, and this is due to the parallel execution of SP2 and SP3 because these SPs are independent of each other.

Figure 13 also shows the recorded times of the interactions between the coordinator and the Fabric network. These values are exceptionally close because they are related only to Fabric's internal communications. The overall coordinator execution time is not presented here, where the only factor is the coordinator's final merging of all results. Big-size results can increase the overall time.

For an accurate evaluation, memory consumption of the coordinator and all SPs are recorded and presented in Figure 14. In contracts to the evaluation conducted by [10], who have presented the overall consumption of the execution, our strategy consists of evaluating each application alone because, in reality, all components are distributed across the internet. We can observe that the memory space used remained the same for the coordinator, about 600 megabytes. This is because the coordinator does not generate the query plan or execute the subqueries but assists the query plan execution and merges the final results, which consumes less memory. In contracts, the memory space used for each SP is related to the data size because this latter is managed locally without a remote resource server. In contrast, all sub-queries execution results are saved remotely in IPFS.

5 Discussion

In this section, firstly we provide the security analysis related to our work and secondly, we make a comparison between PrSChain and related works in the literature.

5.1 Security analysis

This section demonstrates the efficiency of our method to deal with the security and the privacy of the service composition in decentralized environment, we put forward the threats that could possibly face the proposed method and how we can resolve them.

The decentralization feature of BC has ensured that an adversary would not tamper the distributed ledger, hence the BC has endeavored to tackle several attacks, where the use of a permissioned BC encounters the privacy of the participant parties by affecting each entity with a public key as a pseudo-anonymity, as a result no disclosure of the real identity on one hand, in addition to protect the data from leakage we have used a shared offchain address among parent-children services, which means only the real child who could get the data from the off-chain storage, and, furthermore, we save the hash of the data on the BC to guarantee the integrity, on the other hand. Another merit the use of permission layer that manages the access control that allows only the member who is authorized to access the distributed ledger. We highlight some assumptions that may threaten our method and how they are thwarted while we use the BC:

- ✓ **Assumption 1**: The coordinator attempts to change the service composition or the query plan by adding new SPs or changing SPs positions.
- ✓ **Solution**: The coordinator has no chance to make any modification because all data about service composition are stored in the Blockchain and they have never been altered where their integrity is assured.

- ✓ **Assumption 2:** A BC peer attempts to change the service composition or the query plan by adding illegitimate SP.
- ✓ **Solution:** It can make changes and store it in its local database but during the chaincode invocation the BC return only the correct service composition because Hyperledger Fabric uses BFT algorithm to mitigate such attack type. Generally, all BC networks assure the data integrity where in the case of Fabric, there are authorities that control the identities and access policies.

- ✓ **Assumption 3**: A service provider attempts to add parents or children to the service composition using its sub-query plan.
- ✓ **Solution**: Firstly, it cannot add parents because before it takes its execution turn, all its parents have been completed execution. Secondly, it cannot add any child because it permitted only to invoke chaincodes that save the cipher texts of its IPFS addresses encrypted by public keys of its real children.

- ✓ **Assumption 4:** A SP tries to know the owner of the input data in order to break the privacy condition. Thus, the SP privacy can be breached.
- ✓ **Solution:** It cannot be able to make such association because our strategy ensured that each SP does not know any SP in the composition.

- ✓ **Assumption 5:** In our work, the coordinator is in charge of merging the final results and hence it is able to associate the results with SPs and then it can breach every SP privacy.
- ✓ **Solution:** It cannot get such association because it has only the private service composition with anonymized nodes and therefore it can guess the relation between the results and the final nodes with probability degrees.

- ✓ **Assumption 6:** The blockchain peers can get the ciphertexts of the IPFS addresses related to the sub-queries results and they attempt to access result data or modify their content.
- ✓ **Solution:** The ciphertexts cannot be decrypted without children private keys and hence BC peers

cannot get or modify the IPFS data. In regarding to the blockchain protocols for keeping the data integrity, there is no chance to add an illegitimate pub key to a subquery of a given parent in order to obligate this latter to encrypt the IPFS address using the added public key.

- ✓ **Assumption 7:** If someone decrypts the IPFS address and gets access to the result of a given sub-query in order to modify it and influence all children that use the changed data.
- ✓ **Solution:** Each child can check the data integrity using its hash which has been generated by the parent of this child. If this latter detects any changing then it stops the execution and the service composition will fail.

5.2 Comparison

Compared to PrSChain, the related works have several drawbacks and they lack important features to ensure full security and privacy protection. Some important lacks are illustrated by the following:

- ✓ Most approaches support centralization where a mediator is in charge of service composition generation and query execution. Even the mediator is entrusted, it can use these responsibilities to for example creating query plan that contains only selected service providers.
- ✓ In most works, the service providers know the generated plan where this feature enforce restrictions that can have negative effect on scalability like using K-protection. In this situation, the query returns additional unnecessarily values.
- ✓ All related works do not support access control and operation logs for future audit and verification.
- ✓ Most works support exchanging subquery results via a mediator that can be distrust entity. In this situation, if the data is not encrypted then the mediator can directly breach the data privacy. In case of encrypted data that uses k-protection, the mediator can share the data without k-protection with the child of a service provider.

Our method is compared with [26,9,10] regarding security and privacy requirements. Let us start with Tbahriti et al [26] pointing to a privacy model that verifies the compatibility between privacy policies and privacy requirements services to enhance service composition. This approach is based on a mediator to exchange the intermediate data and does not employ any cryptographic method to encrypt and protect the data. We propose that coordinator is not in charge of exchanging the data but this latter is exchanged via BC. The recent works for preserving privacy in service composition [9] and [10] have used the same strategy to protect critical data from leakage. Bahramgi et al. [9] encrypted the intermediate data via OPES, while Tiwary et al. [10] used the hash instead of encrypting if the data were non-numerical. In the two cases, using the K-anonymity takes a considerable time when the amount of data is significant, and k is too big. On the one hand, for us, we have not needed to use k-protection because we have proposed that the service providers do not know each other where the authentication is guaranteed using BC.

On the other hand, we do not need any third party or central entity to generate and create the composition plan, whereas the BC peers are in charge, and the probability of tampering with the QoS is very low. We have used the data hash to avoid modification and ensure data integrity. Tiwary et al. [10] propose an in-memory table to handle the problem of re-executing the same query. It does not prevent privacy disclosure between service providers, especially when k is too small. A service provider can save in-memory tables of several queries and it can infer the common values to get the real values of its service parent. We tackle this issue when the service providers are unaware of the actual plan where every SP does not know his parents.

We have incorporated the BC to eliminate the trust in third parties. It demonstrates its scalability of computing and the decentralization feature. In addition, if the query covers critical attributes, then the other works like [10] and [26] will fail to retrieve results because the mediator is in charge of merging results, and it shows only noncritical attributes. In contracts, our work can deal with the last issue because the coordinator can see every attribute without knowing its data owner. From the query processing point of view, the recent works give the mediator the responsibility of generating the service composition. At the same time, they were declared as a distrust entity. In this situation, for example, the mediator can prefer to incorporate some SPs rather than others that can enhance the query execution. In contracts, our work uses the BC network, where BC peers perform the service composition and query generation without a central entity.

6 Conclusion

In this article, we propose a platform to enhance service composition privacy. The design maintains the privacy between service providers, the main idea is how to eliminate the trust on third party which is substituted with Blockchain technology for its high confidence and it plays the role of a trust mediator. The method is focusing on how we investigate the performance computation of a permissioned BC for managing, executing and preserving the privacy in service composition. Moreover, the distributed ledgers for keeping the information for both the service providers and their sub-query in order to eliminate the tampering with data service providers and their privacy. We have used a real-world data service for demonstrating the efficiency of our method and its resilience with any type or amount of data. Compared to recent works that take more time and memory consumption, our experiments have improved performance because the usage of BC without k-anonymity which is the source of increasing time and memory usage.

Different stakeholders such as users and service providers can benefit from PrSChain functionalities. For example, data service providers in the medical domain can participate in service composition under BC privacy preserving in order to protect their data. Users like doctors can perform advanced queries that uses several distributed datasets to get information about patients. In case of illegal manipulation, PrSChain can perform audit verification from the log ledger that keeps data about all executed operations. For interoperability, our proposal supports using REST API that ensures the communication between different internet applications. The implementation of PrSChain in the real world situation is related to Fabric Hyperledger and its capabilities to process users' transactions and deal with scalability issues.

For future directions, our approach must be extended to deal with the situation when the requester must participate using his data in the service composition. In this situation, the requester should allow access to his data and put an access control and policies in order to protect his privacy from illegal access. The requester can use our blockchain network for keeping his data privacy and integrity by allowing access control via blockchain. PrSChain can be extended to support rewarding mechanism where a service provider can be rewarded if it participated in several service composition.

Another research direction is to use IOTA [33] which is based on tangle technology that uses directed acyclic graph rather than list to save transactions. The main goal of using IOTA is to ensure rapid transaction without fees. PrSChain can use this technology to ensure fast execution of the service composition plan.

References

[1] Boeckl, K. and Lefkovitz, N. (2020), NIST Privacy Framework: A Tool for Improving Privacy Through Enterprise Risk Management, Version 1.0, Other, National Institute of Standards and Technology, Gaithersburg, MD, [online], (Accessed October 16, 2023).
https://doi.org/10.6028/NIST.CSWP.01162020

[2] J.-W. Lee, H. Kang, Y. Lee, W. Choi, J. Eom, M. Deryabin, E. Lee,J. Lee, D. Yoo, Y.-S. Kim, and J.-S. No, Privacy-preserving machine learning with fully homomorphic encryption for deep neural network, IEEE Access, vol. 10, pp. 3003930054, 2022.
https://doi.org/10.1109/access.2022.3159694

[3] Wang, C., Wang, D., Xu, G. et al. Efficient privacy-preserving user authentication scheme with forward secrecy for industry 4.0. Sci. China Inf. Sci. 65, 112301 (2022).
https://doi.org/10.1007/s11432-020-2975-6

[4] C. Zhang, M. Zhao, L. Zhu, W. Zhang, T. Wu and J. Ni. FRUIT: A Blockchain-Based Efficient and Privacy-Preserving Quality-Aware Incentive Scheme, in IEEE Journal on Selected Areas in Communications, vol. 40, no. 12, pp. 3343-3357.
https://doi.org/10.1109/JSAC.2022.3213341

[5] J. Song, W. Wang, T. R. Gadekallu, J. Cao and Y. Liu, "EPPDA: An Efficient Privacy-Preserving Data Aggregation Federated Learning Scheme," in IEEE Transactions on Network Science and Engineering, https://doi.org/10.1109/TNSE.2022.3153519

[6] Laskey, K. B., & Laskey, K. (2009). Service oriented architecture. WIREs Computational Statistics, 1(1), 101–105. Portico.
https://doi.org/10.1002/wics.8

[7] N. Guermouche, O. Perrin and C. Ringeissen, "A Mediator Based Approach For Services Composition," 2008 Sixth International Conference on Software Engineering Research, Management and Applications, Prague, Czech Republic, 2008, pp.273-280
https://doi.org/10.1109/SERA.2008.43

[8] Zhang, Yu, Xiangmin Zhou, and Yiyue Gao. "Optimizing the data intensive mediator-based web services composition." Asia-Pacific web conference. Springer, Berlin, Heidelberg, 2006.
https://doi.org/10.1007/11610113_62

[9] M. barhamgi, C. Perera, C. -M. Yu, D. Benslimane, D. Camacho and C. Bonnet, "Privacy in Data Service Composition," in IEEE Transactions on Services Computing, vol. 13, no. 4, pp. 639-652, 1 July-Aug. 2020.
https://doi.org/10.1109/TSC.2019.2963309

[10] G. P. Tiwary, E. Stroulia and A. Srivastava, "Improving Privacy in Data Service Composition," in IEEE Access, vol. 9, pp. 95716-95729, 2021.
https://doi.org/10.1109/ACCESS.2021.3094188

[11] Cynthia Dwork. Differential privacy. In Automata, languages and programming, pages 1–12. Springer, 2006.
https://doi.org/10.1007/11787006_1

[12] Chen, J.; Liu, G.; Liu, Y. Lightweight Privacy-preserving Raw Data Publishing Scheme. IEEE Trans. Emerg.Top. Comput. 2020, 1.
https://doi.org/10.1109/tetc.2020.2974183

[13] SWEENEY, Latanya. k-anonymity: A model for protecting privacy. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 2002, vol. 10, no 05, p. 557-570.
https://doi.org/10.1142/s0218488502001648

[14] Zyskind, Guy, and Oz Nathan. "Decentralizing privacy: Using blockchain to protect personal data." 2015 IEEE Security and Privacy Workshops. IEEE, 2015.
https://doi.org/10.1109/spw.2015.27

[15] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. Consulted, 1(2012):28, 2008.

[16] Dwivedi, A., Srivastava, G., Dhar, S., & Singh, R. (2019). A Decentralized Privacy-Preserving Healthcare Blockchain for IoT. Sensors, 19(2), 326.
https://doi.org/10.3390/s19020326

[17] Al Omar, A., Rahman, M.S., Basu, A. and Kiyomoto, S., 2017, December. Medibchain: A blockchain based privacy preserving platform for healthcare data. In International conference on security, privacy and anonymity in computation, communication and storage (pp. 534-543). Springer, Cham.
https://doi.org/10.1007/978-3-319-72395-2_49

[18] Djeddai, A., & Khemaissia, R. (2023). PrivyKG: Security and Privacy Preservation of Knowledge Graphs Using BlockChain Technology. Informatica, 47(5). https://doi.org/10.31449/inf.v47i5.4698

[19] Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., Caro, A.D., Enyeart, D., Ferris, C., Laventman, G., Manevich, Y., Muralidharan, S., Murthy, C., Nguyen, B., Sethi, M., Singh, G., Smith, K.A., Sorniotti, A., Stathakopoulou, C., Vukolic, M., Cocco, S.W., & Yellick, J. (2018). Hyperledger fabric: a distributed operating system for permissioned blockchains. Proceedings of the Thirteenth EuroSys Conference. https://doi.org/10.1145/3190508.3190538

[20] S. Haber, W. S. Stornetta, How to time-stamp a digital document, in: Conference on the Theory and Application of Cryptography, Springer, 1990, pp. 437–455. https://doi.org/10.1007/3-540-38424-3_32

[21] Beata Strack, Jonathan P. DeShazo, Chris Gennings, Juan L. Olmo, Sebastian Ventura, Krzysztof J. Cios, John N. Clore, "Impact of HbA1c Measurement on Hospital Readmission Rates: Analysis of 70,000 Clinical Database Patient Records", BioMed Research International, vol. 2014, Article ID 781670, 11 pages, 2014. Link: https://archive.ics.uci.edu/ml/datasets/Diabetes 130-US hospitals for years 1999-2008. https://doi.org/10.1155/2014/781670

[22] Djeddai, A. (2021, November). KGChain: A Blockchain-Based Approach to Secure the Knowledge Graph Completion. In International Conference on Mining Intelligence and Knowledge Exploration (pp. 218-224). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-031-21517-9_22

[23] Srivastava, U., Munagala, K., Widom, J., & Motwani, R. (2006, September). Query optimization over web services. In Proceedings of the 32nd international conference on Very large data bases (pp. 355-366).

[24] S Sridevi S., Karpagam G. R., Vinoth Kumar B., & Uma Maheswari J. (2021). Investigation on Blockchain Technology for Web Service Composition. International Journal of Web Services Research, 18(1), 70–93. https://doi.org/10.4018/ijwsr.20210101.oa1

[25] Roger Clarke. 2006. What's privacy. In Proceedings of the Australian Law Reform Commission Workshop, Vol. 28.

[26] Tbahriti, S.-E., Ghedira, C., Medjahed, B., & Mrissa, M. (2014). Privacy-Enhanced Web Service Composition. IEEE Transactions on Services Computing, 7(2), 210–222. https://doi.org/10.1109/tsc.2013.18

[27] Chunxia Yu, Luping Zhang, Wenfan Zhao & Sicheng Zhang (2019): A blockchain-based service composition architecture in cloud manufacturing, International Journal of Computer Integrated Manufacturing. https://doi.org/10.1080/0951192X.2019.1571234

[28] Radmanesh, S.-A., Haji, A., & Fatahi Valilai, O. (2021). Blockchain-based cloud manufacturing platforms: A novel idea for service composition in XaaS paradigm. PeerJ Computer Science, 7, e743. Portico. https://doi.org/10.7717/peerj-cs.743

[29] Ridhawi, I. A., Aloqaily, M., Boukerche, A., & Jaraweh, Y. (2020). A Blockchain-Based Decentralized Composition Solution for IoT Services. ICC 2020 - 2020 IEEE International Conference on Communications (ICC). https://doi.org/10.1109/icc40277.2020.9149031

[30] Viriyasitavat, W., Da Xu, L., Bi, Z., & Sapsomboon, A. (2018). Blockchain-based business process management (BPM) framework for service composition in industry 4.0. Journal of Intelligent Manufacturing, 31(7), 1737–1748. https://doi.org/10.1007/s10845-018-1422-y

[31] Aghamohammadzadeh, E., & Fatahi Valilai, O. (2020). A novel cloud manufacturing service composition platform enabled by Blockchain technology. International Journal of Production Research, 58(17), 5280–5298. https://doi.org/10.1080/00207543.2020.1715507

[32] Y Yu, C., Zhang, L., Zhao, W., & Zhang, S. (2019). A blockchain-based service composition architecture in cloud manufacturing. International Journal of Computer Integrated Manufacturing, 33(7), 701–715. https://doi.org/10.1080/0951192x.2019.1571234

[33] IoTA. 2022. IoTA. Retrieved September 14, 2022 from https://www.iota.org/get-started/what-is-iota/.

[34] M Makhdoom, I., Zhou, I., Abolhasan, M., Lipman, J., & Ni, W. (2019). PrivySharing: A Blockchain-based Framework for Integrity and Privacy-preserving Data Sharing in Smart Cities. Proceedings of the 16th International Joint Conference on E-Business and Telecommunications. https://doi.org/10.5220/0007829803630371

[35] Djeddai, A., & Khemaissia, R. (2022). Keeping the Privacy and the Security of the Knowledge Graph Completion Using Blockchain Technology. 2022 4th International Conference on Pattern Analysis and Intelligent Systems (PAIS). https://doi.org/10.1109/pais56586.2022.9946869

[36] Casati, F., Ilnicki, S., Jin, L., Krishnamoorthy, V., & Shan, M.-C. (2013). Adaptive and Dynamic Service Composition in eFlow. Seminal Contributions to Information Systems Engineering, 215–233. https://doi.org/10.1007/978-3-642-36926-1_17

[37] Khemaissia, R., Derdour, M., Djeddai, A., & Ferrag, M. A. (2021). SDGchain: When Service Dependency Graph Meets Blockchain to Enhance Privacy. Proceedings of the 2021 ACM Workshop on Security and Privacy Analytics. https://doi.org/10.1145/3445970.3451157

[38] Khemaissia, R., Derdour, M., Ferrag, M. A., & Djeddai, A. (2021). Network Countermeasure Selection Under Blockchain Based Privacy Preserving. 2021 International Conference on Recent Advances in Mathematics and Informatics (ICRAMI). https://doi.org/10.1109/icrami52622.2021.9585916