PrivyKG: Security and Privacy Preservation of Knowledge Graphs Using BlockChain Technology

Ala Djeddai[1*], Rofaida Khemaissia[2]
[1]Chadli Bendjedid El-Tarf University, B.P 73, El Tarf 36000, Algeria
[2]Laboratory of Mathematics, Informatics and Systems (LAMIS), Echahid Cheikh Larbi Tebessi University, Tebessa 12002, Algeria
Email : a.djeddai@univ-eltarf.dz, khemaissia.rofaida@univ-tebessa.dz

Keywords: knowledge graph, blockchain, hyperledger fabric, knowledge graph integrity, smart contracts, privacy

Received: February 22, 2023

Recently, knowledge graph has proved its benefits in AI applications and especially in knowledge representation and reasoning. Therefore, with the huge amount of published data, knowledge graph privacy, trust and integrity take an important role to protect it from illegal access and modification. In this paper, we propose PrivyKG, a new framework which uses the blockchain technology along with an access control based on it to preserve the knowledge graph privacy, integrity, and ensure security and trust. The proposed work has two essential parts to build a decentralized knowledge graph, where the first one uses off-chain storage for ensuring the blockchain scalability and the second one integrates a permissioned Hyperledger Fabric powered with GO smart contracts. The proposed framework is evaluated with a DBpedia dataset that illustrates the efficiency and the feasibility of our proposed against potential threats.

Povzetek: Predstavljena je metoda decentraliziranih grafov znanja za bločne aplikacije.

1 Introduction

Recently, a massive amount of data is published on the web with the increasing usage of applications and web services. Therefore, processing these data has been the goal study of many researchers, where the main objective is trying to learn more about users and its behaviors. Knowledge graph (KG) which connects real world entities by relations, it plays a central role when the application needs a representation and reasoning about knowledge in its processing. Many artificial intelligence applications profit from KG processing in order to help in recommendations and predicting new knowledge. Dealing with security and privacy of KG is a very important task, in order to help KG applications to run under secure, trust and data privacy preserving.

With the widespread usage of KG in artificial intelligence, its security and privacy remain a big challenge due to its critical content. Especially when personal data takes an important place such as social networks and healthcare fields. Several techniques have proposed to maintain the KG privacy such as anonymization and differential privacy [24] which have been used widely to keep personal data privacy. For instance, the anonymization is based on anonymizing the entities with new undistinguished identifiers, this technique has several extensions like k-anonymity [23] which aims at ensuring that the entities are indistinguishable from at least k-1 other entities. Besides the advantages of the anonymization technique, an attacker may de-anonymize the data content to obtain the real entities and its sensitive information, an example of this attack is given by [22]. Differential privacy approaches perturb the data for hiding the real contents, thus keeping the privacy of the sensitive information. In practice, the perturbation can increase the data size and the computation when extracting the original data, therefore the differential privacy is not always efficient. The novel blockchain technology which is a decentralized, distributed and secure database, has been used to tackle many issues of security, privacy and trust in KG [1, 2]. Blockchain has been recently integrated to protect and keep the privacy of the personal data [3, 4, 5] which can be represented by knowledge graphs. In contrast to privacy preservation approaches, BC ensures data integrity and provides an access control over the data, especially in case of permissioned BC where all clients must have digital certificates to achieve the authentication.

Several approaches have integrated KG to enhance BC data querying and its reasoning capabilities such as [12], [13] and [14] but a few works like [15] have used BC with KG to keep data sharing privacy and security. In regards to keep only the KG privacy, [1] has proposed to merge BC with IPFS to keep only the KG integrity by saving the data hash on the BC without any access control. Among the shortcomings that are noticed by recent works do not give a complete KG privacy protection, and marginalize the access control on its data, which complicates the sharing of its content. In contrast, our focus is only on KG data by proposing PrivyKG that uses BC with an off-chain storage in order to promote privacy and integrity protection where the security is improved by supporting a permissioned BC.

PrivyKG is an extension work of [21] and [26] that propose an approach to secure the KG completion tasks such as link prediction and triple classification by running them in decentralization using Blockchain. Compared to [21] and [26], our work focuses on KG privacy by providing access policies, reward mechanism, audit and several KG querying tasks that execute in coordination with blockchain and under permission from KG owner.

The main contributions of this work are listed below:

- ✓ Using a permissioned Blockchain for keeping the integrity of critical data used to build and rebuild the original KG. Thus, these data are saved in secure and decentralized manner where the access is achieved only using permission and a valid certificate.
- ✓ Improving the PrivyKG scalability by using an off-chain storage in order to save the critical data on the blockchain for avoiding the non-necessary and preventing significant transactions.
- ✓ Similar to [21] and [26], PrivyKG supports KG completion tasks such as link prediction and triple classification where they run in decentralization in a secure manner. PrivyKG ensures the privacy of the embedding vectors of the learned model.
- ✓ Securing and improving the trust of KG management tasks such as updating and deleting triples. These latter are performed using decentralization by the BlockChain peers powered by smart contracts.
- ✓ PrivyKG proposes to preserve the privacy of several KG querying tasks by running under BC technology such as filtering critical attributes from query results, checking the truth degree of a given sub graph…etc.
- ✓ Implementing and evaluating PrivyKG using a DBpedia dataset and Hyperledger Fabric [17] powered by Go smart contracts and cooperated with MangoDB for off-chain storage.

This paper is structured as follows: section 2 presents the background and related works where we give definitions about KG, blockchain and how to merge these two recent technologies. Section 3 presents the notion of decentralized KG and describes how securing and keeping privacy of KG using blockchain and off-chain storage where we give details about the proposed design and its main components, we also describe how building, rebuilding and updating the decentralized KG along with main scenarios. Section 4 presents the implementation of PrivyKG. The evaluation results are depicted in section 5 whereat the discussion part is given by section 6. Section 7 concludes the paper with future directions.

2 Background and related works

2.1 Identity management and access control

The main objective of the identity management is creating and verifying the identities of the entities that can use a service from a particular system. The access control aims at creating and verifying the access to a particular resource or service after successful identity verification. Therefore, these two tasks are seen as authentication and authorization of entities in the system. As mentioned in the work of [20], the authentication methods can use different factors for human and non-human such as knowledge factor, possession factor, inherence factor and context-aware factor. The authorization methods can be classified as: Discretionary Access Control (DAC), Mandatory Access Control (MAC), Role-Based Access Control (RBAC), Attribute-Based Access Control (ABAC), and Capability-Based Access Control (Cap-BAC). Identity and access management can be classified into five models: isolated, centralized, federated, user-centric, and self-sovereign models. In contrast to the four models, the self-sovereign model is based on the user where she can store and manage her digital identity. In our work, IAM is based on knowledge factor in isolated settings because we have used local authentication and authorization, however the critical information about IAM is stored in decentralized ledger in the blockchain. The Capability-Based Access Control is supported by PrivyKG because it uses token-based access control that contains all information about data access and the needed operation. We have chosen a permissioned BC that uses ECC [19] and ECDA rather than others like RSA [18] in order to ensure scalability of the encryption.

2.2 Knowledge graph

Knowledge graph describes real word facts by connecting entities with relations, for example, Algeria location north-Africa where Algeria and north-Africa is respectively a subject and an object entities and location is a relation. Therefore, every knowledge graph is a set of triples, each one is a form of subject-predicate-object. We can define the KG as a couple of (E, R) where E is the set of entities and R is the set of relations that connects the elements in E. The KG is appeared with the Google KG in 2021, after that many KG was created and published like: Freebase [6], DBpedia [7] and WordNet [8].

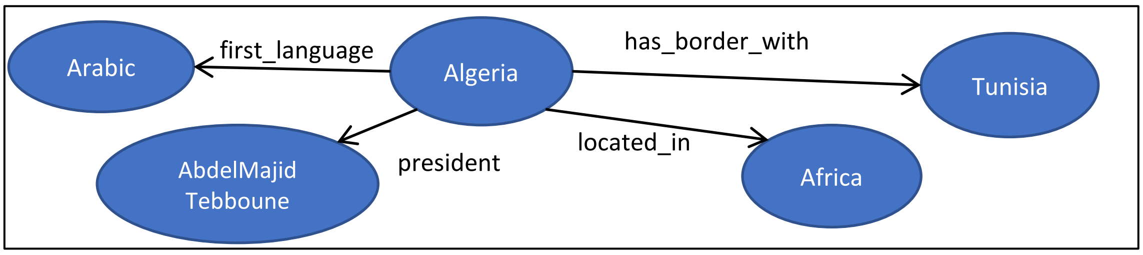


Figure 1 : Example of knowledge graph about the country of Algeria.

In comparison with the ontology, the KG does not necessarily contain the class and property hierarchy but it contains mainly the instance level. The KG is constructed using different methods like data crawling, knowledge extraction, knowledge fusion and processing. For the application level, the KG has a widespread usage in artificial intelligence application like link predication, entity resolution, recommendations...etc. The KG has been also used in network security for modeling the attacker behavior and predicting his steps. The figure 1 illustrates an example of a KG about the country of Algeria and its relations with other countries.

2.3 BlockChain

The blockchain technology is a distributed database that records all transactions executed by its distributed peers. These transactions are saved in interconnected blocks and they cannot be modified, therefore the blockchain keep the data integrity, confidentiality and privacy. This technology was born by the Bitcoin [9] peer-to-peer cash system and especially by the white paper of Nakamoto Satoshi. There are three types of blockchain: permissioned, public and consortium.

By the evolution and the success of the blockchain, many application domains like: the Internet of Things, agriculture, Smart Grids, Knowledge management…etc., have integrated this technology in order to keep their data integrity and privacy. 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.

2.4 Knowledge graph privacy and BlockChain

In recent years knowledge graphs and BlockChain have been merged to support artificial intelligence applications in many domains where the data knowledge and security paly central role. The former can be used to enhance the querying and reasoning capabilities of the BlockChain where the latter is used mostly to keep the knowledge graphs privacy, trust and security by making it protected and decentralized. The merging of these two technologies has been supported in recent years due to their success in many domains like data sharing [10] and recommendation systems [11].

In this section our focus is not on works that integrate KG in BlockChain because they are out of the scope of this paper. We refer the reader to [12], [13] and [15] to get more information about Blockchain enhanced by KG. We highlight the works which aim to securing the KG using the BlockChain technology. The only survey that we found about privacy in KG is [2] where the authors investigate privacy problems in KG and propose possible solutions to protect the KG privacy under isolated setting to support KG merging, query representation and completion. These possible solutions use secret sharing in order to keep the data more private during computation. In isolated setting the parties have their own KG and cannot share them with others. Therefore, it is essential to keep the KG content privacy during computing for example the embeddings of entities and relations.

The authors of [1] propose a new schema to improve the KG security using the BlockChain and distributed storage system. After processing the KG files using the distributed storage, their hashes are saved on the BlockChain and therefore, [1] work preserve only the hash integrity without data access control or the integrity of the whole KG. The work is evaluated using Hyperledger Fabric and Ethereum [28] where the former has given improved results compared to the latter. In [11] a new approach of deep recommendation system using KG is proposed. The construction process of KG is different from the traditional methods by using decentralization assured by BlockChain and smart contracts. Thus, KG problems like security, integrity and trust are the main objective of the work.

Table 1: Summary table of the related works

Approach	Blockchain	Offchain	Access control	IPFS	KG Management	Audit and Logs	Authentication	Rewards
[1]	Fabric 1	No	No	Yes	No	No	Fabric authentication	No
[13]	Ethereum	No	No	No	No	No	Ethereum authentication	No
[21]	Fabric 2.0	Yes (MangoDB)	No	No	Limited	No	Fabric authentication	No
[26]	Fabric 2.0	Yes (MangoDB)	No	No	Limited	No	Fabric authentication	No
PrivyKG	Fabric 2.0	Yes (MangoDB)	Yes	No	Yes	Yes	Fabric authentication	Yes

The work of [10] proposes OpenKG Chain, it is a network based on blockchain to share knowledge graphs in secure and trusted manner. The aim of the approach in [13] is not to preserve the KG privacy but its main goal is analyzing the cost between storing KGs or JSON into BC. The authors of [13] found that saving KGs takes more costs than JSON. Other works like [15] is centered on KG semantic sharing based on blockchain in order to decentralize the sharing process and protect the KG integrity. The authors found that using decentralization with blockchain improves the scalability and execution time compared with centralized KG sharing approaches.

In [21] and [26], blockchain has been used to keep the privacy and ensured the security of KG completion tasks such as link prediction and triple classification where they are executed in decentralization by BC peers. In the same endeavor, authors of [21] and [26] have used an off-chain and on-chain storage in order to ensure the BC scalability by storing only sensitive data which will be participated to rebuild the whole KG. Our work comes to fulfill the limitations of [12] and [6] such as marginalizing the KG privacy and access control, supporting only limit types of KG management, big offchain size…etc.

Table 1 illustrates the capabilities that can be supported by our work and other related works. In regards to the aforementioned approaches, they do focus on preserving only the privacy of KG content without taking into account several considerations such as access control, policies and audit mechanisms, as well as KG tasks must be protected because it is not enough focusing only on KG content and omitting the tasks that are crucial to put a complete KG privacy preserving approach.

Our work aims at proposing PrivyKG which takes into account the shortcoming of the previous works where the privacy protocol must cover all KG data from content to tasks for example querying and completion. In addition, we incentivize sharing KG content by using a reward mechanism that gives PrivyKG coins to KG owners who grants access to data requesters. PrivyKG supported permissioned blockchain rather than public ones, for several reasons such as high throughout, access control, trust among peers, scalability, and low energy consumption...etc. It also supports offchain KG data (data stored outside blockchain) for preventing unscalable blockchain processing.

3 Proposed approach of securing and preserving privacy of KG

In this section, we are to give details about how securing and keeping the privacy of KG using BC. Firstly, we are to introduce the notion of decentralized KG and how building it using on-chain and off-chain storage in order to keep its privacy. Secondly, we are to present PrivyKG architecture which uses access control, logs and reward mechanisms based on BC smart contract.

3.1 Decentralized knowledge graph

PrivyKG goal is keeping the security and privacy of KG using the blockchain. PrivyKG proposes to decentralize the KG which is illustrated in the figure 2. The process uses two main components of which offchain and onchain. The former contains off-chain KG data whereas the latter contains BC sensitive data about access control and critical data used to rebuild the original KG.

3.1.1 Off-chain knowledge graph privacy and storage

All offchain KG triples are stored according to their subjects, and therefore each off-chain entity is associated with its triples where it acts as subjects. Table 2 presents the data structure of the off-chain contents that contains two attributes: ObjectID and TripleSet.

Table 2. The offchain data about every KG entity.

Information	Description
ObjectID	The identifier that has been associated to the entity by the off-chain management.
Triples	It contains triples's relations and objects where the entity is the subject. A set of couples where each one is a form of (RelationID, ObjectID) where: - RelationID: The relation identifier that has used in the OffChain KG. - ObjectID: The identifier of the object that has used in the OffChain KG.

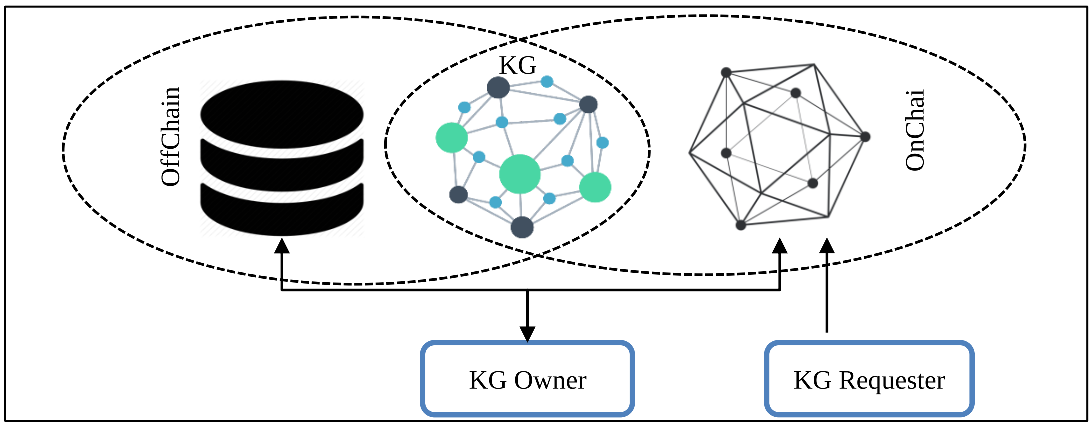


Figure 2: BlockChain based knowledge decentralization.

3.1.2 On-chain data about knowledge graph

In our design, the BlockChain has not been used to store the knowledge graph but its main goal is associating the entities with their positions in the off-chain storage. Our design proposes to use two data structures, one for the entities and the second is for the relations. Another important goal is avoiding the illegal changes and keeping the data integrity using the saved hashes. Each instance of these data structures contains important information about entities and relations such as the ciphertext of its name, Offchain ID and hash of its offchain triples. Only the entity structure contains the hashes of its triples in order to verify the off-chain storage integrity. Tables 3 and 4 show respectively the data structure for entities and relations.

Table 3: The blockchain transaction content about KG entities

Information	Description
EntityEnc	The ciphertext of the entity's name.
EntityID	The entity identifier that is used in the OffChain KG after decentralization.
EntityHash	The hash of the entity's name
Hash	The hash of the triples that is associated the entity.

Table 4: The blockchain transaction content about KG relations

Information	Description
RelationEnc	The ciphertext of the relation's name.
RelationID	The relation identifier that is used in the OffChain knowledge.
RelationHash	The hash of the relation's name.

3.1.3 Decentralized knowledge graph management

In this section, we show how the onchain and the off-chain storage are built from the original KG in order to keep both privacy and integrity. Besides we provide how updating, deleting and creating triples in the new decentralized KG. The KG management is very important because the continuous evolution of the knowledge data, it is also necessary in case of KG completion in order to add new true and predicted triples.

Decentralized KG. Firstly, the original KG entities and relations are associated with numerical identities to ensure the anonymization and decrease the offchain storage space. Secondly, the entities and relations names are encrypted to get ciphertexts. The data about every entity such as the ciphertext, identity in the offchain and the hash of its triples are stored on BC. The off-chain contains the new anonymized KG where every entity is stored using it off-chain identity and its triples according to table 2.

Decentralized KG Modifications. Three main operations can affect the state of the KG: creating new triples, updating or deletion of existing ones. Our design supports KG modifications by querying the offchain and the onchain without rebuilding or modifying the whole KG and making it decentralized again. It is necessary that after modification, the onchain hash associated with the subject entity must be recalculated to take into account the new changes.

Some supported KG are described by the following:
Triple Creation. A new triple can contain existing or new entities and relations. For the first case, the triple is inserted directly in its offchain after identifying the object ID of the subject from the BC. For the second case, we follow the strategy described during the creation of decentralized KG to create new relations and entities, after that, the same steps of the first case are forwarded.

Triple Update. In this situation, three cases can be occurred: subject or object or relation update. Before updating, we verify if we have new entities or relation and follow the same strategy of creating new triple components. For the first case, the new triple is moved from its triple set to the set of its new subject. For the second case, we modify its object directly with the object ID of the new object retrieved from the BC. In the third case, the triple is updated by the new relation ID.

Triple Deletion. In this case, the triple is removed directly from its off-chain set after identifying the object ID of its subject using BC.

The figure 3 shows an example about building and rebuilding the KG shown by the figure 1 using on-chain and off-chain data. The onchain data about "Algeria" must also contain its hash and the hash of all triples where it acts as a subject (4 triples). The offchain KG is stored according to table 2.

3.2 PrivyKG architecture and its main functionalities

PrivyKG components and their actors are illustrated by figure 4 where two main parts are presented: onchain and offchain. The first one contains all smart contracts proposed by PrivyKG in order to allow users interact with BC network. The second one covers offchain KG management and querying where the full privacy protection is guaranteed only by interacting with on-chain components.

In the next, we give detailed descriptions about PrivyKG components and their users along with their interactions, BC transaction types, access controls on using smart contracts, audit and reward mechanism…etc.

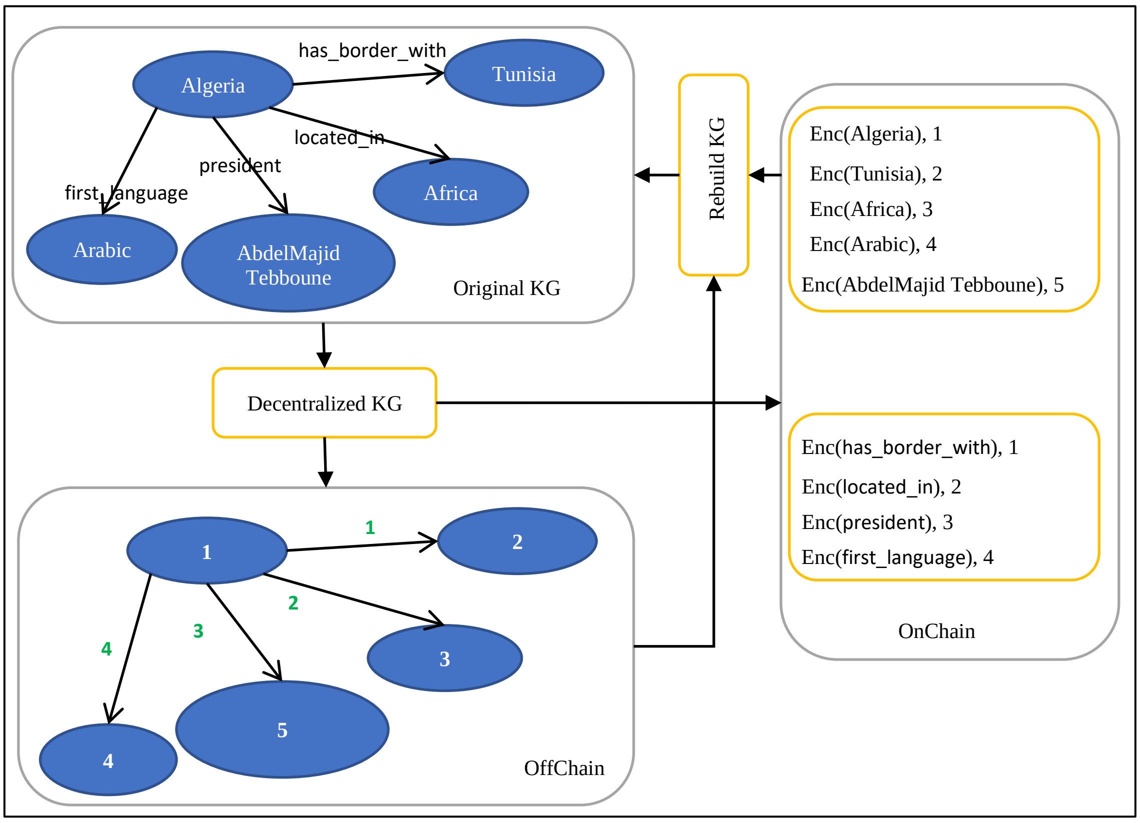


Figure 3: Example of knowledge graph decentralization and rebuilding.

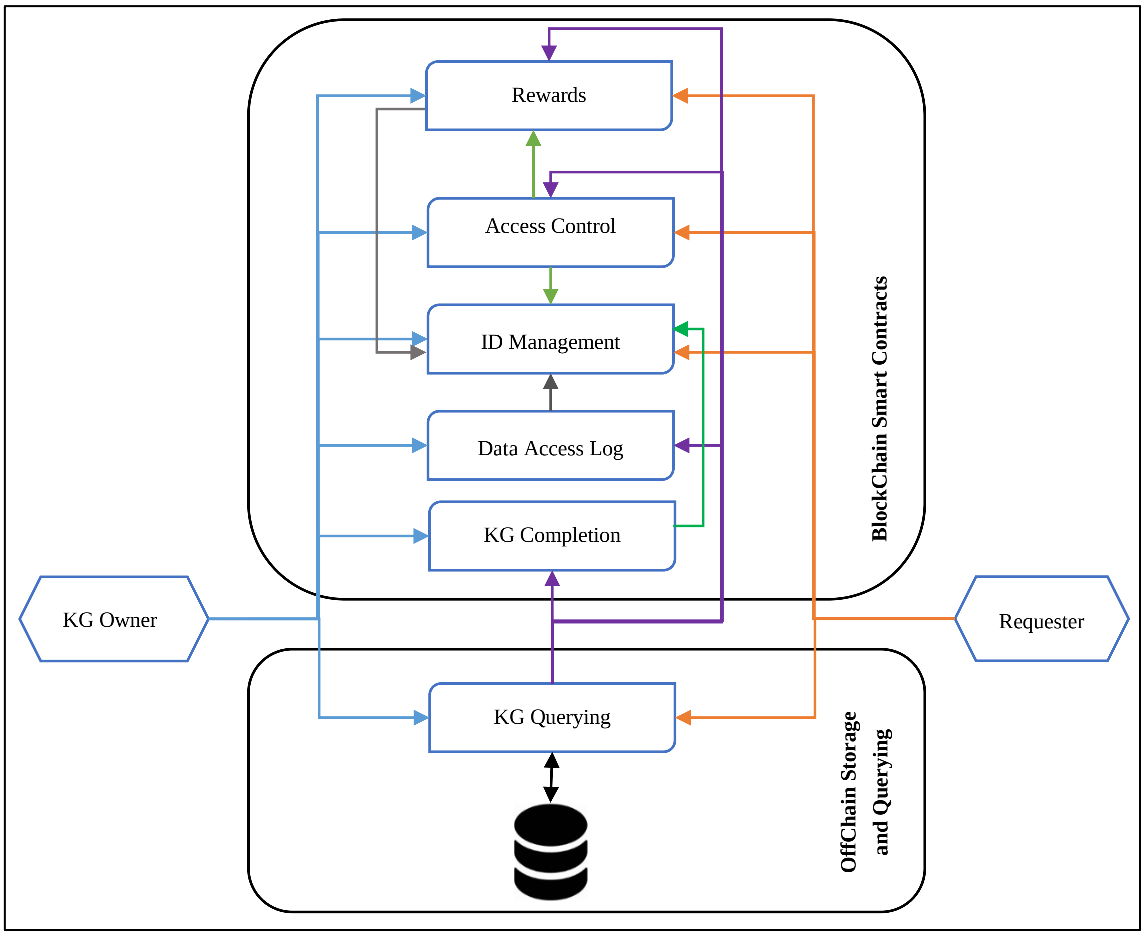


Figure 4: The main components of PrivyKG architecture along with theirs interactions.

3.2.1 Users roles and responsibilities

The KG Owner: the current data owner. It can be a person, enterprise, clinic, laboratory, etc. Recent data regulations like GDPR[1] insist that the data owners must have the full control over its data. Thus, they have imposed an access control and policies in order to restrict the access and the manipulation of their data. They put two types of policies: default and non-default where the first indicates that the target operation is granted without return back to KG owner, whereas the latter means that permission must be requested directly from the KG owner.

Requester: It acts as data stakeholders by requesting data access in order to use the requested KG in its processing. Therefore, every request is saved on the blockchain for future verification to detect illegal data manipulations. Thus, the requester must accept the rules of data processing before access the data. Every requester must register and enrolled by the blockchain in order to manage its authorization and access control. The data about the requester registration is under its responsibility and therefore every illegal access by another entity with its critical information is not the responsibility of the blockchain network.

3.2.2 On-chain components

Access Control: It verifies if the data requester has the right to query the KG with PrivyKG specific queries. The process is achieved by requesting the blockchain to get the access policies of the requested data. If the needed operation is in the default policy, then the access is granted, else the request is forwarded directly to the KG owner. In other situations, the requester can request data access after a permission that has been already given to him. Therefore, the AC verifies if the requester had an existed acceptation from the KG owner

Table 5: The blockchain transaction content about access control

Information	Description
KG owner identifier	The owner identity that is generated by the IDM.
KG identifier	A unique identity of the knowledge graph subject to access.
Requester identifier	The requester unique identity that is generated by the IDM
Query content	The content of the query to be executed for the requester.
Permission	The permission that is given to the requester for the requested query.

[1] https://gdpr-info.eu/

Table 5 gives an example of a transaction content which is supported by the AC smart contract in order to store a new permission.

Table 6: The blockchain transaction content about access logs

Information	Description
Requester identifier	The requester unique identity that is generated by the IDM
Query content	The content of the query to be executed for the requester.
Permission	The permission that is given to the requester for executing the query.
Date and Time	The date and time of the executed query.

Audit: its main objective is performing an advanced verification which detects inconsistencies in the history of authorization and access control components. It merges data from all ledgers and performs advanced checking. For example, if the AC component grants access permission without checking the authorized permission of the KG owner, then this illegal access can be detected using an audit verification. The process of auditing is started only in demand of the KG owner. Table 6 shows an example of a transaction content which is used by the audit smart contract in order to store a new log data about specific operation.

Identity management: it has two main tasks: the first is creating identities and registering the new owners and KG requesters (along with its attributes), the second is verifying if a given identity is valid or not using the blockchain. The IDM component returns a registration certificate to every accepted demand for registration where these latters contain critical information about enrolling PrivyKG users with different roles. All identity information is stored in the BC in order to protect it from fraud identities.

KG Completion: Its main task is assisting the KG owner in completing the data with missing ones. It suggests new KG triples that are predicted using KG completion tasks like link prediction or triple classification. It starts its task in demand from the KG owner and if this latter accepts the new triples ,then the KG is updated with these latter. The KG owner can provide to KG completion the plaintext or the ciphertext of the original KG. Tables 7 and 8 give an example of two transactions which is supported by the KG completion smart contract in order to store a new entity or relation embedding which will be used to compute the truth degree of a given triple.

Table 7: The blockchain transaction content about entity embeddings

Information	Description
EntityId	The ciphertext of the entity name.
VectorEmb	The vector embedding which was generated by a KG embedding method.

Table 8: The blockchain transaction content about relation embeddings

Information	Description
RelationID	The cipher text of the relation's name.
VectorEmb	The vector embedding which was generated by a KG embedding method.

Rewards: PrivyKG integrates a reward mechanism in order to incentivize KG owners to share their content with requesters. The KG requester rewards KG owners by giving coins which they have been purchased by requesters from the BC network. The collected coins can be used by KG owners to gain further PrivyKG services such as storage space, new functionalities, etc.

An example of a reward transaction content is given by the table 9 where every transaction is associated with information related to the number of coins, who gives them, and their destinations.

Table 9: The blockchain transaction content about rewards

Information	Description
Requester identifier	The requester unique identity that is generated by the IDM
KG owner identity	The owner's identity that is rewarded by the requester.
KG owner coins	The number of coins that are given by the requester to KG owner.

3.2.3 Off-chain components

KG querying: It has several specific functionalities for different KG owner and requester needs. It starts its work after receiving a request access from a KG requester which already has a valid access permission from the AC. It has a blockchain access and offchain access in order to target the correct entities and relations by associating their identifiers with the real ones in the offchain. In the next, we give detailed descriptions about some tasks provided by the KG Querying component.

- ✓ **Query result filtering:** If the result of a given query contains critical data about the KG owner, then these data are eliminated from the result in order to protect the privacy of the KG owner. The KG querying decides if a given data is critical or not using KG owner descriptions about critical attributes from the onchain.
- ✓ **Data about specific entity or relation:** The requester can request data that uses a specific entity or relation. Therefore, after getting the onchain data about the entity or relation, the KG querying component queries the decentralized KG for retrieving all triples that use the given entity or relation.
- ✓ **Data about relations that connect specific entities:** The requester can send query that contains only specific entities in order to get the relations that connect them.
- ✓ **Checking the truth value about a specific sub graph:** The requester can send a query which contains only specific subgraph (set of triples); thus, it considered as checking its truth i.e., testing if the KG contains the provide subgraph or not.
- ✓ **Using the KG completion to predict data:** the requester can demand from the system to provide the degree of truth of a set of triples. In this situation, the KG completion is requested for doing the predictions and send the result to the querying component.

3.3 Main scenarios in PrivyKG and their smart contracts

In this section, we put forward the three main scenarios that can be controlled by PrivyKG smart contracts.

The first one which is turns about how the permission is requested directly from the BC using two smart contracts: Access Control and Identity Management. These latter interacted with each other in order to achieve the request permission process. After successful identity verification, the AC verifies if the permission is listed in the access policy of the data requested and returned a positive response by updating the BC data with new granted permission.

The second one consists on how requesting the permission in case the BC cannot give it directly because the KG owner must be contacted. Moreover, the request permission is not specified in the default access policy and KG owner is the only one who gives it to the requester. The process uses the same smart contracts as the first interaction.

The last one presents how the requester gets the data after he got the permission. Firstly, the KG querying component interacted with two smart contracts (AC and IDM) in order to verify the requester identity and the given access permission. After that, it interacted with offchain server to get the data via using its information that has been given from the AC, besides it transfers the data to the requester. In the end, the audit smart contract is invoked in order to save the information about the operation.

PrivyKG supports other scenarios as follows:

- ✓ A given requester can update its number of coins which will be used in the future to reward the KG owner.
- ✓ A given KG owner can update its KG data ,thus updating the offchain and onchain entities and relations using respectively the KG querying and their smart contracts.
- ✓ After grant permission from KG owner, a requester can query the KG ,for example to check the degree of truth of a given subgraph ,where this task can incorporate the KG completion smart contracts.

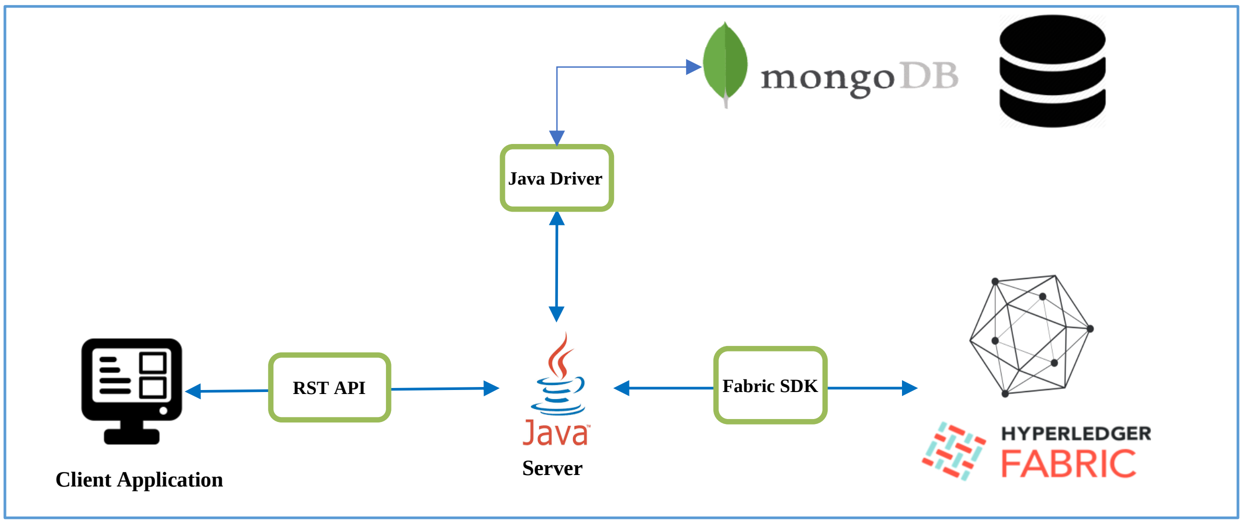


Figure 5: PrivyKG implementation architecture.

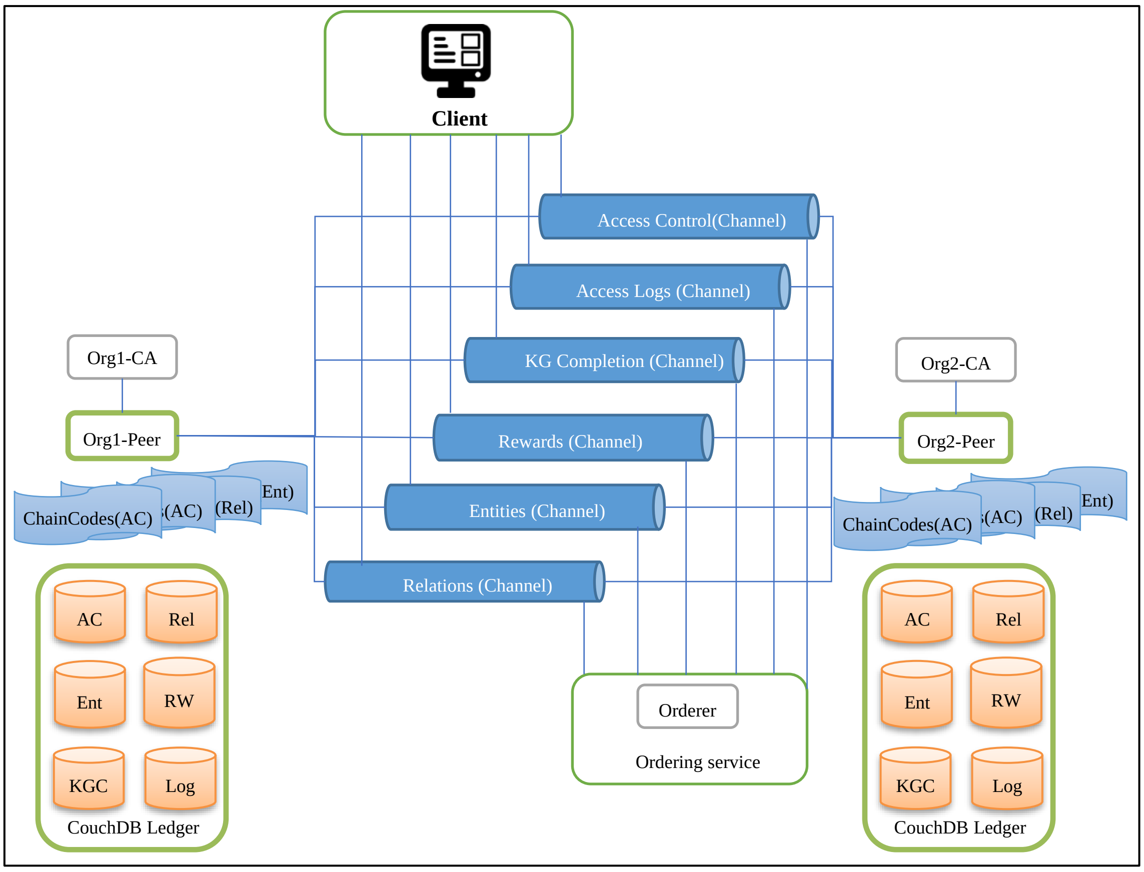


Figure 6: The hyperledger fabric network used in PrivyKG.

4 Implementation

PrivyKG is implemented under Eclipse using various Java APIs such as JGraphT, and JSON, Fabric SDK. The implementation architecture is illustrated in figure 5 where the main components are as follows:

The Hyperledger Fabric Blockchain: [17] is used with the configuration of two organizations and one peer node for each one. The fabric network uses CouchDB as world state database and one ordering service. It was built with one certificate authority for each organization. Six channels are created for access control, access logs, knowledge completion, rewards, entities and relations named respectively "Access Control", "Access Log", "Knowledge Completion", "Reward", "Entities" and "Relations". Six Fabric smart contracts are deployed using Go language (one for each channel). The Hyperledger Fabric network used by the proposed method is given by the figure 6 where every channel is associated with its ledger and its smart contract.

MangoDB: is used as off-chain storage where every data is stored as JSON objects. Queries are specified using the NoSQL to interrogate the database in order to get or modify the triples.

Client Application: a user that needs data from the decentralized KG.

All these latter are interacted with the main program using their specific Java API.

4.1 Fabric chaincodes and distributed ledgers in PrivyKG

Every peer in HLF has its local database (ledger) with contains all transactions executed by the network via HLF chaincodes. Thus, every peer can have several installed chaincodes for one HLF channel. The distributed ledgers in HLF are updated using smart contracts in demand by the blockchain external users. PrivyKG proposes to use 6 distributed ledgers where each one is associated with one smart contract and several peers. These ledgers store critical data about PrivyKG functionalities such as knowledge graph completion, rewards, access control, access logs, on-chain KG (entities and relations). In the next, we give some details about access control, reward and KG entities and relations chaincodes.

4.1.1 Access control chaincode

The AC chaincode defines functions that executed by HLF peers for managing the KG access control and defining the permission required to execute KG operations. This chaincode is installed on a channel identified by the same name "Access control" and it is associated with a local ledger that saves information about the access control on KG data. The AC chaincode uses the Golang structure which is illustrated by the listing 1. AC chaincode functions like creating new permission must use a JSON key value passed in the invocation call by the HLF users. All information included in listing 1 are already explained by the table 5. Table 10 presents some functions provided by AC chaincode.

```
type AccessControl struct {
      OwnerID string `json:"OwnerId"`
      PolicyKG [] PolicyTypeKG
`json:"PolicyKG"`}
type PolicyTypeKG struct {
      KgID string `json:"KgID"`
      DataPermissions []
DataPermissionType
`json:"DataPermissions"`}
type DataPermissionType struct {
      QueryType string
`json:"QueryType"`
      DefaultPolicy [] string
`json:"DefaultPolicy"`
      Permissions [] PermissionsType
`json:"Permissions"`}
type PermissionsType struct {
      Target string `json:"Target"`
      Op [] string `json:"Op"`}
```

Listing 1: The Golang structure used by the access control chaincode

Table 10: Some smart contract functions that are implemented by the access control chaincode.

Function	Description
addPermission	Create new permission for a given requester for accessing a given KG data.
chkPermission	Check if a given requester has already registered with a given permission.
updPermission	Update a given permission by removing or extending or restricting it.

4.1.2 Rewards chaincode

The RW chaincode uses the Golang structure which is illustrated by the listing 2 where All information are already explained by the table 9. Table 11 presents some functions supported by RW chaincode.

```
type Rewards struct {
      RequesterID string
`json:"RequesterID"`
      Coins int `json:"Coins"`
      TransferCoins [] TransferType
`json:"TransferCoins"`}
type TransferType struct {
      OwnerID string `json:"OwnerId"`
      Coins int `json:"Coins"`}
```

Listing 2 : The Golang Structure used by the Reward Chaincode

Table 11: Some smart contract functions that are implemented by the Reward chaincode.

Function	Description
rewardUser	A given requester rewards a KG owner with coins.
putCoins	A given requester updates its coins with new ones.

4.1.3 Knowledge graph chaincodes

Two chaincodes are created for managing the critical data used to identify the KG entities and relations and theirs related triples in the off-chain. The first one is for the entities while the second one is for the relations. The content of every ledger is already discussed in the tables 3 and 4. The entities and relations chaincode uses the Golang structures that are illustrated by the listings 3 and 4 respectively. Tables 12 and 13 shows some functions implemented respectively by entities and relations chaincodes.

Table 12: Some smart contract functions that are implemented by the Entity chaincode

Function	Description
createEntity	Create new critical data about a new entity for a given KG.
readEntityData	Get the information stored in Entity ledger about a given KG entity.
updateEntity	Update the blockchain data about a given entity such as the hash and the offchain ID.

Table 13: Some smart contract functions that are implemented by the relation chaincode

Function	Description
createRelation	Create new critical data about a new relation for a given KG.
readRelationData	Get the information stored in relation ledger about a given KG relation.
updateRelation	Update the blockchain data about a given relation such as the relation ID.

```
type Entity struct {
      EntityEnc string
`json:"entity_enc"`
      ObjectID string
`json:"object_id"`
      EntityHash string
`json:"entity_hash"`
      Hash string `json:"hash"`}
```

Listing 3: The Golang Structure used by the Entity Chaincode

```
type relation struct {
      RelationEnc string
`json:"relation_enc"`
      RelationID string
`json:"relation_id"`
      RelationHash string
`json:"relation_hash"`}
```

Listing 4: The Golang Structure used by the Relation Chaincode

5 Evaluation

5.1 Knowledge graph dataset

DBpedia [25] ontology is a large-scale dataset extracted from Wikipedia, it has widespread usage in many artificial intelligence domains such as KG completion and reasoning. It plays a centric role in the link data project where it acts as a mediator ontology that connects several datasets. *PrivyKG* is evaluated using a DBpedia dataset named DBpedia 50k [27] that contains 30436 entities, 365 relations and 50000 triples. In the first experiment, 20k triples are selected. For the next one another 20k is added and so on until reach the whole dataset. In every experiment, a MangoDB collection is created from the selected dataset in order to build the offchain data which is associated with an onchain data.

The table 14 gives some properties about the DBpedia 50k samples such as the number of triples, relation and entities.

5.2 Experiment configuration

To validate the functionality and test the performance of *PrivyKG*, a number of experiments have been 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. The components of the fabric network are deployed as Docker 2.3 images (Organizations, certificate authorities, peers, CouchDB. etc.). The offchain data is represented as a database that uses MangoDB 4.4.1 as a database management system. In regards to the implementation architecture, PrivyKG server is implemented as JAVA REST web application that uses the Tomcat 9 as a resource server. Every KG owner or requester is depicted as JAVA standalone applications that communicate with the mediator (server) using REST API. They have also interactions with Fabric network using Fabric SDK. The offchain implementation is interacted only with the KG querying component and the KG owners to manage their own KG by using NoSQL queries. Our implementation uses several Java API in different processes such as JENA, MangoDB driver, IPFS API, Fabric SDK…etc.

5.3 Experiment results

Privy KG experiments are focused on decentralized KG building, querying, KG checking and rebuilding, thus results about KG completion tasks, audit and reward mechanisms are not given in this paper.

5.3.1 PrivyKG on-chain and off-chain creation

Table 15 presents onchain and offchain sizes after KG decentralization where the entities' ledger sizes are related to the number of KG entities while the same is for the relations. The size of the former is big than the latter because the number of entities is bigger than the number of relations. PrivyKG also save additional information in the entities' ledger (such as triples hash) compared to relations ledger. There is a significant reduction of offchain sizes compared to previous works [21, 26] due to enhancements applied during offchain creation like removing object identifiers and replace them with numerical numbers.

Figure 7 shows execution times during KG decentralization which contains offchain construction, relation's ledger creation and encrypting entities and relations. There are few differences between the amount of time needed to complete the previous tasks. In some cases, the execution time for the current dataset is less than the previous one even the size of the latter is big than the former. This is because in Java environment, executing the application many times can reduce the execution time due to the optimization applied by JVM.

Table 14: KG datasets used to evaluate PrivyKG.

Datasets	**DBP10K**	**DBP20K**	**DBP30K**	**DBP40K**	**DBP50K**
Size (triples)	10000	20000	30000	40000	50000
Relations	309	332	345	362	365
Entities	8686	16202	23086	29179	30436

Table 15: Sizes of KG datasets (in KB) after onchain and offchain creation

Datasets	**DBP10K**	**DBP20K**	**DBP30K**	**DBP40K**	**DBP50K**
Entities	6860,8	18432	26112	28364,8	36556,8
Relations	212,7	397,8	614,4	819,2	921,6
MangoDB	69,63	454,66	651,26	839,68	921,6

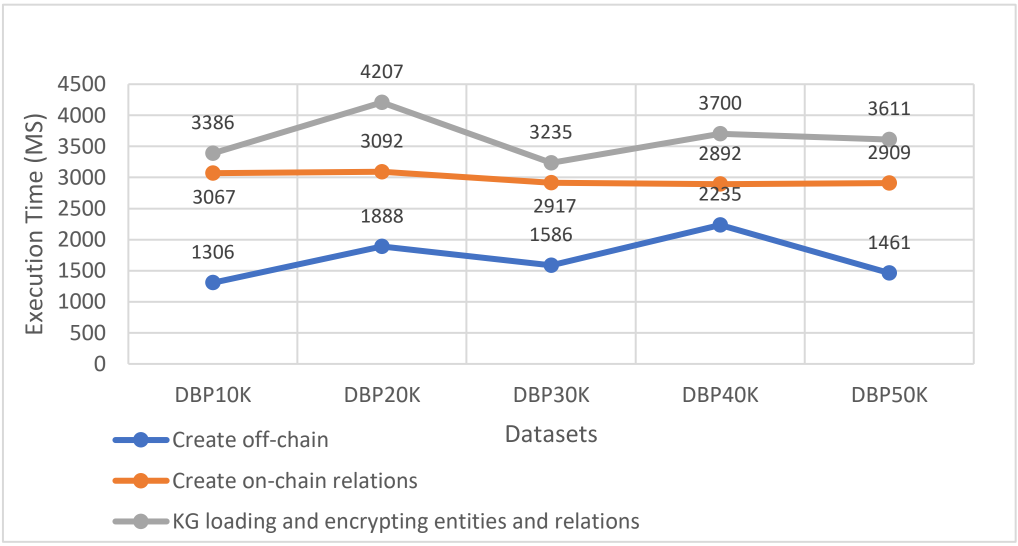
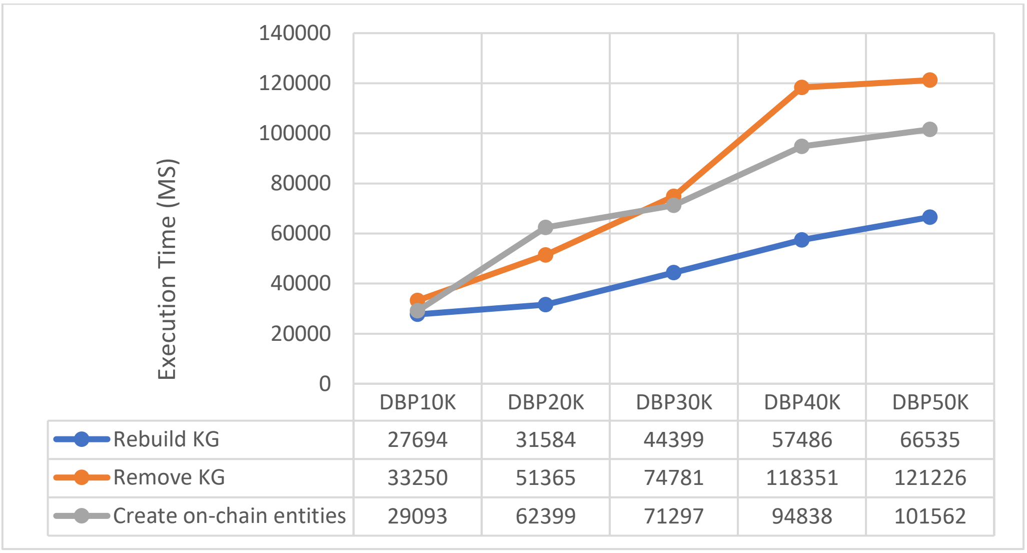


Figure 7: PrivyKG evaluations about encryption, offchain and onchain relations creation.

Compared to previous evaluations, figure 8 presents increasing execution times related to creating onchain entities, removing decentralized KG and rebuilding original KG. All these tasks are related extremely to the number of entities. The execution time of the first two tasks is less than the third one, this is because creating or removing tasks use BC transaction that needs writing on the entity's ledger while rebuilding need only reading from it. In our experiment, we have used a Hyperledger Fabric network that limits the size of transaction to 4 MB using GRPC protocol. Therefore, it must increase this default value to decrease the execution times of these tasks.

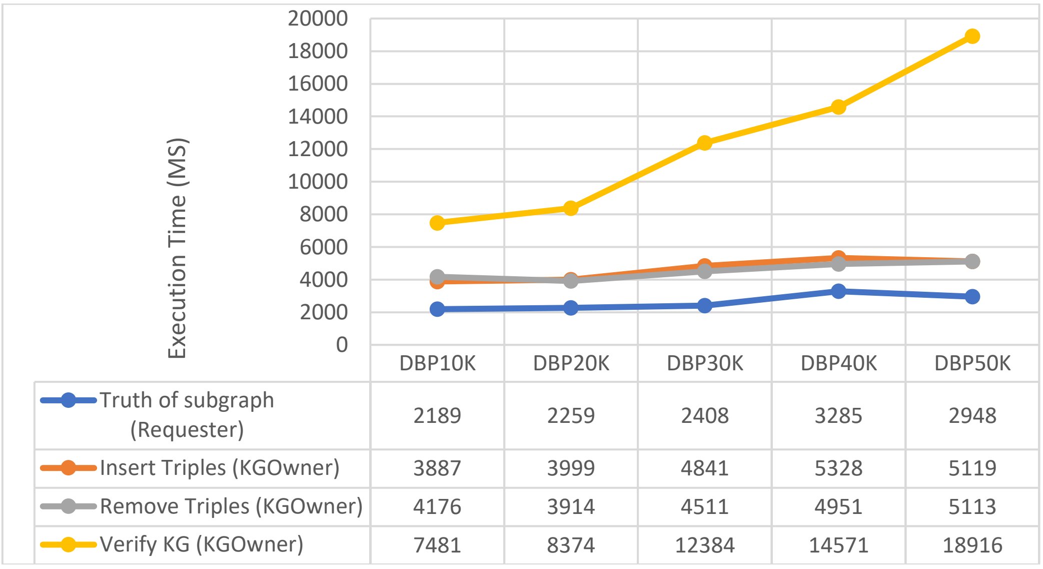
5.3.2 PrivyKG querying and KG management

In this experiment, PrivyKG is tested using four types of queries where every query contains 50 triples, one for the requester and three for KG owner. The first one checks the truth of a given subgraph and it is submitted by the requester after getting the access permission from the KG owner. The next two queries are about updating and inserting new KG triples where they are submitted by KG owner. The last query is sent by KG owner to verify if there are illegal changes in KG by using the onchain hash of triples. Figure 9 illustrates the execution times of every query. From the evaluation results, we can see that there are few changes between the times for the requester query where the KG size has little influence on the execution. The same is for the insert and update queries. As explained before, the JVM can influence the execution times like between the last dataset and the previous one. The query of verification takes more execution times compared to previous queries because it requires reading all offchain data and get all hash of triples from the BC, whereas the other queries can verify or add or remove a limited number of entities and relations.



	DBP10K	DBP20K	DBP30K	DBP40K	DBP50K
Rebuild KG	27694	31584	44399	57486	66535
Remove KG	33250	51365	74781	118351	121226
Create on-chain entities	29093	62399	71297	94838	101562

Figure 8: PrivyKG evaluations about On-chain entities creation, rebuilding and removing the decentralized KG.



	DBP10K	DBP20K	DBP30K	DBP40K	DBP50K
Truth of subgraph (Requester)	2189	2259	2408	3285	2948
Insert Triples (KGOwner)	3887	3999	4841	5328	5119
Remove Triples (KGOwner)	4176	3914	4511	4951	5113
Verify KG (KGOwner)	7481	8374	12384	14571	18916

Figure 9: PrivyKG evaluations about updating and querying the decentralized KG.

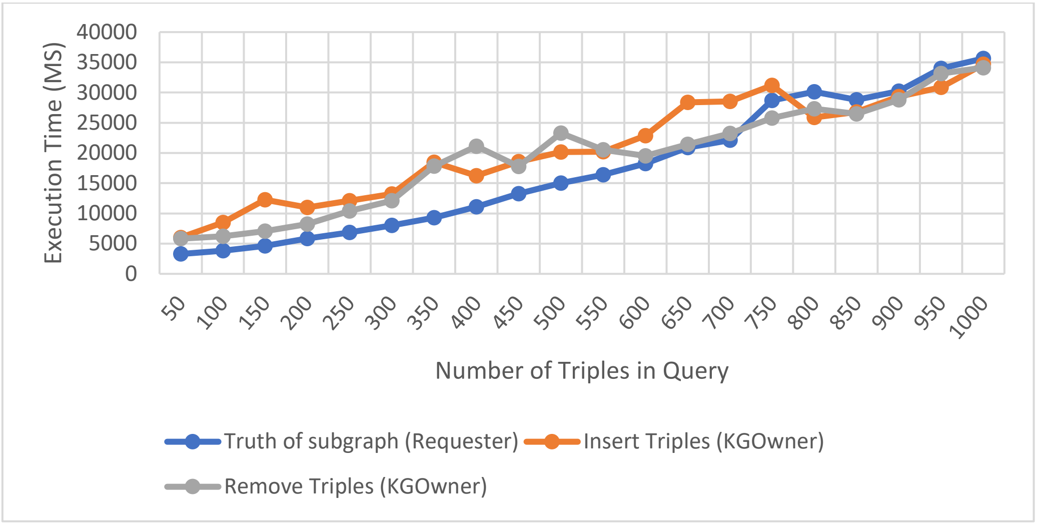


Figure 10: PrivyKG evaluations about updating and querying DBP50K with increasing number of triples in every query.

The last experiment is performed on the whole dataset DBP50K by using three types of queries (checking subgraph truth, insert and remove triples) with a variable number of triples. In each time, 50 triples are added to the previous query in order to construct the current one. The execution times are given by the figure 10 where the needed time to execute a query is related to the number of triples. In some cases, there is a decreasing execution time because for example during inserting or removing new triples, the current query can contain less new entities triples, compared to previous one and this affects the onchain writing time.

5.4 Discussion

The most evaluation results are related extremely to the number of entities because the PrivyKG strategy to build a decentralized KG consists in creation a distributed ledger entry for every entity. Therefore, most execution times have been influenced by transactions that write new entities. If Hyperledger Fabric network is configured for allowing big size transactions then the execution times would be decreased.

PrivyKG strategy to create an offchain data has given its advantage to reduce the onchain data size. In this situation, offchain storage system will manage data about triples where onchain ensure the privacy and integrity of the stored entities.

6 Discussion

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

6.1 Security Analysis

This subsection demonstrates the efficiency of our method to deal with the security and the privacy of building and querying KGs in decentralized environment, we put forward the threats that could possibly face the proposed method and how we can treat with them.

We highlight some assumptions that may threaten PrivyKG and how they could be resolved.

- ✓ **Threat 1**: We assume that an external adversary tries to make KG modifications.
  Resolution: In order to achieve this goal, the attacker must have the secret key and public key which has been used respectively to encrypt and decrypt the entity and relation names, the hashing strategy, the certificate to access the blockchain and the smart contact code, the user-name and password to access the off-chain storage. It is very difficult to obtain all these data together. Any changing in the KG can be detected using hash checking of triple sets.
- ✓ **Threat 2:** We assume that an external adversary has made off-chain changes by adding triples or update their components.
  Resolution: Any changing in the offchain data can be detected by checking the onchain hash of triples.
- ✓ **Threat 3:** We assume that a registered KG requester tries to invoke chaincodes related KG owner.
  Resolution: PrivyKG puts access controls on chaincode invocation using client identities and roles. For example, during the registration of a KG requester, the BC uses the role “requester” to identify the client type. In this situation, every requester can only invoke chaincodes related to the requester role. The same method is used when register KG owners.
- ✓ **Threat 4:** In the situation when there are several KGs with different owners, a KG owner tries to query another KG without permission.
  Resolution: PrivyKG chaincodes are implemented to only allow the real KG owner to get information or modify its KG. Therefore, every KG owner must specify one of its KGs to achieve chaincode invocation without any fail.
- ✓ **Threat 5:** In BC network, every node has a copy of all executed transactions and therefore it can access to onchain data about entities and relations, it tries to get information in order to rebuild the original KG.
  Resolution: All entities and relations names are stored on the BC using only their ciphertexts. Thus,

it is impossible to get real names without the KG owner secret key. In this situation, BC nodes cannot benefit from saved transactions.

Hyperledger Fabric as a permissioned BC takes an important role to resolve most threats due to its features that provides access controls, authentication and authorizations of PrivyKG clients. By using Fabric chaincode, PrivyKG implements various access controls and policies to allow only legitimated clients to use system functionalities according to their roles.

6.2 Comparison

As it is mentioned before in the related works, that there are few works that have strong relation with PrivyKG where they focus only on protect the KG content and keep its privacy. Compared to PrivyKG, the related works have several drawbacks and they lack important features to provide ensure the KG privacy and security. Some lacks are illustrated by the following:

- ✓ All related works do not use an access control and policies over the KG data.
- ✓ The majority of works do not support offchain data outside the blockchain in order to prevent unscalable situations.
- ✓ Future audit and log verifications are omitted by all related works and therefore they cannot ensure the verification of illegal KG data access.
- ✓ All works do not support the KG management such as updating or adding new knowledge without getting the whole KG data.

All the aforementioned drawbacks are taken into account by PrivyKG in order to put full KG privacy protection and allowing KG owners to manage and share their data in secure and trusted manner.

Compared to PrivyKG, the work of [1] do not support several important features to ensure full KG privacy protection such as access control and audit mechanisms, it uses the BC only to store the hash of KG data for ensuring future KG verification. Our direction to use an offchain complies with the results found by [13] where the researchers found that storing the whole KG into BC affects its scalability and increase its processing costs.

PrivyKG extends the works of [21] and [26] by enhancing the privacy protection using access control and allow new types of queries to be performed on decentralized KG. PrivyKG also add a reward mechanism in order to encourage KG owner to give permission to requesters. It also reduces the offchain storage size compared to [21] and [26].

7 Conclusion

In this paper, we proposed PrivyKG, which is a new approach for securing and keeping the privacy of knowledge graphs. PrivyKG uses onchain and offchain storage to ensure BC scalability by incorporating respectively Hyperledger Fabric and MangoDB. PrivyKG supports several types of queries and access controls on KG data and it allowed audit and reward mechanisms. The implementation and evaluation of PrivyKG demonstrate its feasibility to achieve the goal of preserving the privacy and ensuring the security of KG querying.

PrivyKG can be seen as a general framework, as well as it can be used to enhance the security and privacy during the KG processing in artificial intelligence application.

Future works may include extending PrivyKG by the following:

- ✓ Instead of using MongoDB, PrivyKG can store its off-chain data in IPFS (Inter Planetary File System) to ensuring the scalability of the storage.
- ✓ Extending PrivyKG to support new types of queries because if SPARQL queries are used then it is necessary to rebuild the original KG. Therefore, PrivyKG must support SPARQL queries on decentralized KG without getting the whole KG.
- ✓ Extending PrivyKG to deal with distributed knowledge graphs where the data is published and stored in multiple sources. In this situation, PrivyKG must keep the privacy and ensure the data integrity of all KGs and allow for example secure KG alignments.

References

[1] Wang Y., Yin X., Zhu H., Hei X.: A Blockchain Based Distributed Storage System for Knowledge Graph Security. In: Sun X., Wang J., Bertino E. (eds) Artificial Intelligence and Security. 2020. LNCS, vol 12240. Springer, Cham.

[2] Chen, C., Cui, J., Liu, G., Wu, J., Wang, L. (2020). Survey and Open Problems in Privacy Preserving Knowledge Graph: Merging, Query, Representation, Completion and Applications. ArXiv, abs/2011.10180.

[3] Zyskind, G., Nathan, O., Pentland, A. (2015). Decentralizing Privacy: Using Blockchain to Protect Personal Data. 2015 IEEE Security and Privacy Workshops, 180-184.

[4] Truong, N., Sun, K., Lee, G., & Guo, Y. (2020). GDPR-Compliant Personal Data Management: A Blockchain-Based Solution. IEEE Transactions on Information Forensics and Security, 15, 1746-1761.

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

[6] Bollacker, K., Evans, C., Paritosh, P.K., Sturge, T., & Taylor, J. (2008). Freebase: a collaboratively created graph database for structuring human knowledge. SIGMOD Conference.

[7] Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P.N., Hellmann, S., Morsey, M., Kleef, P.V., Auer, S., & Bizer, C. (2015). DBpedia - A large-scale, multilingual knowledge base extracted from Wikipedia. Semantic Web, 6, 167-195.

[8] Miller, G. (1995). WordNet: a lexical database for English. Commun. ACM, 38, 39-41.

[9] Nakamoto, S. (2009). Bitcoin: A Peer-to-Peer Electronic Cash System.

[10] Chen, H., Hu, N., Qi, G., Wang, H., Bi, Z., Li, J., & Yang, F. (2021). OpenKG Chain: A Blockchain Infrastructure for Open Knowledge Graphs. Data Intelligence, 1–18.

[11] Wang, S., Huang, C., Li, J., Yuan, Y., Wang, F. (2019). "Decentralized Construction of Knowledge Graphs for Deep Recommender Systems Based on Blockchain-Powered Smart Contracts". In: IEEE Access, vol. 7, pp. 136951-136961.

[12] Abu-Naim, B., & Klas, W. (2019). Knowledge Graph-Enhanced Blockchains by Integrating a Graph-Data Service-Layer. 2019 Sixth International Conference on Internet of Things: Systems, Management and Security, 420-427.

[13] Cimmino, A., García-Castro, R., & Cano-Benito, J. (2020). Benchmarking the efficiency of RDF-based access for blockchain environments. SEKE.

[14] Tuán, A., Hingu, D., Hauswirth, M., & Le-Phuoc, D. (2019). Incorporating Blockchain into RDF Store at the Lightweight Edge Devices. SEMANTiCS.

[15] Zhang B., Li X., Ren H., Gu J. (2020) Semantic Knowledge Sharing Mechanism Based on Blockchain. In: Liu Y., Wang L., Zhao L., Yu Z. (eds) Advances in Natural Computation, Fuzzy Systems and Knowledge Discovery. ICNC-FSKD 2019. AISC, vol 1075. Springer, Cham.

[16] Boschin, A. (2020). TorchKGE: Knowledge Graph Embedding in Python and PyTorch. ArXiv, abs/2009.02963.

[17] 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.

[18] R. L. Rivest, A. Shamir, and L. Adleman. 1978. A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM 21, 2 (Feb. 1978), 120–126. https://doi.org/10.1145/359340.359342

[19] Araki, K., Satoh, T., Miura, S. (1998). Overview of elliptic curve cryptography. In: Imai, H., Zheng, Y. (eds) Public Key Cryptography. Lecture Notes in Computer Science, vol 1431. Springer, Berlin, Heidelberg. https://doi.org/10.1007/BFb0054012.

[20] CREMONEZI, BRUNO, Borges, Alex, Miranda Nacif, José and Nogueira, Michele. (2020). Survey on Identity and Access Management for Internet of Things. https://doi.org/10.21203/rs.3.rs-66793/v1.

[21] A. Djeddai and R. Khemaissia, "Keeping the Privacy and the Security of the Knowledge Graph Completion Using Blockchain Technology," 2022 4th International Conference on Pattern Analysis and Intelligent Systems, 2022, pp. 1-6, https://doi.org/10.1109/PAIS56586.2022.9946869.

[22] A. Narayanan and V. Shmatikov, "Robust De-anonymization of Large Sparse Datasets," 2008 IEEE Symposium on Security and Privacy (sp 2008), 2008, pp. 111-125, http://doi: 10.1109/SP.2008.33.

[23] Latanya Sweeney. k-anonymity: A model for protecting privacy. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 10(05):557–570, 2002.

[24] Cynthia Dwork. Differential privacy. In Automata, languages and programming, pages 1–12. Springer, 2006.

[25] Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P.N., Hellmann, S., Morsey, M., Kleef, P.V., Auer, S., & Bizer, C. (2015). DBpedia - A large-scale, multilingual knowledge base extracted from Wikipedia. Semantic Web, 6, 167-195.

[26] Djeddai, A. (2022). KGChain: A Blockchain-Based Approach to Secure the Knowledge Graph Completion. In: Chbeir, R., Manolopoulos, Y., Prasath, R. (eds) Mining Intelligence and Knowledge Exploration. MIKE 2021. LNCS, vol 13119. Springer, Cham. https://doi.org/10.1007/978-3-031-21517-9_22.

[27] Shi, B., & Weninger, T. (2018, April). Open-world knowledge graph completion. In Proceedings of the AAAI conference on artificial intelligence (Vol. 32, No. 1).

[28] G.Wood et al. Ethereum: A secure decentralised generalized transaction ledger. Ethereum project yellow paper, 2014.