# Proxy re-encryption for privacy enhancement in Blockchain: Carpooling use case

Damien Zonda, Maroua Meddeb

# Proxy re-encryption for privacy enhancement in Blockchain: Carpooling use case

Damien Zonda
*IRT-SystemX*
*damien.zonda@irt-systemx.fr*

Maroua Meddeb
*IRT-SystemX*
*maroua.meddeb@irt-systemx.fr*

***Abstract*—Blockchain is an especially promising and revolutionary technology that brings transparency in a scalable way for multiple organizations and this is thanks to its several features. There are some exciting blockchain features but among them, decentralization is undoubtedly the most interesting one. Organizations can share data within a distributed ledger. As a consequence, each one can access synchronized data stored in its local Blockchain node. This functionality improves transaction tracking and facilitates access to data within a private group of organizations. However, in some cases, even if organizations accept to share data, they require to hide some private information related to their users or their business model. To tackle privacy and trust issues between organizations, this paper presents a blockchain architecture based on the proxy re-encryption scheme. This scheme is integrated within smart contracts to provide a very efficient, fast, and secure platform. The proposed architecture is implemented in an Hyperledger Blockchain and tested in a real transport and mobility use case.**

***Index Terms*—Privacy, Blockchain, Proxy re-encryption, smart contract, mobility**

## 1. Introduction

A blockchain can be viewed as a distributed ledger that is shared, replicated, and synchronized among different nodes in the network. In addition to shared data, this ledger can hold records of transactions in order to track all the interactions between nodes as well as exchanged assets. All the records within a blockchain are held permanently in a sequential chain of blocks and can never be altered. This feature makes a blockchain tamper-evident. An update of a record is performed by adding a new record with new updates [1].

The most interesting feature provided by a blockchain is decentralization. Blockchain architecture is managed without the involvement of a central authority. In fact, the participant organizations in the blockchain govern together and decide together if they agree or not to add a new block based on consensus mechanims [1]. This mechanism leads to provide a consistent ledger with reduced errors and to increase trust among organizations as well as transaction integrity. The agreement between different parties is automated. This is called a smart contract. It's a self-executing code that implements the terms of the agreement [2]. Smart contracts within blockchain natively offer fast, dynamic and real-time updates, and low cost of operation.

Despite the design efforts to make the blockchain a secure and trustful architecture, such as ensuring data integrity or applying user access control in smart contracts, the blockchain is still suffering from some privacy issues. In fact, due to its transparency and decentralized nature, a blockchain allows any node to access transactions. Malicious users can then trace information anonymously. This can be considered as a lack of privacy. In this paper, we target this issue [3]. The notion of privacy concerns two main areas; Confidentiality and Control. For confidentiality, privacy concerns the protection of personal data against unauthorized or anonymized accesses. Many different mechanisms can be employed to ensure integrity, anonymity, unlinkability, communication protection, undetectability, and unobservability. For control, privacy concerns the right given to users to control and manage personal data at any time, ensuring user self-determination [3].

This study is part of a project in the context of mobility. In order to encourage and facilitate carpooling behavior, the project proposes a platform based on blockchain technology that targets carpooling operators interoperability. The carpooling platform aims to interconnect different carpooling operators so they can push their carpooling offers in the blockchain. In this way, the carpooling platform makes independent carpooling services interoperable. As a consequence, it allows the matching and payment between drivers and passengers registered in different operators.

The main purpose is to guarantee operators' confidentiality from the business model point of view. In fact, the carpooling platform distributes offers in each operator node. Consequently, each operator can see in its local node other carpooling operators' offers as well as bookings and can then compute the dynamic and the gain of other operators. To address this issue we propose a proxy re-encryption-based carpooling platform. This architecture aims to keep distributing these entities in all nodes, however, all identity-reveal fields will be hidden and need authorization to be seen.

The remainder of this paper is organized as following; in Section 2, we will start by introducing our blockchain-based carpooling platform and explaining its functionalities as well as its privacy requirements. Then, in Section 3, we will

present a state of the art. The proposed scheme is explained in Section 4. Discussion and limitations are presented in Section 5. Finally, Section 6 presents our conclusions.

## 2. Carpooling platform overview

The main goal of our project is to reduce traffic jams. Our challenge is to increase the occupancy rate of individual vehicles. Carpooling is a decisive response, but the dispersion of offers in different carpooling operators and the absence of a market place at the scale of a territory affect its performance and credibility. The objective is therefore to increase the number of carpooling offers by consolidating the services on a regional scale by making available tools to interconnect the various operators. In this context of interoperability, users will be able to benefit from a globally aggregated offer and inter-operator trips. In this context, the blockchain-based carpooling platform is designed. This platform proposes to share carpooling offers created by different operators, so a passenger from carpooling operator A can travel with a driver from carpooling B and this while preserving the user journey. In other words, end-users will continue to use their favorite operator's user interfaces and they will just notice that the offers catalog has gotten richer.

### 2.1. Carpooling use case

We briefly present in this subsection, the functionalities of the carpooling platform to understand its privacy requirements. Users of our carpooling platform are carpooling operators and not end-users. That means passengers and drivers do not interact directly with the blockchain. Each carpooling operator maintains a blockchain node in which they can publish their carpooling offers. The scenario starts with a driver from operator A who proposes a carpooling offer. The offer is stored locally in the carpooling operator A then pushed to the carpooling platform. Once the offer is created and stored in the Blockchain, passengers from different carpooling operators can fetch it. A passenger can search for an offer with an origin, a destination, and a date through a carpooling operator B. The carpooling operator B sends a request to its blockchain node which responds by sending a list of offers that meets the given criteria. The list includes offers belonging to different carpooling operators. The passenger chooses an offer that suits him and books it. The booking transaction is then stored in the blockchain and the platform notifies operator A about the booking. The driver can then confirm or reject the booking. Confirmation/Rejection transaction is also stored in the Blockchain and the passenger is notified through his carpooling operator B. In the case of confirmation, the number of available seats decreases. The passenger may decide to cancel its booking, then a Booking_cancelation transaction is stored and the driver' operator A is notified. The number of available seats is updated. With the same logic, a driver can also delete his offer. In this case, the number of available seats is set to 0 and if there are bookings, they will be canceled automatically by the smart contract, and owners are notified. The carpooling platform also proposes a carpooling proof. The proof is generated automatically based on transactions stored in the blockchain. To recapitulate, the blockchain platform manages Offer, Transaction (booking, confirmation/rejection, booking_cancelation ), and Proof entities.

### 2.2. Privacy requirements

In this paper, we address the privacy requirements imposed by carpooling operators within the blockchain-based carpooling platform. Privacy is considered in two levels; end-users and carpooling operators. From end-users point of view, it's worth noting that the blockchain does not manage end-users accounts. However, some end-users information is shared with other operators. These pieces of information are stored with offers and transactions and are used to facilitate linkage between driver and passengers (e.g. username, userId). Even if shared information is RGPD compliant, it is possible to track end-users mobility. For example, a malicious node can know that a person called Alice travels every day from a specific origin to a specific destination. From operators point of view, privacy issue concerns the disclosure of operators business models. It's possible that operator A checks, in its local node, how many offers operator B has and how much money it earns.

To improve confidentiality, techniques need to be applied at the blockchain level so that members cannot observe the content of stored entities, but also cannot link which operator performed which transaction on which entity. To improve control, a secure exchange protocol must be set up so that data can be shared between members.

## 3. Privacy solutions : Litterature survey

We provide in this section a state of the art of cryptographic technologies used to enhance privacy in blockchain or distributed systems in general. We list below the main techniques and protocols that may be relevant in our case.

### 3.1. Anonymous signatures

Anonymous signatures are schemes where signatures do not reveal the signer's identity, as long as some parts of the message are unknown. The two most important and typical anonymous signatures schemes for Blockchains are Group Signature and Ring Signature [3]. Group Signature is a cryptography scheme allowing any member of a group to sign a message for the entire group anonymously by using his personal secret key, and any member with the group's public key can check and validate the generated signature. This process reveals nothing about the true identity of the signer except the membership of the group. Due to the group manager who manages adding group members, handling events of disputes, etc., group signature is suitable for consortium blockchain. The ring signature scheme is anonymous, it is difficult to determine which member of the group uses his/her key to sign the message. It differs

from group signatures because there is no group manager, so the real identity of a signer cannot be revealed in the event of dispute. Also, any users can group a "ring" by themselves without additional setup, so it is applicable to public blockchain.

### 3.2. Homomorphic encryption

It's is a powerful cryptography method that can perform certain types of computations directly on ciphertext and ensure that the operations performed on the encrypted data will generate identical results to those performed by the same operations on the plaintext. The use of homomorphic encryption technique offers privacy protection, and allow ready access to encrypted data. It enables private queries to a search engine, searching on encrypted data and improves the efficiency of secure multiparty computation [4]. It helps to protect the integrity of data by allowing others to manipulate its encrypted form while no one can understand or access its decrypted values. There are three main types of homomorphic encryption; (1) Partially Homomorphic Encryption that allows only one operation to be performed an unlimited number of times on the ciphertext, (2) Somewhat Homomorphic Encryption that allows limited operations to be performed a set number of times and (3) Fully Homomorphic Encryption that allows efficiently computable functions any number of times [5]. Authors in [6] propose a framework for Secure Data Sharing based on homomorphic encryption and proxy re-encryption schemes.

### 3.3. Privacy Preserving Record Linkage (PPRL)

Privacy-preserving record linkage (PPRL) aims to identify and link records that correspond to the same real-world entity across several data sources held by different parties without revealing any sensitive information about these entities. It consists of two main phases, Privacy Preserving Matching (PPM) and Privacy Preserving Blocking (PPB) [7]. Privacy Preserving Matching is the process of accurately matching datasets using elaborate algorithms that do not compromise data privacy. Since these algorithms are elaborate, they manage to achieve high matching performance but they also exhibit low time performance. Time performance is an important aspect of the problem since we usually need to integrate large volumes of data. As such, Privacy Preserving Blocking techniques have been introduced which aim at reducing Privacy Preserving Matching times. Some studies proposed tools to implement PPRL solution. In [8], Primat is proposed. It's a toolbox for fast privacy-preserving matching that covers the whole PPRL life-cycle and improves applicability by providing various components.

### 3.4. Proxy Re-Encryption (PRE)

Proxy re-encryption is a special type of public-key encryption that permits a proxy to transform ciphertexts from one public key to another, without enabling the proxy to learn any information about the original message. Thus, it serves as a means for delegating decryption rights, opening up many possible applications that require delegated access to encrypted data. In a PRE scheme, three parties are involved [9]; the delegator who is the one that delegates his decryption rights. To do this, he needs to create a re-encryption key that he sends to the proxy. The delegatee is the one that is granted a delegated right to decrypt ciphertext that was re-encrypted for him. And the proxy which handles the re-encryption process that transforms ciphertexts under the delegator's public key into ciphertexts that the delegate can decrypt using his private key. A PRE system has different properties that may or not be desired depending on the goals of the implementation [10]. There are also many different proxy re-encryption schemes such as type-based, key private [11], identity-based [12], attribute-based, conditional [13], time-based, threshold-based [9], secure certificateless [14] and much others [15].

### 3.5. Hyperledger-based solutions

The carpooling platform is implemented using Hyperledger Fabric [16]. Hyperledger organization provides various aspects to achieve data privacy. Privacy can be enhanced by segregating the network into channels, where each channel represents a subset of participants that are authorized to see the data for the chaincodes that are deployed to that channel. Channels are useful in cases where a subgroup of the blockchain network's participants have a lot of transactions in common and these transactions can be processed with no dependency on state controlled by entities outside this group [17]. Private data are also used to keep ledger private from other organizations on the channel. Also called private transactions. Private transactions offer transaction privacy at a more fine-grained level than channels. The database storing the private data is updated alongside the public ledger as transactions containing references to private data are committed [17].

### 3.6. Privacy techniques comparison

In this subsection, we compare different techniques presented above and we try to make a choice that can fit our use case and our requirements. We remind that our objective is to share offers between carpooling operators while hiding private information to avoid the possibility to link an offer, a transaction, or a proof to a specific carpooling operator or a specific end-user. We assume that the data will be stored encrypted in the blockchain. The solution must meet our needs, such as being able to keep offers searchable and shared across the entire network.

The presented methods can be classified into three categories; (1) methods used in the authentication level to manage access to the blockchain such as anonymous signatures, (2) techniques focusing on allowing the search of specific information in an encrypted content called Searchable encryption [18] such as Homomorphic encryption and PPRL,

and (3) methods interested in only protecting private data called Privacy enhancing technologies such as PRE.

In this study, we exclude including privacy at the authentication level since our Blockchain is private and different operators agree to share their offers and create transactions related to them. Thus, the anonymous signatures or the Hyperledger Fabric methods concerning authentication, such as Identity Mixer or Zero-Knowledge Asset Transfert (ZKAT) are left for future improvement.

Searchable encryption techniques are used when we need to perform some data analyses on encrypted data (homomorphic encryption) and the result of the process will not reveal any information about the data. It's also used when we want to check if a content exists, so the response will be yes or no, or also to match an encrypted data with another without disclosing any identity (PPRL). These searchable encryption techniques present a relevant choice. However, they are not sufficient in our case since we need to search and retrieve a non-encrypted data from a dataset of encrypted data. It is necessary to implement an exchange protocol or to use a complementary method, such as proxy re-encryption.

Concerning Hyperledger Fabric privacy techniques, they don't address our target. Channels are not relevant here because it would require the creation of a channel for each pair of operators, and this does not resolve our problem. A malicious operator can still compute and analyze the gain of each operator using different channels. This technique will just complexify our search process. For private data, the purpose is to protect access to certain data between certain organizations. In our case, the aim is that the data is potentially accessible to everyone, which defeats our purpose. This would require that all data from each operator is isolated in its local node and a distributed search is performed to retrieve offers from each operator. This solution is not scalable and the response time increases while increasing the number of nodes (operators). This could negatively affect the performance of the network. It also requires and intervention when a new operator is connected to the carpooling platform in order to add his address within the distributed requests list.

We choose the proxy re-encryption technique for a first version to enhance privacy in the carpooling platform. This method allows storing encrypted data in the blockchain while allowing the owner to delegate decryption rights to other members and the proxy entity manages all the interactions. We detail, the PRE-based solution in the following section.

# 4. PRE based blockchain scheme: architecture and implementation

We present in this section the architecture of our proposed scheme as well as the technological choices used in the implementation.

## 4.1. Architecture

In Figure 1, we present an overview of the architecture of our carpooling platform. The carpooling solution is a web application provided for all partners (operators), where they interact and communicate through the blockchain. We detail from top to bottom different blocks.

- **Carpooling operator Services** The first layer, presented in blue, includes different operators' services used by end-users via each operator web application. This layer is specific for each operator and presents the already used services without the interoperability platform. These services allow passengers and drivers to propose a carpooling offer, search for a carpooling offer, exchange messages, confirm or refuse a booking. They provide also access control services to manage their end-users.
- **Carpooling Platform API Services** The second layer, presented in orange, includes the carpooling platform services introduced to provide interoperability between operators. This layer provides services to carpooling operators such as pushing or retrieving an offer, a booking (called transaction here) or a proof in the blockchain, and uses notification service to notify different carpooling operators when they are concerned. To ensure interoperability, this layer defines governance rules in the blockchain. The Key Management System (KMS) service is added to address the privacy requirement and it is responsible for generating different used keys for our cryptographic solution. The carpooling platform is based on a REST API server written in NodeJS. This service layer is mainly used by operators to interface with the blockchain. All the actions to register an operator, retrieve or store data, as well as generate keys are launched via the carpooling platform API using simple REST APIs.
- **Chaincodes** Then, we have the blockchain responsible for managing smart-contracts, storing and retrieving offers, transactions and proofs, and managing keys used for privacy. Smart contracts Offer, Transaction and Proof are also responsible for encrypting and decrypting entities. Chaincodes are developed in Typescript (NodeJS) with version 1.4 of the Fabric Shim API.
- **Hyperledger Fabric Blockchain** The blockchain network used is Hyperledger Fabric version 1.4. The network contains two organizations, each containing two peers with their couchDB database, a fabric certificate authority server, and a fabric client. Carpooling platform services use the Hyperledger Fabric Client version 1.4 to interact with Peers of the Fabric network and to send transaction invocations or perform chaincode queries. It also uses the Hyperledger Fabric-CA Client version 1.4 to register operators with operator name as parameter.
- **Proxy** Finally, the proxy entity is added with the PRE scheme. The proxy is a blockchain node

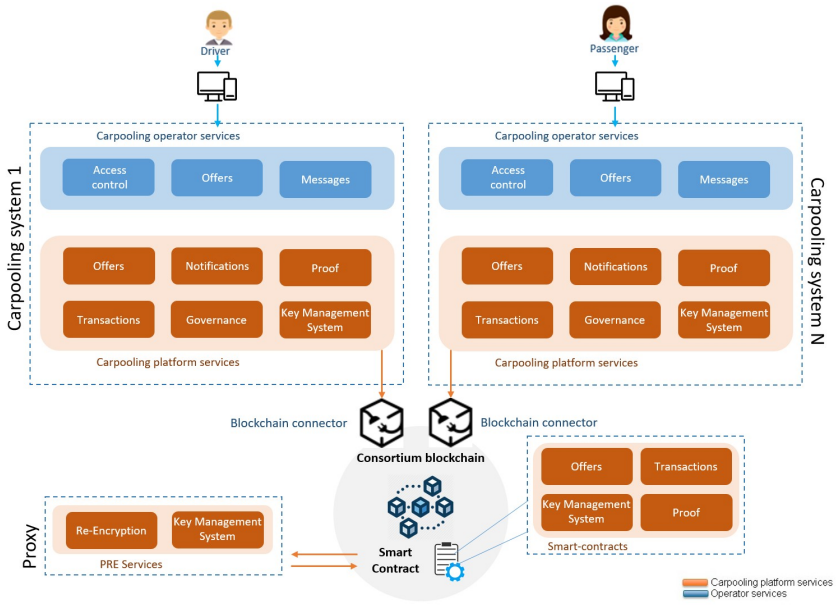


Figure 1. Architecture

that implements smart contracts for managing re-encryption keys and re-encrypting data. The proxy is considered as a trusted third party.

## 4.2. How it works?

The proxy re-encryption scheme is implemented using the Skycryptor Javascript SDK [19]. The Skycryptor library is composed of four functions; keys generation, encryption, decryption, and re-encryption. There are many different implementations of the proxy re-encryption scheme. Some implementations use the concept of encapsulation and de-capsulation rather than encryption and decryption [20]. The Skycryptor library is one of them. These implementations use encapsulation, decapsulation, and re-encapsulation terms to describe the implementation, however, they can always be considered as encryption, decryption, and re-encryption as in literature. These terms are equivalent.

- **Keys generation**
  The proxy re-encryption process needs private and public keys of different operators, the private and public keys of the proxy, as well as the re-encryption keys.
  The public and private keys, of different operators, used for encryption and decryption, are generated for each operator off-chain. The generation of these keys is triggered by each operator in its local node, by calling an API of the KMS service in the Carpooling Platform Services. They are thus generated off-chain and sent to the KMS chaincode via the transient field, this field is excluded from the channel transaction. These keys are stored using a feature in Hyperledger Fabric called Private Data, which allows data to be isolated between organizations. Public keys are stored in a collection that includes all organizations, while private keys are stored in a collection that is accessible only by the organization that owns the key and are never shared.
  Re-encryption keys are also generated off-chain. The non-interactive property of the proxy system allows operator A, who wants to delegate its re-encryption right to a specific operator B, to generate a re-encryption key without the intervention of Operator B. The generation of the re-encryption key of the couple (Operator A, Operator B) is then launched by Operator A, which is the delegator, using the Carpooling Platform KMS service. This service will invoke the chaincode KMS which will generate the re-encryption key using the public key of operator B (delegatee) and the secret key of Operator A. The generated key is called re-encryption key. This key is not stored in the Blockchain. It is sent to the Proxy using PRE services while identifying which operator is the delegator and which is the delegatee. The proxy stores the key in a database with an id that is generated with the couple (idOperatorDelegator, idOperatorDelegatee). We use MongoDB as a database. It is worth to note that if the same couple switches roles of delegator and delegatee (Operator B, Operator A), another re-encryption key is generated for that.
  In addition to these keys, the Proxy has its RSA public and private key pair. These are generated by the proxy off-chain. The proxy public key must be distributed to all organisations (operators). In this solution, this is done before the installation and instantiation of the chaincodes. Thus, the URL and

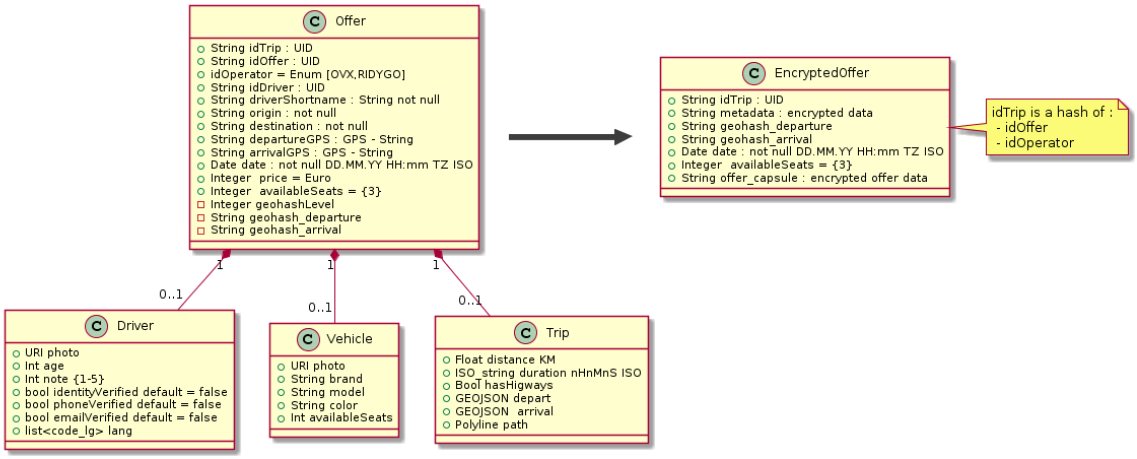


Figure 2. Encrypted offer entity

the public key of the PRE services (proxy) are hard-coded into the chaincodes. We explain later when proxy keys are used.

- **Encryption**
  The encryption function is integrated into all chaincodes that manage entities (Offer, Transaction, Proof). The encryption is done before storing any entity in the Blockchain. We present in Figure 2 offer entity before and after encryption. As we can see, only sensible fields are encrypted and stored in the field *encrypted_entity*, some are left unencrypted in order to be able to perform rich queries against data values (Origin, destination, date, and availableSeats). Since idOperator should be known by the proxy in order to choose which re-encryption key to use, this field is encrypted using the public key of the proxy and stored in *metadata* field. To encrypt the metadata, we use the RSA encryption algorithm.
  In Skycryptor implementation, entities are not encrypted directly with the owner's public key, but with a symmetric key randomly generated and stored in a capsule. Putting this symmetric key, used to decrypt the data, in the capsule is then called encapsulation. As a consequence, to create an entity, the entity chaincode calls the KMS chaincode to retrieve the public key of the operator invoking the request to be used to generate the *capsule*. The capsule is stored as a field in the entity.
- **Re-encryption**
  When an operator requests a data stored in the Blockchain and belonging to another operator, this encrypted entity is sent to the proxy. In PRE services, the capsule as well as the metadata are retrieved from the encrypted data. The proxy extracts the idOperator owning the data from the metadata using its secret key and chooses the corresponding re-encryption key. Then, it re-encapsulates the capsule using the re-encryption key in order to be decapsulated by the operator which requested the data. The new capsule is stored in the encrypted data and sent back to the chaincode managing the entity.
- **Decryption**
  The chaincode decapsulates the capsule using the private key of the operator invoking the request and extracts the symmetric key to decrypt the data. Finally, decrypted data is sent to the operator.

### 4.3. Process view

For better understanding, we explain in Figure 3 the process with an example. The example details the Offer entity starting from a driver who proposes an offer to a passenger who retrieves this offer.

1) A driver of Operator 1 proposes a carpooling offer and the offer entity is sent to the carpooling platform services.
2) The carpooling platform services formats the entity (Left entity in Figure 2 ) and sends it to Offers chaincode.
3) The chaincode encrypts the offer entity with the public key of Operator 1 and stores it in the blockchain (Right entity in Figure 2).
4) A passenger of Operator 2 searches for an offer and Operator2 sends the request to the carpooling platform services.
5) The carpooling services formats the request and sends it to Offers chaincode.
6) The encrypted offer entity is retrieved from the blockchain.
7) The encrypted data is sent to the proxy and sent back as re-encrypted data to the chaincode.
8) The chaincode in Operator 2 node decrypts the offer entity with Operator 2 private key.
9) The decrypted offer entity is sent back to the carpooling platform services.
10) The carpooling platform services formats the offer and send it to Operator 2.

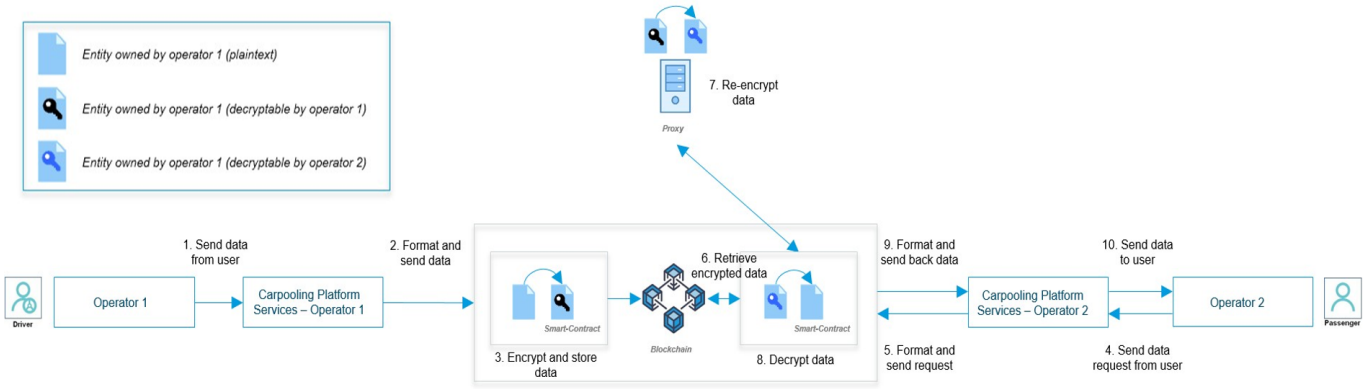


Figure 3. Proxy re-encryption scenario: Offer entity

## 5. Experiments and discussions

Our experiments took place on a machine whose OS is Ubuntu 16.04, with 2 VCPUs and 8 GB RAM. All components, i.e. the blockchain network, carpooling platform, and PRE services are deployed locally using Docker. Operators' services and applications are deployed in partner servers.

The first version of the carpooling platform, without the integration of privacy enhancement, required a timeout transaction of 5000 milliseconds. With the re-encryption proxy solution, this constraint is respected. A test should be carried out on a large number of carpooling offers to see what impact this solution has in terms of scalability.

The proof of concept validated the feasibility of the proposed architecture. Partner operators have tested this solution and were satisfied by the level of privacy provided by the platform as well as the response time. They didn't notice any increase in response time compared to the first version.

For future works, we identified some improvements. An improvement of the proxy here could be the use of external chaincode, a new feature of Hyperledger Fabric 2.0, instead of a rest HTTP server used here in Proxy. In this way, requests made to re-encrypt entities will be exchanged directly between chaincodes in a more secure manner. At the time of writing this paper, chaincodes are only proposed with Go language. Skycryptor library is also proposed in Go language. Since the first version of our carpooling platform was developed in Typescript, we choose to continue with this language. We are currently studying the integration of external chaincodes.

Another improvement could be imagined in the encryption step when generating the capsule and the metadata. These encryptions produce non-deterministic results. In our case, this is not an issue given our current endorsement policy that allows each organization to store entities without being validated by other ones. However, if this policy changes, the results of different chaincodes will produce different encrypted entities and the transaction couldn't be validated.

## 6. Conclusion

In this paper, we proposed a PRE-based Blockchain with the aim to ensure the confidentiality of shared information between organizations in a trusted way. This solution could be used in different use cases. In this study, we detail it in the context of carpooling.

To recapitulate, we propose an architecture that allows sharing entities in the blockchain while preventing from analyzing end-users mobility behavior and operators' business KPI. In fact, shared information becomes unusable if it is retrieved directly from local nodes. Entities could only be accessed via APIs on which we apply access controls. We used the proxy re-encryption scheme to address this issue and we proved that it's a suitable solution. It's worth to keep in mind that the PRE scheme imposes by design to keep the owner of an encrypted entity known in order to choose the right re-encryption key. However, this was problematic in our case because among our objectives is to also hide operator identities to avoid KPI computations. Thanks to the encryption of the idOperator (here called metadata), the encrypted entities in the blockchain reveal nothing about their owners, except for the proxy.

## References

[1] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An overview of blockchain technology: Architecture, consensus, and future trends," *2017 IEEE International Congress on Big Data (BigData Congress)*, pp. 557–564, 2017.

[2] J. Cheng, N. Lee, C. Chi, and Y. Chen, "Blockchain and smart contract for digital certificate," in *2018 IEEE International Conference on Applied System Invention (ICASI)*, 2018, pp. 1046–1051.

[3] R. Zhang, R. Xue, and L. Liu, "Security and privacy on blockchain," *CoRR*, vol. abs/1903.07602, 2019. [Online]. Available: http://arxiv.org/abs/1903.07602

[4] C. Gentry, "A fully homomorphic encryption scheme," Ph.D. dissertation, Stanford, CA, USA, 2009.

[5] P. Martins, L. Sousa, and A. Mariano, "A survey on fully homomorphic encryption: An engineering perspective," *ACM Computing Surveys*, vol. 50, pp. 1–33, 12 2017.

[6] B. K. Samanthula, G. Howser, Y. Elmehdwi, and S. Madria, "An efficient and secure data sharing framework using homomorphic encryption in the cloud," 08 2012.

[7] D. Vatsalan, P. Christen, and V. S. Verykios, "A taxonomy of privacy-preserving record linkage techniques," *Information Systems*, vol. 38, no. 6, pp. 946 – 969, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0306437912001470

[8] M. Franke, Z. Sehili, and E. Rahm, "Primat: a toolbox for fast privacy-preserving matching," *Proceedings of the VLDB Endowment*, vol. 12, pp. 1826–1829, 08 2019.

[9] D. Nuñez, I. Agudo, and J. Lopez, "Proxy re-encryption: Analysis of constructions and its application to secure access delegation," *Journal of Network and Computer Applications*, vol. 87, pp. 193 – 209, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1084804517301078

[10] G. Ateniese, K. Fu, M. Green, and S. Hohenberger, "Improved proxy re-encryption schemes with applications to secure distributed storage." vol. 2005, 01 2005.

[11] G. Ateniese, K. Benson, and S. Hohenberger, "Key-private proxy re-encryption," vol. 2008, 04 2009, pp. 279–294.

[12] M. Green and G. Ateniese, "Identity-based proxy re-encryption," in *Applied Cryptography and Network Security*, J. Katz and M. Yung, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 288–306.

[13] P. Wang, "Identity-based multi-condition proxy re-encryption," 01 2016.

[14] V. Kuchta, G. Sharma, R. Anand, T. Bhatia, and O. Markowitch, "Secure certificateless proxy re-encryption without pairing," 07 2017, pp. 85–101.

[15] A. Khurshid, G. Fiaz, and A. Khan, "A comparison of proxy re-encryption schemes – a survey," *International Journal of Computer Science and Information Security (IJCSIS)*, vol. 14, pp. 392–397, 07 2016.

[16] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. D. Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, S. Muralidharan, C. Murthy, B. Nguyen, M. Sethi, G. Singh, K. Smith, A. Sorniotti, C. Stathakopoulou, M. Vukolic, S. W. Cocco, and J. Yellick, "Hyperledger fabric: A distributed operating system for permissioned blockchains," *CoRR*, vol. abs/1801.10228, 2018. [Online]. Available: http://arxiv.org/abs/1801.10228

[17] "A blockchain platform for the enterprise," https://hyperledger-fabric.readthedocs.io: https://hyperledger-fabric.readthedocs.io/en/release-2.0/private-data/private-data.html, accessed: 2020-01-29.

[18] Dawn Xiaoding Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proceeding 2000 IEEE Symposium on Security and Privacy. S P 2000*, 2000, pp. 44–55.

[19] "Javascript skycryptor sdk," https://github.com/skycryptor/js_proxy, accessed: 2020-07-02.

[20] D. Nunez, "Umbral: A threshold proxy re-encryption scheme," 2018.