2018 IEEE Confs on Internet of Things, Green Computing and Communications, Cyber, Physical and Social Computing, Smart Data, Blockchain, Computer and Information Technology, Congress on Cybermatics

Hybrid Cryptographic Protocol for Secure Vehicle Data Sharing over a Consortium Blockchain

Kei Leo Brousmiche*, Antoine Durand*, Thomas Heno*†, Christian Poulain*†,
Antoine Dalmieres*‡, Elyes Ben Hamida*
*IRT SystemX, Paris-Saclay, France, Email: {firstname.name}@irt-systemx.fr
†Groupe PSA, Route de Gisy, 78140 Vélizy-Villacoublay, Email: {firstname.name}@mpsa.com
‡Covéa, 86-90 rue Saint Lazare 75009 Paris, France, Email: adalmieres@gmf.fr

***Abstract*—The blockchain technology has recently attracted increasing interests in a wide range of use-cases. Among those, the management of vehicles' data and life cycle over a blockchain has sparked various research initiatives on a global scale, with the promise to prevent automobile frauds and to enable more collaborations between the involved stakeholders. In this paper, we investigate the problem of securing and sharing vehicles' data over a consortium blockchain, and we describe the architecture of the implemented proof-of-concept. Then, we introduce a novel hybrid cryptographic protocol to secure the access to vehicles' data between the involved stakeholders. Finally, we discuss the lessons learned acquired from the preliminary trials and we highlight the future research challenges and opportunities.**

I. Introduction

Introduced by an anonymous contributor under the alias of Satoshi Nakamoto a decade ago [1], the blockchain technology has recently gained a lot of interests from a variety of sectors such as government, finance, industry, health or research.

More recently, the introduction of smart contract has extended functionalities by offering the possibility to execute computational programs over the network and store any kind of information. Today, blockchain applications field goes far beyond cryptocurrency, its initial purpose, and can be applied to various use cases. Among those, the management of vehicles' data over blockchain has sparked various research initiatives on a global scale [2], [3], [4], [5], with the promise to prevent automobile frauds and to provide more collaborations and transparency between the involved stakeholders, including automotive manufacturers, insurance companies, repair shops, car dealerships, etc.

Indeed, despite the introduction of digital odometers within vehicles, mileage fraud affects currently over 30% of the used cars that are sold in Europe, costing the European consumers approximately 5.6€ to 9.6€ billion per year [6]. Another example of vehicles frauds, that has great economic and societal impacts, is related to the *rolling wrecks*, *i.e.* damaged vehicles that are not allowed to go on the roads, but are put back onto the second-hand car market, with the help of corrupt professionals. These *rolling wrecks* are often the cause of severe accidents, involving injured people, and costing insurance companies and consumers a lot of money.

Nowadays, the primary way to keep track of vehicle maintenance and mileage information are paper based car log books and invoices. However, these can be incomplete, corrupted or missing, and cannot prevent the aforementionned frauds either. In this context, various initiatives ([2], [3], [4], [5]) have been recently launched to investigate the potential of the blockchain technology in securing vehicles information over a tamper-proof and decentralized ledger. However, these projects are still at an early stage of development, and many research challenges still need to be addressed, including the privacy and the security of the vehicles data that being written on the blockchain.

In this paper, we investigate the potential of the blockchain technology in securing and tracking vehicles maintenance and mileage informations over a consortium blockchain, that is governed by a set of key stakeholders, including an automobile manufacturer, an insurance company and a service provider. Our contributions are many-fold. First, we design a novel architecture to enable the management of vehicles life-cycles and data over a decentralized and tamper-proof ledger, hence enabling collaborations and data sharing between the involved parties. For example, vehicles mileages, that are written on the blockchain, can enable a vehicle owner to certify the actual value of his car; while it can also enable an insurance company to provide usage-based insurance services (e.g. pay how/as you drive). Second, we design a novel hybrid cryptographic protocol that address the limitations of existing blockchain technologies and enable secure and private data sharing between the involved parties. Finally, the discussed the lessons learned from the preliminary trials and we draw various research challenges and opportunities.

II. Related Works

A. Vehicle life-cycle over blockchain

We present in this section some related works that have been recently done in the context of vehicle life-cycle and data management over blockchain.

The VINChain project [2] is a blockchain based solution that can be seen as a decentralized database recording relevant information pertaining to vehicles. For each vehicle, a "passport" tied to the vehicle identification number (VIN) is stored on the ledger. Information corresponding to a specific VIN number can be searched, selected, and pulled into a report on the blockchain in exchange of VIN tokens, the currency of

978-1-5386-7975-3/18/$31.00 ©2018 IEEE
DOI 10.1109/Cybermatics_2018.2018.00223

the solution. At the time of this writing, this project is in development according to their road-map. Moreover, the main limitation of this solution is the lack of parties/members that supports the blockchain.

Indeed, we believe that a blockchain based architecture is meaningful when multiple members have to share information and work-flows in a trust-less environment. With only one member governing the solution (i.e. VINCHain), the use of blockchain is not entirely justified and a traditional centralized architecture could be considered instead. [5] and [4] are two other similar projects led by major industrial groups that share the same limiation.

CarVertical [3] is yet another blockchain based solution to store car history registry at the stage of white-paper. As for the members participating to the blockchain, the project suggests that some governmental institutions could take part to the solution as validators. However, the project does not address the problem of data access management within the consortium. Any members participating to the blockchain has equal right to access all the data.

In our solution, due to the presence of three antagonistic entities that want to limit the visibility of some sensitive information, there is a need for an on-chain security and access management protocol.

B. Data privacy over blockchain

While blockchain technologies were initially designed following a philosophy of openness where anyone could contribute, use or audit the system, new applications require additional features that were not originally supported such as volume scalability, system responsiveness or data privacy. From these needs have emerged a variety of technologies categorized as consortium blockchains with restricted network access, lighter and faster consensus mechanism and other specific features. Among them, one interesting feature is data privacy: some technologies enables to limit the access to private data published on the blockchain to a set of authorized members.

Quorum [7], a technology based on Ethereum, supports private transactions by the use of asymmetric cryptographic functions and parallel private ledgers. In this system, each Ethereum node embedding the public ledger, is locally bound to a Constellation node with a private ledger. However, private data cannot be disclosed to new members: the list of nodes that (will) have access to a private transaction is static. This limitation prevents for instance the update of data access rules or the audition of previous private transaction by a third party (e.g. a governmental institution).

HyperLedger Fabric [8] also proposes private transactions through the use of *channels* (i.e. partitions of blockchain with predefined participating members). Moreover, the ability to dynamically add new members to a channel has been introduced recently, although it requires a complex sequence of 10 steps [9]. However, in order to revoke the right of a member, it would require to instantiate a new channel that exclude the member, copy the history of the former channel and update the channel information in all remaining members systems. In other words, Fabric is not designed to support dynamic channel membership.

In this paper, we propose a hybrid cryptographic protocol over Ethereum based blockchain that enables the dynamic access management to smart contract fields. This algorithm and its implementation are described in the section V.

III. Use case overview

The present research work is conducted within the scope of the *Blockchain for Smart Transactions* (BST) research project [10] that has been recently launched by IRT SystemX in partnerships with key industrial and academic partners, and whose main objective is to unlock the potential of blockchain-based architectures by developing new usages based on human-centric data empowerment.

The first case study that has been addressed by the project was sparked by the need for innovative and viable technologies to address the challenges that are currently affecting the automotive industry, mainly: 1) the lack of collaborations and data sharing between the involved stakeholders; and 2) the lack of transparency and trust on used-car markets due to various types of vehicles frauds (e.g. odometer frauds, wrecked or salvaged vehicles for sale, etc.)

In this context, we propose an innovative blockchain-backed vehicles data and life-cycle ledger to digitize the vehicles life-cycle over a consortium blockchain. The initial stage of the developed Proof-of-Concept includes already major actors from the industry, including: 1) *PSA Group*: a leading French automotive manufacturer; 2) Covéa: a leading French mutual insurance company; and 3) Docapost: a leading provider of document management services (*i.e.* invoices, certificates, etc.). The integration of additional members is underway.

As shown in Figure 1, the benefits of our proposed framework is highlighted across five main steps of a typical vehicle life-cycle. During *step one*, when a consumer acquires a new vehicle, the automotive manufacturer (*e.g.* PSA Group) creates a new digital and secure car book on the consortium blockchain, and whose access is initially restricted to only the actual car owner and the automotive manufacturer, including the members of its dealership network (*e.g.* official repair shops). Eventually, the car owner may grant an access to the secure car book to his insurance company (*e.g.* Covéa).

During *step two*, as modern vehicles already include sensing and communication capabilities, the current mileage of the vehicle is automatically transmitted to the carmaker's cloud infrastructure at regular intervals, and is written on the secure car book, at a reasonable frequency to attest the value of the vehicle.

During *step three*, the car owner may service or repair his vehicle within or outside the dealership network. In the first case, since the automotive manufacturer is already a member of the consortium blockchain, and have already access to the secure car books, all vehicles operations and data (*e.g.* repair, maintenance, invoices, etc.) can thus be recorded on the blockchain. In the second case, a consumer might also decide

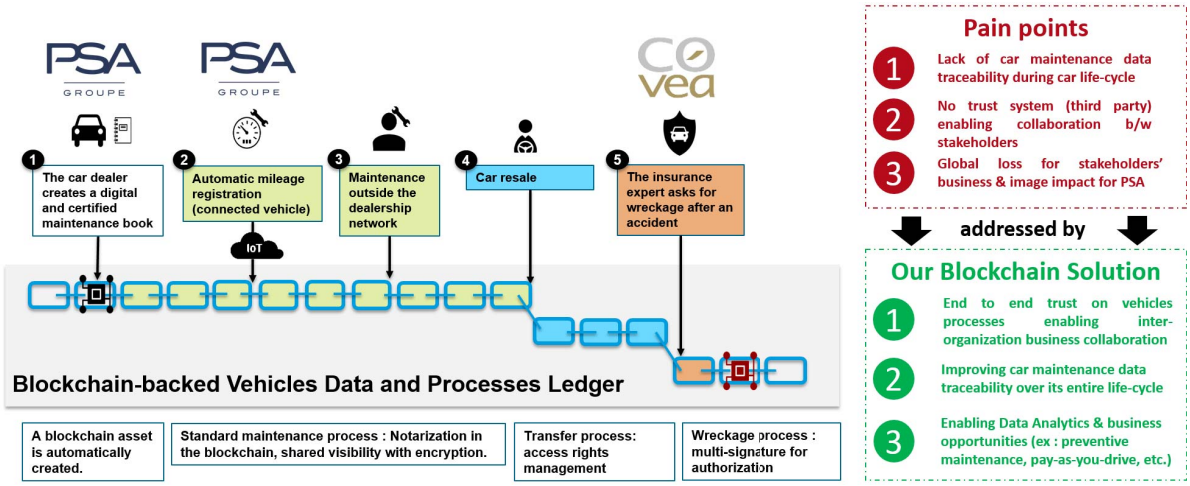

Fig. 1. Digitizing Vehicles Life-cycle over a Consortium Blockchain: Pain Points and Benefits of our Framework.

to select an *independent repair shop*, who is not an actual member of the consortium blockchain, and hence grant him a temporary access to his secure car book. This independent repair shop will hence have access to the complete data history of vehicle and will be able to record new information and operations through the APIs that are exposed by the actual members of the consortium. Once the operations are done, the access to the secure car book is revoked, and the independent repair shop will not have access to any future information.

During *step four*, the car owner may decide to sell his vehicle on a used-car market. The interest that can be found in having this secure car book is then obvious. The current owner can prove the good maintenance state of his vehicle and certify its mileage and actual value, increasing thus transparency and trust with the potential buyers. We believe that this can help in minimizing the problem of odometer frauds and related costs. In this case, the current vehicle owner will transfer the ownership of his secure car book to the new owner.

Finally, during *step five*, in case of accidents, the insurance company, who has already access to the secure car books of its customers, will be able to better evaluate the damages on the vehicle and provide adequate compensations. The result of the insurance expertise will be permanently recorded to the secure car book. Depending on the severity of the damages, the insurance expertise could allow the vehicle to go again on the roads, or could declare the vehicle as a wreck. In this later case, the owner will thus transfer the property of his vehicle to his insurance company, who will take care of destroying the vehicle, limiting thus the frauds related to rolling wrecks.

IV. System and Blockchain Architecture

A. General Architecture

The architecture of the solution is fully described in our previous work [11]. The system is composed of multiple service components.

a) Shared Services: The consortium support a common infrastructure in order to share some core services: 1) A Quorum based Blockchain that stores car maintenance log books, pseudonymized accounts information and their associated access rules. It is a consortium blockchain, fork of Ethereum, that validate blocks using Quorum Consensus [12] which does not requires mining.
2)An identity Database that centralizes accounts information to authenticate users. It stores accounts names, passwords hashes, blockchain keys and identifiers.

b) Entities Services: Each consortium member supports a server with a common architecture which offers web-services that interact with the blockchain through multiple access control layers: a traditional user authentication mechanism based on the identity database and access tokens, smart contract modifiers and an on-chain reading access using a novel cryptographic protocol (see sub-section V). All the services are based on the car maintenance log books (i.e. carbook) that is stored and managed on-chain in the form of smart contracts. We do not describe each services in this paper, for more information please refer to [11]. In the next section, we describe the carbook smart contract.

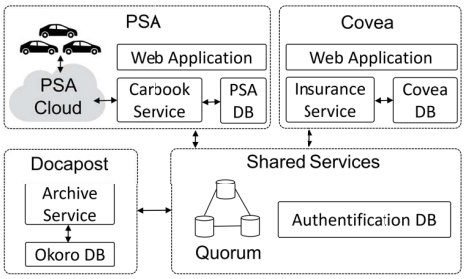

Fig. 2. General architecture

B. Car maintenance log book smart contract

In this section we describe some specific features of the *CarBook* smart contract contract. The link between the *CarBook* smart contract and its corresponding car is done through the hash of the car's VIN (*i.e. Vehicle Identification Number*) written in a field of the contract. The use of the hash enables to anonymize the car identity over the blockchain. Similarly, the ownership of a car book (*i.e.* the link between a car owner and the smart contract) is based on the blockchain address of the owner that is generated at the time the account is created. Those two information are given by the car maker at the initialization of the contract. Using them, we are able to control access according to the known addresses: *e.g.* functions limited to the owner, to the car maker, to the insurer if the owner subscribed to an insurance service. The format and content of new *records* is checked by the smart contract before storing them. For instance, the mileage of the record is checked so that it is not inferior to the last known value. To summarize, the usage of Smart Contracts for implementing *CarBooks* and its related functionalities guarantees the enforcement and sharing of common data model, processes and protocols over the consortium blockchain.

V. Hybrid Cryptographic Protocol

In order to handle data confidentiality of the $records$' labels in CarBook contracts, we design and implement a hybrid cryptographic protocol that enables to add and remove read-authorized users dynamically. Such a functionality is useful in cases where the owner of a given vehicle wants to grant to an external user (e.g. repair shop) a temporary access to his CarBook.

a) Definitions: We use a symmetric encryption scheme S and an asymmetric encryption scheme A, instantiated with appropriate security parameters. Those schemes provide the usual $generate$, $encrypt$ and $decryption$ primitives, as well as $sign$ and $verify$ for A. Note that A is actually a sign-cryption scheme, or alternatively, made of two independent public key encryption and digital signature schemes. For readability we use a unified notation, but care should be taken if implemented as two separate scheme, especially regarding key management: A key pair for A would actually be made of two key pairs, one for encryption and one for signing. For the sake of simplicity, we assume that $verify$ takes $sign$'s output and returns the message that has been signed if the signature is valid and throws an error otherwise. Each entity participating to the protocol is identified by its blockchain address. Moreover, each entity generates its asymmetric key pair using $A.generate$. A smart contract manages the following data structures in its state:

- $channel$, a list of encrypted data chunks.
- $keystore$, a map from blockchain addresses to encrypted keys.

Its owner (the writer) is identified by his blockchain address Ad^w and a corresponding secret credential $Cred^w$ and key pair (P_k^s, S_k^w). We note I the set of participants, i.e. the channel writer and all the readers.

The smart contract implements few functions required for the protocol. The two following procedures are reserved to the writer. They modify state, and needs further access control. Thus, they can only be called through transactions issued to the smart contract by the owner, which is semantically enforced by requiring $Cred^w$. This allows abstraction of the blockchain authentication process.

- $append^{Cred^w}(secret)$: appends the encrypted data $secret$ at the end of the list $channel$.
- $addEncKey^{Cred^w}(encKey, Ad)$ adds an encrypted key to $keystore$ with its associated blockchain address Ad.

The following functions are public getters, to access the data structures. Those function can be called locally without interacting with other peers.

- $retrieve(i)$ returns the encrypted data with index i in $channel$.
- $getKey(Ad)$ returns the encrypted key corresponding to the blockchain address Ad from $keystore$.

b) Writing protocol: enables the writer, given his public key P_k^w and blockchain address Ad^w, to encrypt and then append a data into $channel$ on the blockchain, as shown in Algorithm 1.

Algorithm 1 Writing Protocol

function PUSH(P_k^w, S_k^w, Ad^w, $Cred^w$, $data$)
  $k_{sym} \leftarrow getKey(Ad^w)$
  if $k_{sym} == \perp$ **then**
    $k_{sym} \leftarrow S.generate()$
    $addEncKey^{Cred^w}(A.encrypt(k_{sym}, P_k^w), Ad^w)$
  else
    $k_{sym} \leftarrow A.decrypt(k_{sym}, S_k^w)$
  end if
  $C \leftarrow S.encrypt(A.sign(data, S_k^w), k_{sym})$
  $append^{Cred^w}(C)$
end function

c) Sharing protocol: enables the writer, given his public and private keys P_k^w S_k^w, to let another entity (i.e. a reader), with its blockchain address Ad^r and public key P_k^r, to decrypt the shared data, as shown in Algorithm 2.

Algorithm 2 Sharing Protocol

function SHARE(Ad^w, $Cred^w$, S_k^w, P_k^r, Ad^r)
  $k_{sym} \leftarrow A.decrypt(getKey(Ad^w), S_k^w)$
  $V \leftarrow A.encrypt(k_{sym}, P_k^r)$
  $addEncKey^{Cred^w}(V, Ad^r)$
end function

d) Reading protocol: enables an entity, given its public and private key P_k^r S_k^r, to decrypt and read a data stored in $channel$ at the index $i \leq |channel|$ once it has been authorized by the writer (i.e. shared), as shown in Algorithm 3.

Note that this function only needs the current blockchain state and can be called off-line. Thus, it may be used to create

Algorithm 3 Reading Protocol

function READ(P_k^w, Ad^r, S_k^r, i)
  $k_{sym} \leftarrow A.decrypt(getKey(Ad^r), S_k^r)$
  $D \leftarrow A.verify(retrieve(i), P_k^w)$
  return $S.decrypt(D, k_{sym})$
end function

a list object in the relevant application context, using this function as a proxy for the list getter.

Using this protocol, the owner of the contract can share his data with new entities by calling the function $share$ at any time using their public keys. On the readers side, entities can trust the shared encrypted keys origin and authenticity. Indeed, the smart contract guarantees that only the owner of the contract is able to share keys and the blockchain ensures that they have not been tampered by any attacker.

A. Security considerations

In this section we will provide with a few security arguments that asserts this protocol usability in real world environments. We show that our protocol achieve data confidentiality and authenticity.

We make the following assumptions:

- Blockchain authenticity and immutability: Being able to modify $channel$ or $keystore$ is equivalent to knowing $Cred^w$.
- S and A usual completeness and at least semantic security. Security against Chosen Ciphertext Attacks is not explicitly required, but is recommended since the smart contract may act as a decryption oracle, in case the attacker is able to manipulate the $keystore$ and/or $channel$ contents.
- For all $id \in I$, S_k^{id}, $Cred^{id}$, are only known by id, i.e. secret keys do not get compromised. This is also assumed for temporary data computed during protocol execution.

1) Completeness: Let $P_k^w, S_k^w, Ad^w, Cred^w, P_k^r, Ad^r$ the properly generated credentials. After the i$^{\text{th}}$ successful $WRITE$ call, we have: $retrieve(i) = S.encrypt(A.sign(data^i, S_k^w), k_{sym})$
$getKey(Ad^w) = A.encrypt(k_{sym}, P_k^w), Ad^w)$
with $k_{sym} \leftarrow S.generate()$.
After a successful call to $SHARE$, we have: $getKey(Ad^r) = A.encrypt(A.decrypt(getKey(Ad^w), S_k^w), P_k^r)$
Moreover: $READ(P_k^w, Ad^r, S_k^r, i) =$
$S.decrypt(A.verify(retrieve(i), P_k^w),$
$A.decrypt(getKey(Ad^r), S_k^r))$
Therefore by trivial substitution $READ(Ad^r, S_k^r, i) = data^i$.

2) Security: Our scheme is a straightforward hybrid construction and as such benefits from plethora of existing literature ([13], [14], [15]). However we provide a few informal arguments to assert its security properties, *i.e.* confidentiality and authenticity.

Any external attacker only has a read access to:
$\forall i, S.encrypt(sign(data^i, S_k^w), k_{sym})$
$\forall id \in R \cup \{w\}, Ad^{id}, P_k^{id}, A.encrypt(k_{sym}, P_k^{id})$
With R the set of readers, and $k_{sym} \leftarrow S.generate()$.

S semantic security implies that a probabilistic polynomial time attacker cannot get any information on $data^i$ without having any information on k_{sym}. A semantic security implies that the same attacker cannot get any information on k_{sym} without having any information on $S_k^{id}, id \in I$. Since we assumed the attacker cannot access those, any external attack cannot deduce information on $data^i$, i.e. our construct provide confidentiality.

Moreover, authenticity is directly provided by the signature algorithm, since all occurrence of $data^i$ are signed and verified.

B. Implementation

In this section, we give a practical implementation example of the protocol based on the car log book application. Note that this is not the actual implementation of the original carbook contract but a simplified version that aims to illustrate the cryptographic protocol. The use case consists in a car owner who wants to share with some readers (e.g. a potential buyer, an independent garage) his *CarbookExample* contract on which encrypted labels are published. Readers that have been granted access are then able to read elements pushed by the owner on the contract.

The implementation is composed of a Solidity smart contract that represents the car log book and a client application (off-chain) to access the smart contract and also to manage the cryptographic functions.

The S scheme is instantiated with AES in GCM mode, and A with RSA OEAP with sha256 and MGF1 padding (as defined in PKCS#1).

While the whole source code of the smart contract is given in listing 1, for format purpose, we do not provide the client code. However we give some information on how to implement such client using Java. For information, this section uses a blockchain Ethereum vocabulary, it will be assumed that the reader will be familiar with this jargon.

```
pragma solidity ^0.4.11;
contract CarbookExample {
address public owner;
struct Record {
  string content;
  uint generation;}
uint public currentGeneration;
Record[] public history;
mapping(address => mapping(uint => string))
  keystore;
function CarbookExample(string _key) public {
  owner = tx.origin;
  currentGeneration = 0;
  keystore[owner][currentGeneration] = _key;}
function append(string _content)
public onlyWriter() {
  history.push(Record(_content, currentGeneration))
    ;}
function addMember(address _member, string
    _memberKey, uint _generation)
public onlyOwner() {
  keystore[_member][_generation] = _memberKey;}
function incrementGeneration(string _key)
```

```
  public onlyWriter() {
  currentGeneration ++;
  keystore[owner][currentGeneration] = _key;}
function getCipheredRecordContent(uint _index)
    public constant returns (string _content, uint
    _generation) {
  if (_index > history.length-1) { return; }
  _content = history[_index].content;
  _generation = history[_index].generation;}
function getKey(address _member, uint _generation)
    public constant returns (string _key) {
  _key = keystore[_member][_generation];}
modifier onlyOwner() {
  if (tx.origin != owner) { return; }
  _;}}
```

Listing 1. SecretChannel Solidity source code

a) Initialization: The owner starts by generating on the client side: Ethereum credentials ; a RSA key pair ; an AES key k_c that will serve to encrypt all the data on his contract. The Ethereum credentials can be generated using the Web3j Java library's *Credentials* class [16]. RSA and AES keys generation can be done using the standard javax.crypto.KeyGenerator of the standard java.security package.

Then, the owner encrypts k_c using his RSA public key to obtain the encrypted key k_w.

Once all the keys have been generated, the owner deploys the CarbookExample smart contract using his credentials and initializes it with k_w (see the constructor in listing 1). Deployment and other smart contract function calls can be done using org.web3j.

b) Push: To publish on his smart contract, the owner signs the label data and symmetrically encrypts it using k_w before calling the function *append()* in listing 1. Also, this latter function verifies the identity of the owner by comparing the caller's address to the one stored at the initialization (see the modifier *onlyWriter()*).

c) The concept of generation and removing a member: A generation correspond to the index of the cryptographic key k_c. The owner is able to change the generation when he wants to revoke the access rights of a previously authorized reader by generating a new k_c.

d) Share and read: In order to decipher items of the channel, a reader has to generate his Ethereum credentials and an RSA key pair (rsa_{priv}, rsa_{pub}) and give both public keys to the owner. The owner is then able to encrypt asymmetrically k_c using rsa_{pub} and call the function *addMember()* with the address of the reader, the obtained k_m and the *generation* of the key. The reader retrieves the ciphered message by calling *getCipheredSecret()* and its generation number. He is then able to retrieve his corresponding key k_m using *getKey()* and decipher it to obtain k_c. Once the reader posses k_c, he deciphers the content and checks the signature of the clear data.

VI. Conclusions

In this paper, we investigated the issue of vehicles' data management over a consortium blockchain, which is formed by an automobile manufacturer, an insurance company and a service provider. The proposed blockchain architecture aims at preventing common types of vehicles frauds and to foster the collaborations and data sharing between the involved stakeholders. In this context, we designed a novel hybrid cryptographic protocol to secure the sharing and access to vehicles data. An example of smart contract implementation has been described and various security considerations have been analyzed.

In future works, we intend to investigate the remaining challenges to accelerate the adoption and mass-market deployment of the proposed framework. Among them, the most important topics are: a decentralized on-chain user authentication system that would be based on the use of smart contracts ; a study on transactions rates and volume for scalability issues ; a study on system's regulatory compliance regarding data immutability.

Acknowledgment

This research work has been carried out under the leadership of the Institute for Technological Research SystemX, and therefore granted with public funds within the scope of the French Program Investissements d'Avenir.

References

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.
[2] "The vinchain project - trustworthy vehicle history on the blockchain," accessed: 2018-04-16. [Online]. Available: https://vinchain.io
[3] "The carvertical project," accessed: 2018-04-16. [Online]. Available: https://www.carvertical.com/
[4] Renault, "Groupe renault teams with microsoft and viseo to create the first-ever digital car maintenance book prototype," July 2017, accessed: 2017-11-13. [Online]. Available: http://goo.gl/YxTtye
[5] Bosch and TUV Rheinland, "A solution for odometer fraud," July 2017, accessed: 2017-11-13. [Online]. Available: http://goo.gl/XPJQHV
[6] T. Willemarck, H. Graafland, C. Gonderinger, J. Cobbaut, B. Lycke, J. Hakkenberg, and M. Peelman, "Protect european consumers against odometer manipulation," October 2014, accessed: 2017-11-13. [Online]. Available: goo.gl/CnQuYg
[7] "jpmorganchase/quorum." [Online]. Available: https://github.com/jpmorganchase/quorum
[8] "Hyperledger Project." [Online]. Available: https://github.com/hyperledger
[9] "Build your first blockchain network with cryptographic material from your own certificate authority (CA)," Nov. 2017. [Online]. Available: https://goo.gl/9P78Zj
[10] IRT SystemX, "Blockchain for smart transactions (bst) research project," accessed: 2017-11-13. [Online]. Available: http://www.irt-systemx.fr/en/project/bst/
[11] K.-L. Brousmiche, T. Heno, C. Poulain, E. Ben-Hamida, and A. Dalmieres, "Digitizing, securing and sharing vehicles life-cycle over a consortium blockchain: Lessons learned," in *IFIP NTMS Blockchain and Smart Contracts Workshop (BSC18)*, 2018.
[12] "GitHub - jpmorganchase/quorum: A permissioned implementation of Ethereum supporting data privacy," accessed: 2017-10-20. [Online]. Available: https://github.com/jpmorganchase/quorum
[13] S. Alt, "Authenticated Hybrid Encryption for Multiple Recipients," Tech. Rep. 029, 2006. [Online]. Available: https://eprint.iacr.org/2006/029
[14] R. Cramer and V. Shoup, "Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack," *SIAM Journal on Computing*, vol. 33, no. 1, pp. 167–226, 2003.
[15] J. Herranz, D. Hofheinz, and E. Kiltz, "Some (in)sufficient conditions for secure hybrid encryption." Tech. Rep. 265, 2006. [Online]. Available: https://eprint.iacr.org/2006/265
[16] "web3j: Lightweight Java and Android library for integration with Ethereum clients," Jan. 2018, original-date: 2016-09-04T05:48:49Z. [Online]. Available: https://github.com/web3j/web3j