First year's report update

Anwaar Ali

Abstract

This document tries to address the comments made and concerns raised during my first year report evaluation. In summary the document is organized in way so that it tries to i) provide motivation for a blockchain's use case, ii) shortcomings in the existing state of the art, iii) challenges and potential pitfalls that might occur while employing the blockchain paradigm for a particular use case, and finally the document closes with a iv) discussion on the architecture and the experimental setup to study a blockchain-based use case. I also highlight different metrics for experimental evaluation and opportunities to analyse and compare different blockchain-based techniques and protocols.

I. Motivation for a blockchain's use case

In what follows I describe three motivating scenarios highlighting the shortcomings in their existing systems' implementation where blockchain can be used to address these shortcomings more efficiently.

A. Scenario I: Information sharing and data ownership

Personal data is of utmost importance in today's connected and digital age. With each interaction between a user and an online application, the service providers of such applications gather some information about the users in return for personalised services. Some services like healthcare and online shopping almost always contain sensitive and private information related to users. The service providers are usually trusted with the secrecy and integrity of such a users' data. Unfortunately there have been numerous incidents of users' data being either used or revealed to unauthorised third parties[1][2]. Such unethical activities breach the data protection laws in place. Such incidents create a need for a trusted data storage and sharing system. The trust can be provided in terms of evidence gathering where such an evidence describes the audit of the data flow from one point to another. Such an audit can reveal access violations on a piece of data and reveal the entities responsible for such a malicious activity. The record of such data flow can be considered as a data provenance problem where all the operation on a piece of data are recorded since its inception. Ideally there should exist a system that can record this provenance data in an immutable, trusted and transparent manner.

B. Scenario II: Mutually non-trusting multiple parties

Building upon the discussion in the last subsection we have an environment with mutually non-trusting multiple parties interested in doing business with each other. Specifically, these parties can consist of data owners who wish to subscribe to third party service providers. Some third parties, such an online shopping and social networks, provide their users with personalized recommendations and advertisements. They do so by using their users' data and selling them to different brands and services[3][4]. Now, at a first glance, such tactics might seem intrusive and malignant. But in my view, at the end of the day, we do need such customised services such as personalised health care and recommendations about healthcare products (or simply which product to buy or which book to read) that might be possible with the a user's sensitive/private data. So the problem now is *how to reconcile multiple mutually non-trusting parties in a transparent and trusted manner?* This scenario creates an environment with mutually non-trusting stakeholders (e.g., data owners and data gatherers) who are interested in the trusted management of an underlying resource. Here, the underlying resource is data itself that must be managed in a transparent and trusted manner. This is only possible if there exists a system that can automate interactions (in terms of transactions on the underlying resource) among these parties in a trusted, transparent, auditable and in a way that is compliant to certain policies and regulations.

C. Scenario III: Compliance

Compliance is the next step that can be achieved if a system exists that is capable of automating interactions among mutually non-trusting parties against an underlying common resource. Specifically, as an example such compliance can take the form of digital enforcement of data protection policy. The automation of such a compliance will dictate transparent flow of data among multiple parties with the provision of trusted and transparent auditing. Any breaches of compliance policy will be detected and halted by the system in terms of recording *invalid transactions (as per a policy/regulation/contract at hand)* made

[1] https://www.theguardian.com/society/2018/jan/31/nhs-chiefs-stop-patient-data-immigration-officials
[2] https://www.theguardian.com/world/2013/jun/06/us-tech-giants-nsa-data
[3] https://www.facebook.com/full_data_use_policy
[4] https://www.facebook.com/business/products/ads/ad-targeting

against a piece of data. The record of all transactions (valid or invalid) should ideally be kept in an immutable manner for auditing and evidence sharing process. Data provenance studies such as whole-system provenance [1] and the study of audit compliance with privacy policy in the IoTs [2] emphasize the need for such a system. These works envision a system that can *technologically enforce* a policy that is reflective of the actual laws and regulations.

D. Limitation of the existing state-of-the-art

The two works [1], [2] described above present a solution for evidence capture in terms of data provenance for auditing purposes. There is still a need for an automatic policy enforcement in technological domain and recording of such an evidence in an immutable, trusted and transparent manner. The work [3] considers data provenance in cloud environment using blockchain but it is also essentially evidence gathering system. There is room for research to design and implement a system that can build upon such studies in terms of implementing and automating a policy/regulation.

E. Why blockchain is an appropriate fit?

Blockchain in its simplest manifestation is a distributed and immutable database. Each record is *appended* in this database only after all the concerned authorities reach a *consensus* on the validity of this record. This has the potential to solve the issue of trust in the above mentioned trust-less environment.

Further blockchains can be used to automate smart contracts. These smart contracts have the capability to automate inter-party interactions according to a pre-defined policy. Smart contracts execute in a distributed manner as well. The state of these programs is stored in a distributed and immutable manner as well on top of blockchain. This provides us with the opportunity of automating such system in distributed, trusted and transparent manner.

F. Description of the use case

It is important to note that the use case that I intend to study can be described *generally in an application agnostic manner* as *policy compliant data flow*. The use case described in this subsection can then find its concrete applications in different areas such as medical health record management and trusted cloud computing. To generalize, we can have various data generators and data gatherers as shown in Figure 1. Each third party accesses only the *subset* of a user's data as agreed upon during the signup process. I am making an *assumption* here that each data generator and gatherer is part of a global blockchain network. I will elaborate this assumption further in the next section. Each data access request is treated as a transaction. This transaction is then evaluated as per an *endorsement policy* implemented using a smart contract. If the policy is satisfied and a consensus on such a transaction is reached only then an access to some data (e.g., current geolocation) is shared with the interested third party. It is important to note that only the *hash of the actual data in question* is then recorded as part of the transaction hence maintaining the user privacy. I also assume inter-party interactions regarding data sharing among third parties to also go through the smart contract endorsement layer. Finally a simple audit mechanism, similar to the one described in [3], can be in place that can request all the transactions made against a certain hash (which represents a piece of data). This way not only activities like access violations can be detected but also the integrity of data as well.

II. Experimental setup and system architecture

I intend to use Linux Foundation's Hyperledger Fabric (HLF)[5]. This is an implementation of a *permissioned blockchain* with the features of customised membership services and consensus mechanisms. Permissioned blockchain implies that only those can be part of a blockchain network who are specifically certified by some authority. This certification is managed by *membership service providers (MSPs)* which can be an organization who deploys HLF by issuing cryptographic certificates (specifically x.509 in case of HLF). MSPs abstract certification and cryptographic protocols. The vision behind HLF is automating trusted business-to-business transactions. HLF gives enough freedom to its users to implement their own policies in terms of consensus, particular business logic, and privacy and confidentiality policies while at the same time being part of the overall blockchain network. First I describe the main components of this system and then I will explain the architecture diagram considering the use case described above.

The architecture diagram of HLF is shown in Figure 2. Following are its main components and their descriptions:

Organization X This organization can take any shape or form. It can be an actual organization like a business entity or an end user as e.g., a data owner as described above. Because of the modularity of HLF (it becomes clear as we go though the document) any number of organizations with an arbitrary relationship among them can be established over a HLF-based blockchain network.

Node.js SDK This provides an API to access the HLF blockchain. Clients which are external to the blockchain propose different transactions through this API. We can imagine a web-based application using this API to access the underlying HLF.

[5]https://www.hyperledger.org/

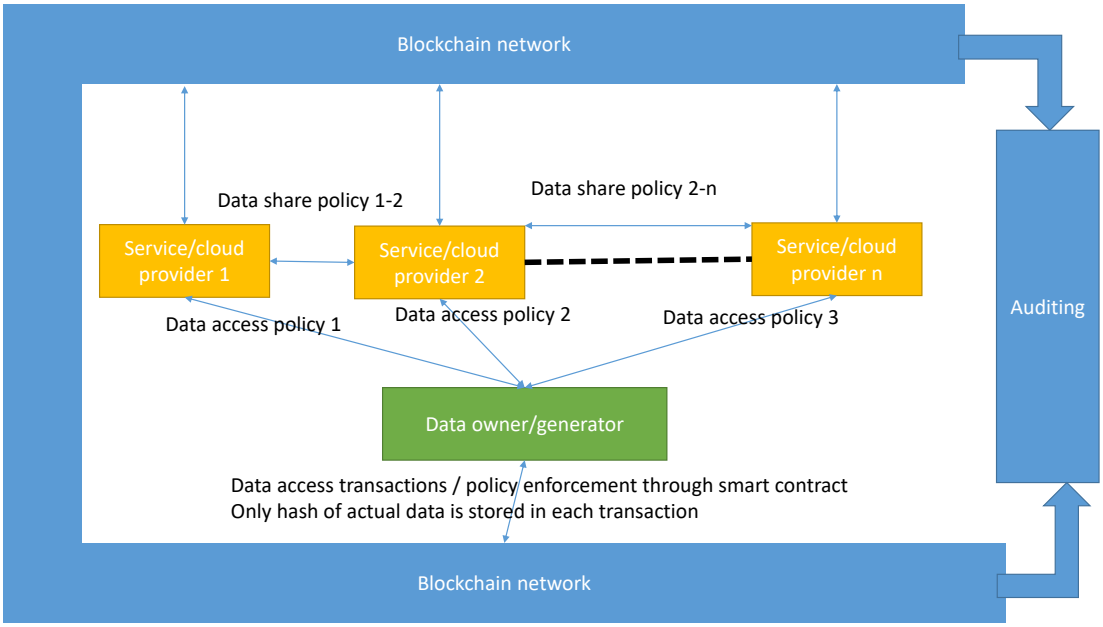

Fig. 1: Blockchain-based policy-compliant data provenance system

Peers Peers are the entities that maintain the state of the (blockchain) ledger and take part in consensus process as per a custom endorsement policy.

Chaincode Chaincode provides a customized interface through which peers and external clients can interact with HLF's blockchain. An instance of the same chaincode is installed on all the peers hence chaincode is simply another name for smart contract in HLF vernacular. The endorsement policy, as mentioned above, is actually defined in this chaincode. This endorsement policy can define the particulars of the consensus mechanism. As for example, it can specify that only 51% of total peers need to take part in endorsing a transaction. This is how one can customize consensus particulars depending upon a use case or an application. A chaincode is mainly used to implement the overall business logic as well.

Orderer Orderer is a very important part. This solves the same *double spending problem* as solved by *proof of work* in Bitcoin. It orders (or timestamps) all the transactions and verifies that all the transactions are valid as per an endorsement policy. Now, at a first glace it seems that there is only one orderer as shown in Figure 2 which can be a caveat in the form of single point of failure. This is just for development purposes in production multiple orderers can be used but then we have to be careful in their synchronisation. The orderer is the one that collects transaction in a block and appends new blocks in the blockchain as maintained by the peers.

Channel Channel is a very important entity. Channel can be considered as an overlay to the same HLF blockchain for data isolation and confidentiality. It provides different levels of confidentiality. Different channels maintain different ledgers and endorsement policies. Cross channel communication is strictly prohibited. Organizations can create different channels and assign different peers and instantiate different chaincodes per channel (with different endorsement policies) in them. The concept of channel is also important in addressing the scalability issue. A channel in itself can be considered as an independent instance of HLF with its own peers, ledger, consensus mechanism/endorsement policy, and set of chaincodes (implementing the overall business logic). As shown in Figure 2 a channel can cover peers from two organizations. This means these peers are able to talk to each other and maintain the same version of a ledger. It is important to note here that a channel can have multiple chaincodes but there can only be one ledger per channel. A same peer, however, of an organization can be part of different channels at the same time.

Docker[6] Docker is light-weight virtualization technology. The fact that each component of HLF is implemented using a separate docker container makes automation and implementation of a new business logic quite easy and manageable. Docker Compose[7] is used to instantiate and configure all the necessary HLF parts using different containers and then these containers can be inter-networked (if multiple machines are being used—something that I am also trying to experiment with currently) using Docker Swarm[8].

In the use case described in the last section one can make use of HLF in a very efficient manner. A user can create different channels for different third parties with appropriate chaincode installed in each channel that takes care of the endorsement

[7]https://docs.docker.com/compose/
[8]https://docs.docker.com/engine/swarm/

policies relevant for the channel. A separate ledger is maintained at orderer which records all the transaction (either valid or invalid). This way it provides a very useful resource for auditing. The detailed diagram with the explanation and particulars about the components can be examined in Figure 2.

A. *Moving ahead*

Currently I have HLF setup and, separately, I have also setup the whole system provenance capture as described in [1]. I am interested in interfacing the provenance stream with HLF as shown in Figure 1. This is a challenging task as I have to define the structure of the transaction based on the provenance data as taken from the provenance stream generated by CamFlow [1]. I also have to define the transaction/unit time. Once I do this I can then start thinking about the endorsement policy/business logic in terms of programming a chaincode. I envision this chaincode to perform the ultimate task of policy compliance data flow among different entitities/processes.

As far as evaluation is concerned then HLF gives me enough freedom to evaluation different customized consensus protocols. Also simple tweaking of transaction structure and the logic behind a chaincode will provide valuable insights of the efficiency of my policy compliant auditing system.

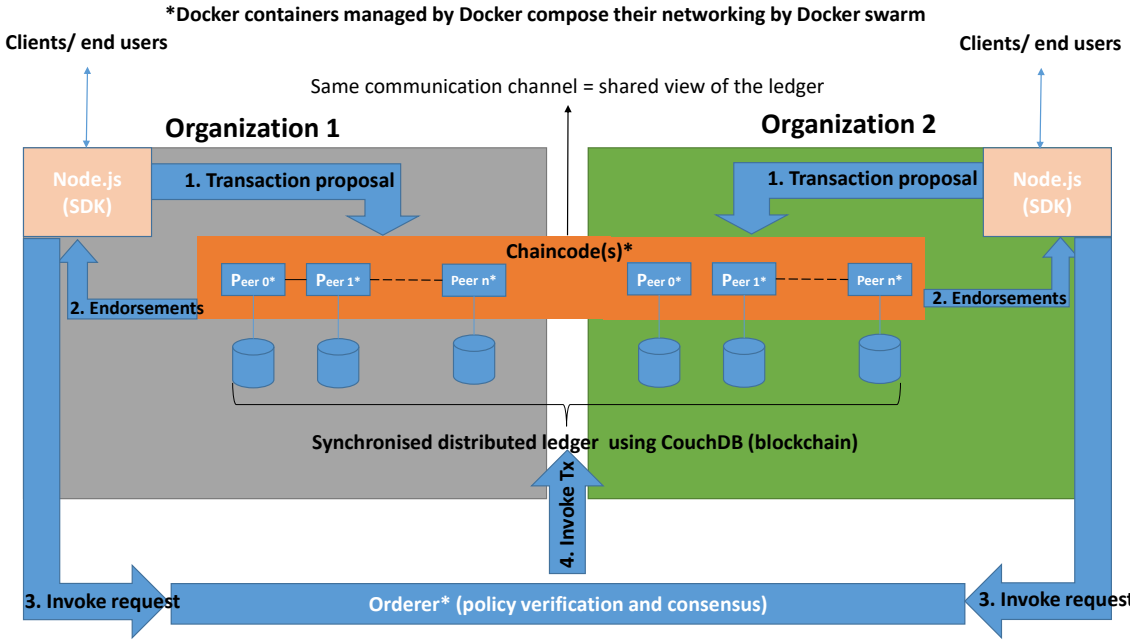

Fig. 2: Hyperledger Fabric architecture

REFERENCES

[1] T. Pasquier, X. Han, M. Goldstein, T. Moyer, D. Eyers, M. Seltzer, and J. Bacon, "Practical whole-system provenance capture," in *Proceedings of the 2017 Symposium on Cloud Computing*. ACM, 2017, pp. 405–418.

[2] T. Pasquier, J. Singh, J. Powles, D. Eyers, M. Seltzer, and J. Bacon, "Data provenance to audit compliance with privacy policy in the internet of things," *Personal and Ubiquitous Computing*, pp. 1–12, 2017.

[3] X. Liang, S. Shetty, D. Tosh, C. Kamhoua, K. Kwiat, and L. Njilla, "Provchain: A blockchain-based data provenance architecture in cloud environment with enhanced privacy and availability," in *Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*. IEEE Press, 2017, pp. 468–477.

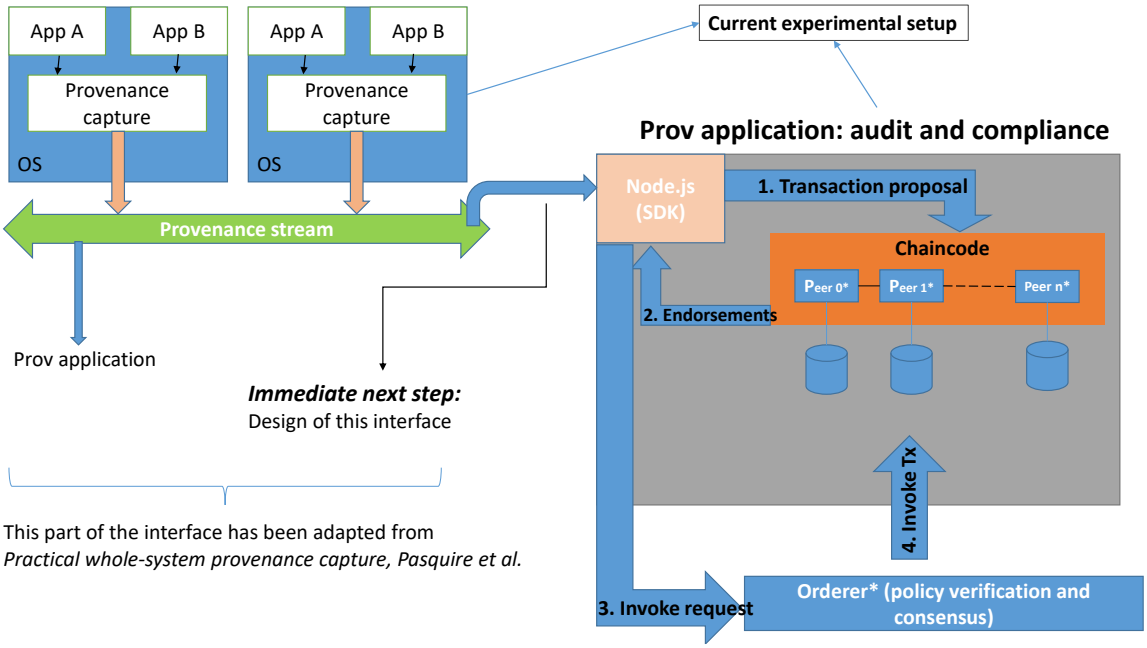

Fig. 3: Data provenance with Hyperledger Fabric architecture