BANCO CENTRAL DO BRASIL

Distributed ledger technical research in Central Bank of Brazil

Positioning report

Technical consultants
Aldenio de Vilaca Burgos
Jose Deodoro de Oliveira Filho
Marcus Vinicius Cursino Suares
Rafael Sarres de Almeida

E-mail
blockchain@bcb.gov.br

Research Manager
Aristides Andrade Cavalcante Neto

Chief Information Officer
Marcelo Jose Oliveira Yared

Authorized by Deputy Governor
Luiz Edson Feltrim

Central Bank of Brazil
Brasilia - DF, Brazil
31st August 2017

Distributed ledger technical research in Central Bank of Brazil

Aldenio de Vilaca Burgos
aldenio.burgos@bcb.gov.br

Jose Deodoro de Oliveira Filho
jose.deodoro@bcb.gov.br

Marcus Vinicius Cursino Suares
marcus.suares@bcb.gov.br

Rafael Sarres de Almeida
rafael.almeida@bcb.gov.br

Abstract

Distributed Ledger Technology (DLT) is a novel technology that powers trusted distributed databases. This paper presents a research of an IT department team of Central Bank of Brazil on the matter, the experience in analyzing potential use cases, and the examination of some of the available platforms to develop working prototypes of a minimal Real Time Gross Settlement System. This work also describes perceived privacy issues related to the technology acquired during the implementation effort, and presents a naive approach to tackle these issues.

1. Introduction

Distributed Ledger Technology (DLT) is a rapidly developing technology that powers trusted distributed databases, applicable to solve problems in various scenarios by eliminating the need of third party intermediation. At this point of history, both private and public institutions around the world acknowledge its potential and consider it a revolutionary technology. They are, however, uncertain of how it could be leveraged, even though multi-billion dollar markets (cryptocurrencies) already present a successful use case of this technology.

The current scenario resembles the introduction of the commercial Internet: a powerful, possibly game changing technology is introduced and, although incumbents can't quite grasp how to best apply it, they are fully aware a race has started. At this point in time, all sorts of companies are trying to figure out how to evolve and adapt the technology in order to apply it to a variety of problems. In this path, they are continually finding qualities and shortcomings, and creating new platforms, standards and protocols in order to mold DLT characteristics to fit application requirements.

Views expressed in this work are those of the authors, and do not necessarily reflect those of the Banco Central do Brasil

This work surveys relevant implementations, reference projects where such technology has been tested, and describes working experiments to assess possible responses to privacy issues. This also work reports on lessons drawn from these experiments with a hope that they will inform future decisions of Central Bank of Brazil with respect to this technology.

It starts by a brief description of the technology, how it came about and which problems it may be applicable to; the next sections bring the current state of research on this topic, by analyzing reports by various public and private financial institutions around the world and academic papers related to the core issue discussed throughout this research (privacy in a blockchain environment); further along, this work gives an account on the Central Bank's team experience on finding a suitable use-case and building a proof of concept to validate the acquired knowledge and DLT's limits; then it closes with a discussion over the experience's results and drawn conclusions.

2. Concept

A *blockchain* is a distributed database used to maintain a continuously growing list of verifiable records, a ledger. The term derives from the way data is stored in a continuous flow of blocks chained together using a cryptographic hash function. Although the term *Blockchain Technology* is usually used to describe any type of shared database that prescinds a trusted third party regardless the way the data is arranged internally, *Distributed Ledger Technology* is formally considered a broader, more suitable term. In this document, terms will be used interchangeably.

In the original design [1], a blockchain resides in a peer-to-peer network composed of client and server (or miner) nodes: the former set comprehends any device interacting with the network to use the services it provides (e.g. financial transactions, smart contracts, etc); the latter are devices in charge of verifying and maintaining the integrity of records and the ledger. Client nodes request server nodes to store transactions, changes in the state of any previous registry on the ledger. Transactions are signed with a public key scheme, guaranteeing their authenticity and non-repudiation properties (i.e. that they were proposed by authorized parties and these parties can't deny their participation at the same time). Server nodes then work to gather transactions into blocks and try to *mint* them into the ledger by solving a cryptographic puzzle, proving that resources have been committed to the proposed update. This process is known as *proof of work* (PoW). By design, although costly to produce, results of a PoW are easily verifiable. Once generated, the block and its proof of work are broadcast to the remaining servers, which verify their validity. Accepted blocks are appended to the ledger and the server which generated the update receives a reward for its work. Thus, as the acceptance rules are the same for every node, the network reaches *consensus*: all nodes agree on and keep a unique, authenticated copy of the ledger.

The ingenious combination of *asymmetric cryptography*, *proof of work* and *consensus* mechanisms represent the greatest novelty of this technology: they allow *blockchains* to be particularly resistant to *double spending*, *Sybil attacks* and *Byzantine faults*. In other words, the blockchain technology is able to support a ledger in which: no user should be spend funds (or assets) beyond the resources they own; no user is able to impersonate another (unless they are able to steal their private keys by unrelated means); and no dishonest server node is able to forge unauthentic transactions (either on their own or by colluding with other nodes).

The original mechanism was conceived by Satoshi Nakamoto [1] (pseudonym) as the core technology supporting the *Bitcoin* network, the first blockchain to go live (in 2009). The Bitcoin network created a new market on its own, the *cryptocurrency* market: it spawned several competing public blockchains, generally from projects that aim to sort out its technological shortcomings (i.e. *alt coins*). Nonetheless, it currently dominates this market by representing close to 50% of its multi-billion dollar capitalization value.

The design has evolved since then: some components may be replaced in order to enable new implementations of blockchains to add features and achieve goals outside the original Nakamoto design. More specifically, in recent projects: blockchains were adapted for working in *permissioned networks*, where nodes are subject to governance policies that dictate who may connect to and use it (as opposed to public networks, such as the Bitcoin network, where anyone is free to connect and use at any time); the *proof of work* algorithm is often replaced (e.g. by a *proof of stake* [2]) and sometimes discarded, as it is considered resource intensive and unnecessary for networks of trusted servers (e.g. for permissioned blockchains); and the *consensus mechanism* may be replaced for alternate schemes in order to achieve highest transaction throughput or different levels of fault tolerance (e.g. Ripple [3]).

Among this plethora of new technologies, some noteworthy projects must be briefly described: Ethereum, Hyperledger, Quorum and Corda, for these will be looked into with some detail further along this document.

Ethereum [4], which may be described as a second generation blockchain, is an open-source project created in 2013. It extends the functionality of previous designs by adding the concept of smart contracts [5], a feature that allows the ledger to store and run computer programs. The first public Ethereum backed network went live in 2015 and supports *ether*, currently one of the highest valued cryptocurrency markets [6].

Hyperledger is a set of blockchain technologies projects hosted by The Linux Foundation since February 2016 [7]. The main vision of this initiative is that only an open source collaborative software development may bring about blockchains to commercial adoption, by enforcing community processes to encourage the adoptions of cross-industry standards over time. It currently counts on the support of well-known companies from both the IT (such as IBM, Intel and Cisco) and financial services (e.g. ABN AMRO, J.P.Morgan Chase, SWIFT) sectors.

During 2016, while participating in the Hyperledger Project, J.P. Morgan Chase introduced *Quorum*, an implementation of a permissioned blockchain built on top of the Ethereum platform source code [8]. This project aims to solve privacy, security and regulatory challenges that currently hinder the adoption of public blockchains by the financial sector.

The recently launched *Corda Distributed Ledger* moves in the same direction. It is a related technology created in 2016 by the R3 consortium, a company founded in 2015 and that currently gathers over 80 of the world's biggest financial institutions [9]. Although inspired by blockchains, designers opted to diverge from the described design in order to drop some characteristics they deem unfit for most banking scenarios [10]. Particularly, it leverages no cryptocurrencies and no globally shared data, while still allowing for regulatory and supervisory features, consensus mechanisms and smart contracts.

3. Benefits and Applications

The first application of a blockchain was powering decentralized payment systems, specifically *cryptocurrencies.* In the days before their inception, payment systems needed payers and payees to trust either in each other, or in third parties to intermediate transactions and guarantee that values would be transferred accordingly (i.e. a trusted authority, a *centralizing* node). In a distributed ledger it is possible to avoid centralization and yet create global reliable peer to peer networks because trust is transferred to technology, notably the robustness of cryptographic algorithms and the consensus mechanism. Thus, the foremost benefit of this technology is *the reliable decentralization of trusted networks.* Aside from peer to peer (pseudo) anonymous payment networks such as cryptocurrencies, typical examples of this case are cross-border payment systems, since it is hard for nations to find such trusted third parties to agree upon.

However, financial transactions are far from the only application of this technology. Cryptography and consensus are applicable to any digital representation of real world assets: in payment systems, assets are balances of whatever monies that users might transfer to each other. Digital ledger records may seamlessly store a great variety of information, such as digital documents that contain the rules of engagement between actors (i.e. smart contracts); health records [11]; or identity records for any type of entity (e.g. individuals and organizations). Furthermore, any information stored under a blockchain may be fully trusted to have been recorded by a verifiable entity (i.e. one which possesses the respective private keys) and untainted by anyone else. Thus, another benefit of this technology is *the creation of a permanent, trusted record of assets and transactions.*

Awareness of these features moved the Central Bank of Brazil to task a study group to evaluate and analyze currently available blockchain platforms in order to further understand both the technology's applicability and shortcomings. A reliable, immutable, trusted ledger might be an ideal tool to tackle solutions for situations that may profit from full decentralization and strong resilience to individual failures, making it a candidate fit to a large set of previously unexplored problems. This study group's goals comprehended both finding such scenarios and identifying any limitations presented by the current state of the art of the technology.

4. Previous and concurrent work

This is a summary of works from other public and private financial institutions that, as the Central Bank of Brazil, have recently investigated this technology to the best of our knowledge.

4.1. Project Jasper

In 2016, Payments Canada, the Bank of Canada, R3 and Canadian commercial banks that were members of the R3 consortium, initiated Project Jasper [12], a proof of concept system to explore a DLT based wholesale payment system in two phases: using an *Ethereum* platform in the first; and a *Corda* platform in the second phase. Institutions used digital depository receipts (DDRs) issued by the Bank of Canada to exchange and settle interbank payments. This arrangement eliminated Credit Risk from the system by requiring backing of DDRs by deposits at the Bank of Canada. However, the study pointed out that DLT would need additional devices to address liquidity and settlement risks. The former is mitigated by using liquidity-saving mechanisms (LSM), reducing funding demands for settling multilateral transactions, while the latter may not be solved by traditional blockchains: blocks are naturally transient for a few cycles due to consensus algorithms, thus there is always some probability that transactions end up in orphaned blocks and never go into the main chain. Both problems were addressed by centralized features in phase 2 of Project Jasper by using *Corda*: a distributed queue mechanism was used to implement a LSM, whereas finality is inherent by design on this platform (through notary nodes). Moreover, authors point out that the inherent transparency of DLT is undesired for a wholesale payment system, and remark that the introduction of privacy preserving features creates additional points of failure, either by the existence of a notary (a centralizing node) or by *sharding* the information set (storing unrecoverable private data at participating nodes), leading to the question of whether this constraint could compromise the resilience benefits of DLT.

This study concluded that wholesale payment systems require inherently centralized features, such as LSM mechanisms, that would compromise the distribution characteristic of DLT, therefore limiting the benefits that could be obtained by applying this technology compared to the current centralized systems. However, it could yield cost saving and efficiency gains by the automation of settlement and clearing processes if assets and payments are integrated on the same ledger (i.e. triple entry bookkeeping [13]), especially in the interbanking market (e.g. over-the-counter markets for stocks, bonds and derivatives).

4.2. Project Ubin

At the end of 2016, the Monetary Authority of Singapore (MAS) partnered with the R3 consortium in order to build a proof of concept (PoC) to conduct interbank payments facilitated by DLT, called Project Ubin, counting with the collaboration of several institutions, such as Bank of America Merrill Lynch, Credit Suisse, J.P. Morgan, among others [14]. In Phase 1, for six weeks, a private Ethereum blockchain supported large-value local currency interbank funds transfers and settlement of script less Singapore Government Securities between participants. MAS opted for a monetary model that demanded banks to exchange cash collateral for Depository Receipts on the distributed ledger at any time, allowing the system to overlook credit and liquidity risk issues, and concentrate the study in technical analysis. Project Ubin is scheduled to go into further phases, which aim to focus on securities settlement, cross border payments and the selection of a DLT platform.

4.3. Emerald

Since currency international payments have no central authority by design, the Royal Bank of Scotland considered DLT a good fit for cross border transactions and assigned a team to evaluate the blockchain technology by building *Emerald*, a Clearing and Settlement Mechanism on top of a modified *Ethereum* platform [15]. By modifying the original source code, they were able to build a Distributed Ledger with increased throughput and decreased latency (block mining cycle) compared to the public Ethereum blockchain. Performance tests showed Emerald is on a par with the current national and international payment schemes (100 transactions per second and 10 second confirmation). Notably, authors point out that some features in the Ethereum blockchain are discardable for the private blockchain use case (such as internal computation cost, i.e. *gas*, and the proof of work algorithm), and consider that replacing these mechanisms would provide even faster, more deterministic block mining time than those obtained through their experiment. The team concluded that, although the Ethereum platform is originally tuned for the public blockchain use case, with modifications, it is well capable to scale and accommodate payment volumes consistent with their local domestic payment systems.

5. Technical experimentation, phase 1

As described in section 3, the Central Bank of Brazil put together a study group in order to evaluate and analyze the blockchain technology and further understand both its applicability and shortcomings. The work was split in two phases:

1. The first phase started on September 2016, its main objectives were: to look for use cases inside Central Bank of Brazil; to elect one of the use

cases and a platform for prototyping; and to produce a minimal proof of concept using it. This phase lasted for sixty days;

2. The second phase started on January 2017: this time, the purpose was to analyze competing blockchain platforms using the selected use case as a benchmark. It lasted for forty five days.

This section contains a detailed report of work done and achievements for the first phase, whereas the section 6 does the same for the second phase.

5.1. Use case selection

In the first phase, the study group searched for candidate DLT use cases inside the Central Bank of Brazil, revealing four candidates for further investigation: identity management systems, the Local Currency Payment System (SML), the Agreement on Reciprocal Payments and Credits (CCR), and the Alternative System for Transactions Settlement (SALT).

These use cases were explored in order for the team to elect one of them for the proof of concept. The ideal case for prototyping would contain a clear core set of requirements that would favor rapid development, in order to allow for examination of how the technology could be leveraged. Furthermore, it was also important that the set of actors (i.e. effective network's participants) were reasonably within reach of the team, in order to facilitate communications and discussion over requirements and results between interested parties.

5.1.1. Identity management

The main objectives of identity solutions are to provide an unique view about users' digital identities, and to give them the power to choose which information might be shared with whom. There is a lot of interest by the financial system in optimization of these systems since *know your customer* processes and compliance with anti money laundering laws both consume a considerable amount of resources.

The Brazilian Federation of Banks (FEBRABAN), for instance, is currently investigating the applicability of DLT in this area, aiming to give customers the ability to seamlessly share their information with more than one bank under a controlled fashion. In a recent event [16], the organization presented prototypes in two distinct platforms.

Furthermore, creating an unique government identity system would allow citizens easy access to public services, thus facilitating financial inclusion, one of the core missions assumed by the Central Bank of Brazil. Unfortunately, time constraints allowed limited research on this topic. However, while it is rather early to draw solid conclusions, preliminary examinations made clear this area is worthy of further exploration in the near future.

5.1.2. Local Currency Payment System

SML is a payment system that enables foreign trade in local currencies by allowing importers and exporters from Brazil to send and receive payments to and from their counterparties in Uruguay and Argentina using local currencies, eliminating the need to conduct currency exchanges for import/export operations [17]. Thus, SML reduces costs of foreign trade transactions for both importers and exporters from those three countries when dealing within that region.

Currently, financial exchange between the local and foreign counterparties in this system is performed in 3 rounds. Roughly, it is as follows:

1. First round:
   (a) Central banks receive transfer orders from importers' local banks;
   (b) Central banks exchange digital files with orders, and agree on a currency exchange rate.
2. Second round:
   (a) Central banks debit reserve account of importers' local banks;
   (b) Central banks net transfer orders and figure out how much is needed to settle;
   (c) Settlement is realized using foreign currency transfers.
3. Final round:
   (a) Central banks credit reserve account of exporters' local banks.

Blockchain technology would be able to streamline this process, creating a shared real time database with input validation to minimize reconciliation efforts and eliminate information asymmetry, since much time is currently lost making sure that both parties share and agree upon on the same data. *Ripple's Interledger protocol* [18], for instance, proposes to speed settlement times by integrating the central banks ledgers through currency exchanges (i.e. connectors). On that matter, a recent Bank of England report [19] presents a successful trial of the Interledger protocol in processing cross-border payments across simulated RTGS systems. The institution deems its trial as positive, although it points out some pending challenges that should be tackled in subsequent proofs of concept. Particularly, the handling of availability of liquidity in cross-border payments is to be explored, since Ripple [3] (the tested blockchain technology) was designed for retail and corporate transactions, an application where such feature is not essential.

However, the Central Bank of Brazil team opted for not further exploring applying blockchain to the SML mainly because it is already supported by running systems. For the proof of concept, ideally, the team sought a candidate system for which there is no currently attainable solution using conventional centralized paradigms. In any case, SML remains a very good case for process optimization using distributed ledger technology and financial innovations.

5.1.3. Agreement on Reciprocal Payments and Credits

The Agreement on Reciprocal Payments and Credits was signed in 1982 to reduce friction on commercial trade between participant countries, cutting down the need of foreign currency through a multilateral netting mechanism. The use of this agreement is voluntary and remaining debts are settled every four months between participants.

The main characteristics of the CCR agreement is the offer of reciprocal guarantees of convertibility, transferability and repayment, respectively: that payments will be immediately converted to US dollars; that dollars corresponding to the payments will be remitted; and that debts imputed to the central banks resulting from transactions carried under the agreement will be irrevocably accepted. The main benefit for exporters is the guarantee receipt of exports, eliminating commercial risk, and for importers is the access to financing by exporters abroad, since the latter have reimbursement guarantees offered by the Payment Agreement.

The CCR mechanism is, in practice, a *Payment Compensation System* operated by the participating central banks through periodic compensation. The settlement system is supported by a Center for Operations located on the premises of the Central Reserve Bank of Peru, under the supervision of the Latin American Integration Association (ALADI). This center processes all payment orders between participant central banks and provides a single state view.

DLT could be used to provide a distributed processing environment to support payment processing, ensuring operations even in case of some central bank's disruption, and eliminating the single point of failure. Using a distributed network, there would be no need of trust in the center of operations to provide an unique view. However, it is yet not clear if privacy requirements could be achieved with the available technology.

5.1.4. Alternative System for Transactions Settlement

A *Real-time Gross Settlement* (RTGS) is a basic structure of any country's payment system. In Brazil, the Central Bank is responsible for hosting a modern RTGS system: the *Brazilian Payment System* (SPB). By its own nature, it is a critical technological structure for the national financial system, through which all participants reserves are exchanged. In the case of its catastrophic failure, RTGS members would be unable to send (or receive) funds to (from) each other, leading to a complete financial halt. This very situation is tackled by the BIS Principles for financial market infrastructures: *"An FMI (Financial Market Infrastructure) should also consider alternative arrangements (for example, manual paper-based procedures) to allow for the processing of time-critical transactions in extreme circumstances."*[20].

In this context, the *Alternative System for Transactions Settlement* (SALT) is a conceptual system for a contingent solution that would be able to immediately replace core functionalities of the main Brazilian RTGS in case of its full collapse. Although detailed requirements and conditions for its activation are currently being debated within the Central Bank of Brazil and between Brazilian financial institutions, Distributed Ledger Technology presents great potential for working as foundation for a highly resilient system due to its distributed nature, i.e. a solution that may operate independently of any Central Bank of Brazil infrastructure, nonetheless trustworthy by all parties.

5.1.5. The elected use case

After investigations, SALT was deemed the ideal candidate: its functional requirements are considered rather simple and straight forward; is was possible to choose a subset of core requirements to build a simple, yet fully working, prototype (e.g. throughput capability and liquidity issues were ignored at this stage); and parties were readily available (i.e. commercial banks that participate in the full RTGS are easily within reach of the Central Bank). Furthermore, since there is currently no backup system in place, the experience could be used as foundation for a future production system.

Going a step further, research was concentrated on deciding if features and shortcomings of blockchain technology would favor maintaining a minimal real-time settlement system in place in the case of a catastrophic event taking every Central Bank(CB) datacenter down (thus, the main RTGS system).

Therefore, the initial design of the backup RTGS would be, in this context: a permissioned blockchain network where financial institutions and Central Bank are validating nodes (e.g. miners), using the Internet as intercommunication infrastructure due to its resiliency. In this scenario, nodes would share a common distributed ledger containing the state of every institution's reserves, and they would be able to continue making transactions against each other governed exclusively by smart contracts, in full absence of the Central Bank's supervision. Thus, using blockchain features, the ledger would be automatically guaranteed to register only legitimate transactions, i.e. non-repudiable transactions guaranteed against double-spending.

As for core requirements for the PoC, the following items were selected:

1. A ledger would be created at the start of contingency;
2. Full money quantity would be issued by a Central Bank controlled node on ledger inception;
3. Nodes would be operated by participants in a permissioned network;
4. No money could be created after inception;
5. Wallets would be created for each participant and necessary keys distributed accordingly and securely, preferably (but not mandatorily) without using backchannels. The term *wallets*, in this context, is used in the sense of *conceptual wallets*: whatever existing mechanism from the available platforms through which nodes are able to hold balance and transfer, i.e. not exclusively address wallets (e.g. Bitcoin's);

6. At the outset, each wallet would be attributed a balance by the Central Bank Node;

7. Each participant may or may not host nodes, i.e. participants may use third parties to submit transactions, if a platform allows so;

8. Participants would send or receive money to each other, restricted only by their available balance (i.e. wallets' balances are non-negative);

9. No participant would be allowed to be aware of any other participants' balances;

10. Participants would know their own balance at all times;

11. Central bank node would be able, at any time, to audit transactions and balances from any node;

12. After inception and initial distribution, the system should operate with or without the central bank node, automatically ensuring privacy and preventing double-spending;

13. Once operations are terminated (contingency is over), balances are transferred back to the Central Bank RTGS;

During the PoCs, depending on each platforms' capabilities, some requirements may have been relaxed to accommodate the implementations. For instance, if a particular platform required wallets to be always hosted in a node, item 7 would be ignored.

However, on the outset, the team identified that blockchains' innate data transparency infringes the Central Bank privacy requirements between financial institutions. The simple approach of storing balances in the ledger and approving signed transactions based on coded business logic (including double-spend protection) would not suffice, as it reveals sensitive financial data to all participants. Neither just encrypting sensitive data is a viable solution: for instance, without access to all data, smart contracts on the available platforms are not able to decide whether a transaction is valid. In this case, a possible solution is to add trusted nodes that would be able to see through that information and validate transactions, but, then again, leaving decisions to trusted party would impair the resiliency of the system and undermine the case of using a distributed technology in the first place. The solution to this dilemma proved to be a challenge greater than expected.

Moreover, some key assumptions were used to keep requirements within a feasible set: the existence of a (undetermined at this time) mechanism to *bridge* between the main RTGS and this backup design, which is able to provide enough information from the last RTGS state for executing steps 2, 5 and 6; another (undetermined as well) mechanism to restart the main RTGS consistently using transactions extracted from the blockchain after the last step (contingency deactivation); all nodes would be provisioned at the same time (new nodes wouldn't be allowed to go online during operations); and global liquidity is fixed at the existing total money stock on the main RTGS when operations begin. These assumptions make this a stateless design: the alternative ledger is created assuming main RTGS last state is readily available for its operation, then information is exported to the main RTGS after its restoration and the alternative ledger is destroyed. In real world settings, it is reasonable to assume the ledger is to be created only once and continuously updated by *bridge* mechanisms in order to be able to be activated (and deactivated) seamlessly at any time (a stateful design). However, the team assumed both mechanisms are equivalent for the PoC, since the stateful design may be considered as a particular case of the stateless design: i.e. the latter may be turned into the former simply by dropping the last two assumptions (node provisioning and money issuance). Thus, by assuming *bridge* mechanisms as given, requirements could be kept within reasonable complexity and, at the same time, the prototype would have enough functionality to prove itself useful.

5.2. Prototyping using BlockApps

The team opted for testing exclusively second generation platforms, i.e. smart contract based blockchains (such as *Ethereum*) rather than pure financial transaction based platforms (for instance, *Bitcoin*-based). Here, the rationale is that, although predicting the full extent of its usage might be by itself an impossible task at this point in time, it is reasonable to assume that a future ledger operated within a permissioned network of the financial institutions might accommodate for several applications, since these institutions deal between themselves on distinct levels (e.g. securities markets). On top of perfectly fulfilling the role of pure financial transactions, smart contracts provide added flexibility that makes them suitable for powering most foreseeable applications in a single DLT platform.

Thus, the first prototype was built using *BlockApps*, an Ethereum-based blockchain software development platform. The main driver of this decision was its *REST API*, which allowed the team to circumvent the complexities of the *Ethereum Web3 library* and focus exclusively on coding the use case. However, like in the public Ethereum blockchain, transaction data in *BlockApps* is fully transparent, compromising the desired data privacy requirements.

The first approach to this drawback was a naive solution, adding an off-chain privacy layer that works as follows: as the system starts, the Central Bank of Brazil registers each participant's public key on the blockchain using regulator-only transactions; on a second phase, the central bank uses these keys to initialize and encrypt each participant's starting balance; finally, the regulator also generates, encrypts and stores a symmetric transaction key for each participant pair on the blockchain. Using this procedure, each financial institution is restricted to access only its encrypted sensitive data, and uses its own private keys to decode its own information.

When a participant needs to send money to another: it reads the suitable (encrypted) transaction key from the blockchain; decodes the transaction key using its own private key; encrypts the transaction using the decoded key; then stores the encrypted transaction in the Ethereum blockchain. As the central bank may retrieve transaction keys at any time, it is able to decode every transaction, whereas financial institutions are restricted to decipher only those they were part of (i.e. for which they may retrieve the key from the blockchain using their own private keys).

However, storing encrypted information on the blockchain causes two unintended consequences: first, smart contracts no longer have access to (now encrypted) transaction information, thus becoming unable to prevent overdraft transactions. The workaround to this problem relies on a two-phase commit scheme, which does not completely solve the problem but provides a reasonable balance between privacy and resiliency. Second, this architecture lacks strong forward secrecy, meaning that compromised transaction keys would make possible for unauthorized parties to disclose all transaction history of a participant. This drawback may be attenuated by changing transaction keys periodically, however, asymmetric keys remain a weak spot.

Figure 1 shows a how a transaction is processed in this system: transactions are registered on the blockchain in the *unconfirmed* state. Unconfirmed transactions are considered transfer proposals, and do not reflect immediately on participants' balances. The creation of transactions starts its *verification window*: a configurable period of time (e.g, 10 minutes) when the regulator node may change its state to either *confirmed* or *blocked*. After the the verification window expires, sender institutions may confirm their transactions themselves. Confirmed transactions are final.

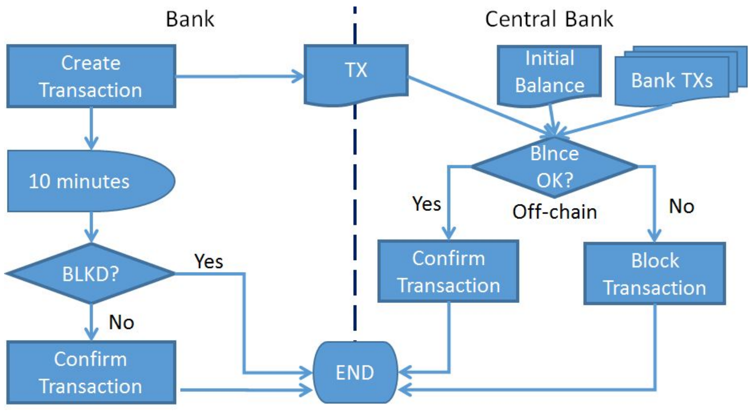


Figure 1: Two Phase Transaction Commit

This design preserves the system against overdrafts when the regulator node is online by providing off-chain, proper balance checks during the verification window. On extreme scenarios (i.e. when the central bank is completely offline), smart contracts are not capable of stopping overdraft transactions by themselves because, as stated previously, transaction data is always fully encrypted.

Although the techniques applied to this prototype do not yield an ideal solution within the stated requirements, this first experiment proves that DLT may already be a tool to allow RTGS members to keep registering transactions while the central node faces outage.

The prototype is available on Central Bank of Brazil Gitlab repository (*BCB Gitlab* [21]), under *rtgsBlockChain*. It contains all functions needed for the system to work, including a monitoring panel for Central Bank teams. The solution is composed by three layers:

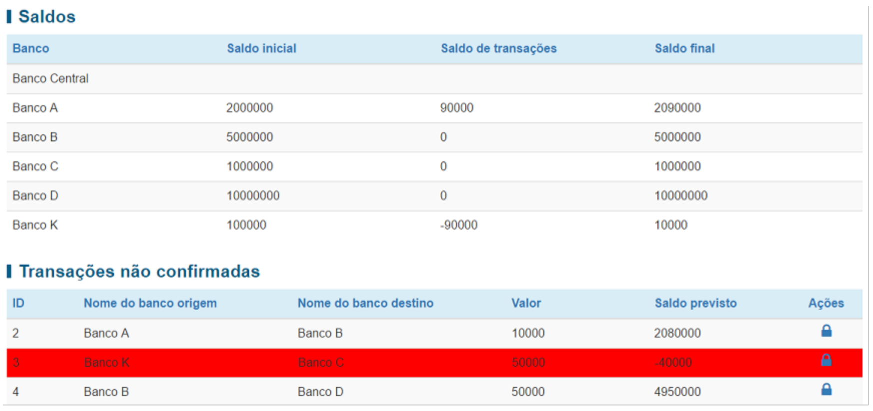
1. *API*: BlockApps specific layer, which facilitates the interaction with smart contracts;
2. *Client app*: web application that communicates with the BlockApps API, and contains the off-chain privacy layer;
3. *Smart contracts*: code to be deployed to BlockApps nodes.

The *API* layer allows using standard web protocols to interact with BlockApps nodes. It contains interfaces for a calling party to deploy smart contracts, create transactions and access the current state of smart contracts. Users that interact with the blockchain need to be created beforehand, and their private keys need to be stored locally. However, the set of interfaces for smart contract interactions is somewhat limited, the team had to extend the default API and create custom endpoints in order to extract additional information from smart contract states.

The *web application* was built using standard web development technologies. For the typical quick approach of a PoC, the team opted for implementing the off-chain privacy layer hosted in the web browser environment, using third party *Javascript* libraries for symmetric and asymmetric cryptography processing. However, this design is far from ideal in production environment, which demands more sophisticated cryptographic mechanisms, like a *Hardware Security Module* (HSM).

Smart contracts were built using *Solidity*, Ethereum's most popular language. The implemented contract allows only its owner (the central bank) to register public keys and transaction keys, as well as starting balances. Moreover, when a participant creates a (encrypted) transaction, contracts store the transaction ID in a mapping, a data structure similar to a *hash table*, in order to facilitate queries later on.

The *monitor panel* is a particularly important piece of the prototype. Using it, central bank users may visually check the current balance of each participant by reading all (encrypted) transactions, decrypting them using transaction keys, and calculating balances. If a financial institution stores a transaction proposal without enough funds to honor it, the central bank node may block that proposal until the participant's liquidity is restored (i.e. the institution receives transfers from other participants). Alternatively, a participant may be completely suspended to operate on the alternative system by the regulator. Figure 2 presents the monitor panel highlighting a transaction proposal that cannot be fulfilled with "Banco K"'s available funds.



Saldos

Banco	Saldo inicial	Saldo de transações	Saldo final
Banco Central			
Banco A	2000000	90000	2090000
Banco B	5000000	0	5000000
Banco C	1000000	0	1000000
Banco D	10000000	0	10000000
Banco K	100000	-90000	10000

Transações não confirmadas

ID	Nome do banco origem	Nome do banco destino	Valor	Saldo previsto	Ações
2	Banco A	Banco B	10000	2080000	🔒
3	Banco K	Banco C	50000	-40000	🔒
4	Banco B	Banco D	50000	4950000	🔒

Figure 2: SALT Prototype Monitor Panel

5.3. Phase 1 Assessment

Final results of the first phase were positive, as the team was able to successfully:

- Identify several use cases that would benefit from its application;
- Develop a working, albeit not ideal, prototype for a selected use case.

Particularly, building a prototype made it clear that DLT was not mature enough to fulfill all requirements, especially when it comes to protecting the privacy of the network's participants. A naive solution was attempted, by registering fully encrypted transactions in the blockchain. However, as discussed, although it is possible to achieve privacy through this method, by blinding contracts to transaction information, the system is no longer able to protect itself against overdrafts, and is still vulnerable to the forward secrecy problems under some conditions.

Throughout the experience, the team also learned that technology companies were trying to figure out how to create business models, and that many working groups were trying to build proof of concepts at the same time around the world, making it particularly hard to get adequate levels of support from suppliers. Furthermore, during interactions with business areas within the Central Bank itself, the research group also detected that lack of maturity was not the only barrier to blockchain adoption: when it comes to this subject, there is currently a lot of bias and misunderstanding among both technical and non-technical crews. The main consequence of this issue is that, once DLT is considered for a project, it yields extra friction on communications between involved parties due to lack of technical discernment.

6. Technical experimentation, phase 2

The second phase of the work focused on analyzing additional platforms in the context of the same use case (SALT). The team chose three platforms to study in this phase: Hyperledger Fabric, Corda and Quorum. As privacy was the unsolved problem in the first phase, the main goal was to seek a platform which allowed the development of a RTGS that agreed with all requirements.

6.1. Hyperledger Fabric

Fabric, led by IBM, is an active project from Linux Foundation's Hyperledger initiative. The main purpose of the framework is to provide a modular architecture, allowing its components to be plug-and-play. Although it is currently available on version 1.0 (since July 2017), the prototype was built on top of a previous version (0.6), the latest available at the time phase 2 took place (January 2017).

Nodes in Fabric 0.6 may be of one of two types: *validating nodes*, which are responsible for executing smart contracts and achieving consensus; *non-validating nodes*, which just have a copy of the blockchain and delegate the execution of smart contracts to validating nodes.

To be part of a Fabric 0.6 network, a node needs to be enrolled using certificates provided by a membership service. After this process, the node registers users through a membership service. Transaction certificates are issued every time an user submits a transaction, which implies pseudo-anonymity, since the relationship between a transaction certificate and an user is known only by the membership service and the user itself. It is worth noting that de-anonymization techniques described on *appendix A* pose a risk to privacy of the participants, specially in smaller networks scenarios.

Consensus is achieved via *Practical Byzantine Fault Tolerance* (PBFT) [22]: a leader is dinamically elected between validating nodes, establishes transaction order and broadcasts them to the remaining nodes; these nodes execute and validate transactions themselves, then broadcast results to the remaining nodes; and once enough nodes acknowledge they agree to the same results, transactions are committed to the ledger. Although it is possible to disable the consensus protocol, it is not recommended.

Fabric 0.6 supports smart contracts written in Java or Go. Unlike Ethereum, which runtime environment (EVM) contains a limited instruction set, Fabric's "virtual machine" is a *Docker* container, with less coding restrictions. This means that it is possible to write non-deterministic code on Fabric, which on its turn might be dangerous in the context of a distributed ledger technology.

Each smart contract contains a state as a set of key-value pairs, and transactions change the state of a smart contract. There are three main functions in a Fabric smart contract: *init* (executed on deploy), *invoke* (changes blockchain state, creates a transaction) and *query* (operates on the current state of the blockchain). Transactions are grouped from two perspectives: Fabric's blockchain is a sequence of transactions linked by hashed blocks; and the *World State* is an auxiliary database that stores the last state of every contract.

Data privacy could not also be fully achieved on Fabric 0.6: node administrators have means to access all information in the blockchain. The REST API embedded in each node allow queries over all blocks, which are just encoded in *Base64* format (i.e. unencrypted). Besides, it is possible to access the World State and view all current information of any smart contract. If privacy requirement is not too strict, a certain level may be achieved through *chaincode policies*, meaning that chaincode query functions will only return data that is relevant to the user. However, this kind of policy needs to be implemented manually by the developer of the smart contract. Further improvement in data privacy would require a secure computing environment.

6.1.1. Hyperledger Fabric 0.6 Prototype

The SALT prototype developed in Hyperledger Fabric 0.6 is available at the *BCB Github* repository [23] under *salt-hyperledger*. It implements the aforementioned relaxed privacy level in chaincode, using *Go* as smart contract language and *Javascript* for the remainder of the code.

The system starts with four financial institutions and the central bank: each institution can see its own balance and transactions; the central bank can see all the transactions and balances. The basic use cases are: checking balance and transferring money from a institution to another.

The prototype can run either in a local blockchain environment, or hosted in a cloud service. The main parts of the prototype are:

1. *REST API*: Exposes a API which talks to the Hyperledger Fabric Client (HFC) library;
2. *Client*: The web application, talks to the REST API;
3. *Chaincode*: The smart contract that is deployed to a Fabric node;

The REST API encapsulates the communication with the Fabric node, which are made by the HFC library, including the issuance of transaction certificates. This decision simplified the development process, making it possible to focus on the smart contract functions and the web application on top of the API.

Users interact with the client layer, a web application. In the prototype, each user belongs to a financial institution and acts in its name, and user information is stored in contract states. It seems a better approach would be using membership certificates to validate users, however the team could not find documentation on how to do it at the time.

6.1.2. Fabric 0.6 Assessment

Overall, the final result was satisfactory: the platform architecture was understood, and the proof of concept was built. The team hopes to be able to improve the prototype using the 1.0 version of Fabric in a near future, and use new promising privacy features, like channels and endorsement policies.

However, it was found that Fabric 0.6 has the same limitations of *Ethereum* about privacy: there is a need of an additional off-chain layer to create private transactions. The imminence of 1.0 release made it hard to find good documentation about how to work with the 0.6 version, as the community was focused on the new release.

6.2. Corda

Corda [24] is a blockchain inspired open source permissioned distributed ledger platform built with focus on financial industry, that was created and is still being developed mostly by a company named *R3* [9]. A Corda network is made up of nodes running Corda and CorDapps (Corda's smart contract), one or more notaries and zero or more oracles, i.e. a third party service that provides external data onto the blockchain.

In Corda, there is a global ledger of transactions but, unlike other blockchain based platforms, there is no globally shared blockchain: instead, each participant node in the network stores a fraction of the distributed ledger. This fraction corresponds to the set of committed transactions in which the hosting node has participated. This way, two peers will see the same version of any on-ledger transactions they share, but peers will not have access to transactions they are not part of.

A transaction in Corda is a proposal to update the ledger, with input and output states. If the transaction is valid and committed, it will consume active states as inputs and it will produce, as output, new active states. Consumed states are no longer available for new transactions: thus, active states are just *unspent transaction outputs* (UTXO). A transaction is valid if it doesn't contain double-spends, all of its states' contracts are accepted, and it is signed by the required parties.

Corda contracts are functions written in a JVM programming language (e.g. Java or Kotlin) that have deterministic execution and are tied to states. It will either accept or reject a transaction based exclusively on its contents. But contracts have limitations, they can only check transactions for internal validity. A contract can not check, for example, if a transaction is in accordance with what was originally agreed with the counterparties. Therefore, peers should check the contents of a transaction before signing it, even if the transaction is contractually valid.

To fulfill this requirement, among others, Corda has the concept of flows: Flows are the part of CorDapps that coordinate communications between nodes and automate the process of agreeing ledger updates. Flows have steps, subflows and checkpoints, they work as persistent state machines that may last days or more. All communications in Corda occurs inside a flow context, and are point-to-point by default.

The notary is a special entity whose main responsibility is to prevent double-spends by granting the uniqueness of each transaction that goes in the ledger. Each notary can run stand alone or in consensus with other notary instances to accomplish its goals. Almost all transactions in Corda need to be notarized and the notary will only sign a transaction if it is responsible for all the transaction's input states. The need of a notary to grant uniqueness of each transaction assigns specific requirements to the notary itself:

1. *Unbreakable*: if the notary is offline, all the active states assigned to it will be blocked for usage, as the node cannot prove the absence of double spends.

2. *Neutral*: as the notary sees all transactions it notarizes, it needs to be a neutral party.

3. *Trustful*: if the notary is breached by an unauthorized party, all system trust is compromised.

4. *Flawless*: if the notary make mistakes in its operation, all system trust is compromised.

5. *Fast*: if the notary isn't fast enough, the system performance is compromised.

6.2.1. Corda Assessment

After some development and testing, the team decided not to go further in constructing a fully functional SALT prototype at that time because the code base was somehow immature: during the lab period, Corda went progressively from version 0.6 to version 0.9.2, suffering various changes on API and functionalities. Furthermore, the platform still missed some important features related to reliability and privacy:

1. *Disaster recovery*: In other platforms (e.g. Fabric or Quorum), if a node breaks and needs to be replaced for a pristine node instance, the normal behavior of the network is to provide the new node all the data it needs to get synchronized and running as soon as possible. In Corda, as mentioned earlier, there is no public blockchain or private channels, each transaction occurs in a peer-to-peer manner. Without a disaster recovery feature built in the platform, the process of retrieving data from the network would be cumbersome and risky. As resilience is the main purpose of SALT project, this feature really plays an important role in the system, and some disaster recovery capabilities are expected in future releases of Corda;

2. *Notary customization*: The early versions of Corda had only two options of notary behavior implemented and included in the platform: validating and non-validating notary. Those available implementations were too simple to meet SALT's monitoring and availability requirements, and no other behavior or customization was possible to the notary node. Later versions included RAFT and PBFT notaries (which solve availability issues) and also made it possible to execute a fully customized notary to meet other business needs.

3. *Privacy of transaction history*: In a project like SALT, assets should travel freely among the few well-known participants, possibly creating a big transaction history chain in a short period of time. The free circulation of the historical chain linked to each transaction, necessary for the transaction validation, is a breach in the privacy control of the balances and transactions between the nodes. However, Corda's transaction tear-off is a privacy technique used in R3's platform to hide information from non-validating notaries or oracles that consists on creating a Partial Merkle Tree with omitted branches, leaving just the branch root hash on its place. This way, only partial transaction data is exposed to external actors while proving the entire transaction integrity [25]. Recently, R3 announced a partnership with Intel to add a feature called remote attestation to CORDA platform, based on Intel's Software Guard Extensions technology (SGX) [26], this feature seems promising in order to solve the transaction validation without breaking the privacy of the transactions' history chain.

6.3. Quorum

Quorum is an Ethereum-based permissioned blockchain platform that extends the original protocol with privacy capabilities, multiple consensus mechanisms and peer permissions management [27]. These enhancements drew attention to implementing SALT prototype using this platform, hoping to solve the transaction privacy and double-spend protection dilemma.

All data stored on the Ethereum blockchain is public and is the result of the consensus algorithm, so there is only one global state shared between participants. Quorum extends this design creating private transactions, a new way to securely exchange data between nodes. Private transactions are encrypted and addressed to specific nodes on the network, so there is no global knowledge of its contents. Nodes that are not part of the transaction only receive hashes of the encrypted private transactions data that is used to complete the Global Transaction Trie, which root is stored on each block. Using that block information, nodes can verify if they are synchronized with the rest of the network without accessing third parties' private transaction content. This approach is similar to the Corda's transaction tear-off aforementioned in section 6.2.1.

As in Ethereum, transactions may create smart contracts. When a private transaction creates a contract, it is stored privately in a local state database. The private state is not part of the blockchain, not subject to global validation rules and, consequently, interaction between private and public contracts are limited. For example, it is not possible for a private contract to change the state of a public contract, however private contracts can read data stored on public contracts.

Ethereum's proof of work (Ethash) is not suitable for a permissioned environment, where network participants are known *ex ante*, and subject to a regulator's authority. That way, Quorum defines a new consensus algorithm called QuorumChain, and leaves the consensus mechanism exchangeable.

QuorumChain is a time-based majority-voting consensus algorithm that defines three roles to every node: maker, voter and observer. Blocks are created at a bounded random interval by the block makers and submitted to the network, where voters analyze the proposed transaction list and compare the resulting public state root hash to the new block's. It is important to note that private transactions are only validated (and voted upon) by the respective parties, while the others just compare the private transaction hash list [28].

While QuorumChain proposes a clever alternative to Ethash, it is not suitable to most payment systems that require a legal settlement finality with the guarantee that transfers are irrevocable. As observed by Project Jasper's team, settlement is probabilistic when using Proof of Work consensus algorithms, because network participants can always agree on an alternate history of the transactions, and create a fork on a blockchain's past block. The same problem is present on QuorumChain [29].

Fortunately, a Raft-based consensus mechanism is available on the Quorum implementation that claims to eliminate the possibility of blockchain forking. Unlike QuorumChain, Quorum-Raft nodes can operate on one of two available roles: leader and follower. There is only one elected leader at a time on the network, and only the leader can produce new blocks. Newly created blocks are proposed to the Quorum-Raft network and, once a consensus about its validity emerges, the new block is irrevocably appended to the blockchain.

Quorum-Raft was not tested on the scope of this work, however it is supposedly adequate for applications that require absolute settlement finality [30].

6.3.1. Quorum Prototype

Programming in Quorum is very similar to Ethereum, they both run the same virtual machine (Ethereum Virtual Machine - EVM) and are compatible with the same smart contract language (Solidity), making it a very good option to developers with Ethereum programming experience. However, when dealing with the same private contracts on different nodes, program logic may be completely different because contracts of this kind are expected to be in different states across the network. The team used variables on the prototype contract code to guide the program logic through distinct paths on different nodes. For example, minimal sender balance check can be executed on the regulator's node, as it supposedly stores all sensitive information of the network participants. Nevertheless, the contract should skip this validation on recipients nodes, obviously because the sender's balance is not stored locally.

The SALT prototype developed in this work is available at *BCB Github* repository [23] under *quorum-examples*. Two smart contracts were developed on a Quorum test environment to implement a basic blockchain RTGS with privacy capabilities:

1. *bankContract*: This contract implements the main logic of the RTGS system. One *private* instance of the contract must be deployed for *each* participating institution on the network and it will store private transaction data and balance. It has many verification steps to avoid institution

personification, fraud and overdraft transactions. Most of the checks are made by the contract itself, but the overdraft protection must be made by an active regulator node;

2. *regulatorTransactionList*: This contract stores and provides the public data to the private contracts, mainly transactions hashes and participating institutions information.

The private contracts store transfers details created by transactions that must only be sent to the recipient institution's contract and to the regulator node using the "privateFor" parameter. As in previous prototypes, transfers are created in an unconfirmed state and are only final if they are confirmed by the regulator or by the sender institution following the same two-phase commit process explained previously.

After publishing the private transaction, the sender institution must publish the same transfer hash in the public transaction log (*regulatorTransactionList* public contract) to ensure that all parts are aware of the transaction. Without this information posted on the public contract, a participant could create an overdraft transfer without addressing the regulator node on the private Quorum transaction, making it impossible for the regulator to block the offending transfer. All private smart contracts enforce this rule before confirming a transaction.

6.3.2. Quorum Assessment

The first advantage seen on Quorum is the Ethereum similarity. One of the objectives of the project is to reuse as much as Ethereum protocol as possible, making it simpler to maintain the platform aligned with innovations introduced to Ethereum [27].

Considering that Ethereum is the oldest and largest public smart contract platforms in production, Quorum benefits from the past and, potentially, future developments on the platform, as they can, in thesis, be merged into the Quorum code. Moreover, smart contract development cycle is basically the same of Ethereum's, but, as stated before, private smart contracts behave completely different.

Ethereum base code tested in the wild for more that two years is expected to be reasonably secure and compliant with permissioned network requirements, however it is noteworthy that the new code introduced to the Ethereum base may bring unexpected security vulnerabilities to the network, specially in the all-new secure messaging code. The security benefits of the similarity extend even beyond the platform base code. Ethereum smart contracts hold millions of dollars on cryptocurrency in custody on the public blockchain nowadays and, unfortunately, virtual heists numbers are growing due to vulnerabilities on the smart contract Solidity code. The Distributed Autonomous Organization and Parity's multisignature wallet are examples of how costly these mistakes can be [31][32], however these attacks are attracting extensive research on how to create secure smart contracts with the Solidity language [33][34].

Quorum privacy capability removes the need to broadcast ciphered sensitive data to every node, however, consensus on the private state is not explicit. QuorumChain documentation states that *"the private state consensus is implicit through a combination of provably synchronized contract inputs (global Transaction Hash validation check), a provably deterministic EVM (public state validation check), and provable chain synchronization (new blocks only added to the canonical chain)"* [28]. Nevertheless, the validation to this assumption requires off-chain processes.

Moreover, it suggests the implementation of an off-chain private state root hash comparison. However, in many use cases the state of the same private contract deployed on different nodes will be different by design. This is clearly the case of the Quorum prototype developed during this research, as follows: A's contract on B's node only stores transactions between A and B. Logically, this contract instance is in a different state than A's contract on A's node that stores every transaction of A, so private states root hashes are not comparable.

In summary, the developed Quorum prototype achieved the privacy requisite at the expense of the double-spending prevention. The bankContract code cannot guarantee the minimum available funds to honor the transfer as sender balances are not present on the recipient nodes. In practical terms, the system still relies on a trusted regulator node to detect and prevent double-spending using the two-phase transaction commit on the Quorum prototype.

Improvement on this matter are expected in the near future with the announced partnership of Zerocoin Electric Coin Company and JPMorgan Chase to integrate zero-knowledge proof technology on Quorum platform[35]. As explained in *appendix A*, this technology can prove that a sender has enough balance to fulfill a proposed transaction without the attestation of a regulator.

7. Discussion and future work

Pseudo-anonymity is the effective privacy level obtained by most cryptocurrency systems in the public blockchain case (e.g. Bitcoin, Ethereum). However, privacy is a desired feature in permissioned networks such as a national level RTGS (i.e. SALT). This kind of network would contain much less participants: in the aforementioned Brazilian case, it would be used by roughly two hundred transactors, whereas a public network such as bitcoin's operate over tens of million of wallets. In such a restricted set, each participant will inevitably interact with many peers and be able to acquire a great deal of sensitive data in a short period of time, hence soon being able to extract the transaction graph for a large part of the network. Such capability could affect the whole financial system by providing participants with otherwise private information about their peers and deals, facilitate spurious market transactions and compromise fair competition.

Privacy is also a desirable feature in the other use cases examined by the team: in identity management cases (section 5.1.1), subjects would like to be able to have a substantial level of control of who may be authorized to view or use their private data; SML and CCR both are, conceptually, payment systems between nations, and it is hard to conceive that participating central banks would allow each other to be aware of their country's full transaction history.

Thus, throughout this work, the team acquired the concept that, although that adopting DLT would bring about several benefits, the technology is not viable for some use cases unless adequate levels of privacy are achieved. Furthermore, the team concluded that, currently, such levels are not fully supported on the four explored platforms with true decentralization, i.e., without relying on a trusted node or party.

The naive first (experimental and implementable) solution was to fully encrypt transactions on the blockchain. However, as discussed previously, this approach presents three major drawbacks: it cripples the ability of the blockchain to enforce smart contract rules; it undermines the forward secrecy property (transaction history may be disclosed under certain key compromise conditions); and it requires off-chain processes to enforce processes that would otherwise be carried out by smart contracts, which requires additional trusted nodes, weakening the resiliency of the ledger.

The survey in *appendix A* presents a summary of research on techniques that could increase the privacy level of DLT: one possible approach to achieve anonymity would be using *mixes* to allow participants to "wash" coins for their transactions. However, as discussed in the *appendix*, this technique also requires undesirable trusted third party nodes. Essentially, for this use case, the Central Bank of Brazil would be a reliable trust party to fulfill this role. However, SALT was conceived to be contingency for situations for which the Central Bank is unable to operate, thus it can't be counted on. For the other payment use cases, however, it might be considered.

Further along, it will be possible to use *ring signatures* or *zero-knowledge proof* techniques to tackle this issue. *Zero-knowledge proof*, as proposed in *Zerocash*, is a promising feature for allowing fully encrypted, yet validated, transactions. However, it brings an undesired consequence: as designed, participants would be able to hide information even from regulators. This level of privacy might not be desirable due to making it impossible for central banks to be aware of participant's states (e.g. liquidity level) by simply inspecting the ledger, i.e. without counting on the latter to give up information voluntarily.

Most of the analyzed technologies have some type of privacy feature planned for next versions: IBM (Fabric's main contributor) claims that full transaction privacy may be achieved by using *channels* between transactors, starting from version 1.0; Corda provides an adequate level of privacy by keeping separate ledgers for each node, but it lacks resiliency when it relies on notaries to achieve uniqueness consensus; and zero-knowledge proof functionality is currently being added to Quorum. Though no implementation presented adequate balance between features like privacy and resiliency at this point, it is reasonable to assume that, by adding recently proposed techniques, this problem may be solved in a short time frame.

Once enhancements are in place to allow for fully distributed private transaction, there might be need of investigating a collateral issue: it seems that privacy, reliability and performance are intertwined, in the sense that, by privileging one of these features, one should expect to hurt another. For instance, zero-knowledge proof provides higher privacy levels, but current implementations hint that it hurts the performance of blockchains by demanding an excessive amount of computation to run its algorithms [36]. Corda's separate ledgers are another way to provide privacy, but it yields less reliability since data is not replicated among all nodes, such as in the classical blockchain model. Mixes and the naive encryption approach present both problems: by adding off-chain processes, i.e. invoking external services to validate the uniqueness of a transaction, both performance (by latency penalty, due to the additional hop) and reliability (off-chain nodes are generally centralizing points of failure) are harmed. Thus, further experiments should investigate the ideal balance between these characteristics, and move on to issues that were purposely left out in this work, such as mechanisms to deal with liquidity risks.

8. Conclusion

This Central Bank of Brazil study on blockchain technology applied to an interbank payment scheme is very similar to other central banks efforts, but it carries one important difference: the main goal is not to substitute or provide a main settlement system, but to create a minimal funds transfer system to the financial sector in case of a complete main RTGS meltdown. As a last line of support, developing a complex system with liquidity-saving mechanism or full support of all messages available in the main system was out of scope, the intent was simply to create a minimal infrastructure for the financial system to endure during a severe regulator outage. Another difference is that this study is strictly technical, so there is no intention of analyzing whether a backup RTGS system may yield economic consequences and, if so, the extent of such impact.

By nature, the value of this technology is intimately tied to the network effect. Partnerships with another government agencies and private enterprises may also be very important: many new possibilities emerge when it comes to share information between agencies, with large foreseeable benefits to society. The Distributed Ledger Technology could make possible to create an unique shared view of a large variety of information fed and replicated across institutions.

However, privacy is still the main challenge: along these experiments, it was not possible to achieve privacy without giving up consensus. There are encouraging solutions in this field, like zero-knowledge proofs and secure computing enclaves (e.g. Intel SGX). More investigation is needed to verify if these options will be able to settle this issue, and the evaluation of other innovative approaches should be in order as well.

References

[1] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.

[2] Sunny King and Scott Nadal. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. *self-published paper, August*, 19, 2012.

[3] Smart oracles: A simple, powerful approach to smart contracts. https://ripple.com/. [Online; accessed 9-August-2017].

[4] Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Project Yellow Paper*, 151, 2014.

[5] Nick Szabo. Formalizing and securing relationships on public networks. *First Monday*, 2(9), 1997.

[6] Digital gold 'done right' with digixdao crypto-trading on openledger. https://www.forbes.com/sites/rogeraitken/2016/04/23/digital-gold-done-right-with-digixdao-crypto-trading-on-openledger/#1f701e7c5204. [Online; accessed 11-August-2017].

[7] Hyperledger.org. https://www.hyperledger.org/. [Online; accessed 11-August-2017].

[8] Why j.p. morgan chase is building a blockchain on ethereum. http://fortune.com/2016/10/04/jp-morgan-chase-blockchain-ethereum-quorum/. [Online; accessed 11-August-2017].

[9] R3 consortium website. https://www.r3.com/about/. [Online; accessed 11-August-2017].

[10] Introducing r3 corda: A distributed ledger designed for financial services. https://www.r3cev.com/blog/2016/4/4/introducing-r3-corda-a-distributed-ledger-designed-for-financial-services. [Online; accessed 11-August-2017].

[11] Deloitte Consulting LLP. Blockchain: Opportunities for health care. https://www2.deloitte.com/us/en/pages/public-sector/articles/blockchain-opportunities-for-health-care.html. [Online; accessed 22-August-2017].

[12] James Chapman, Rodney Garratt, Scott Hendry, Andrew McCormack, and Wade McMahon. Project jasper: Are distributed wholesale payment systems feasible yet? *Financial System*, 2017.

[13] Trevor I Kiviat. Beyond bitcoin: Issues in regulating blockchain tranactions. *Duke LJ*, 65:569, 2015.

[14] Darshini Dalal, Stanley Yong, and Antony Lewis. Project ubin: Sgd on distributed ledger. *self-published paper*, 2016.

[15] Creer David, Richard Crook, Mark Hornsby, Nicolás González Avalis, Mark Simpson, Nick Weisfeld, Ben Wyeth, and Ivo Zieliński. Proving ethereum for the clearing use case. *self-published paper*, 2016.

[16] Ciab febraban. http://www.ciab.org.br/publicacoes/edicao/69/ciab-febraban-apresenta-testes-com-blockchain. [Online; accessed 3-August-2017].

[17] Central Bank of Brazil. Sml - local currency payment system. http://www.bcb.gov.br/rex/sml/ingl/faq.asp. [Online; accessed 21-August-2017].

[18] Stefan Thomas and Evan Schwartz. A protocol for interledger payments. *URL https://interledger. org/interledger. pdf*, 2015.

[19] Bank of England. Ripple - exploring the synchronised settlement of payments using the interledger protocol. http://www.bankofengland.co.uk/Documents/fintech/ripplepoc.pdf. [Online; accessed 2-August-2017].

[20] Bank for International Settlements. Principles for financial market infrastructures. http://www.bis.org/cpmi/publ/d101a.pdf. [Online; accessed 4-August-2017].

[21] Central bank of brazil gitlab repository. https://gitlab.com/bacen. [Online; accessed 1-August-2017].

[22] Miguel Castro, Barbara Liskov, et al. Practical byzantine fault tolerance. In *OSDI*, volume 99, pages 173–186, 1999.

[23] Central bank of brazil github repository. https://github.com/bacen. [Online; accessed 1-August-2017].

[24] Corda. https://www.corda.net/. [Online; accessed 16-August-2017].

[25] Corda transaction tear-offs. https://docs.corda.net/releases/release-M14.0/key-concepts-oracles.html#transaction-tear-offs. [Online; accessed 10-August-2017].

[26] Corda and sgx: a privacy update. https://www.corda.net/2017/06/corda-sgx-privacy-update/. [Online; accessed 16-August-2017].

[27] Quorum whitepaper. https://github.com/jpmorganchase/quorum-docs/blob/master/Quorum%20Whitepaper%20v0.1.pdf. [Online; accessed 1-August-2017].

[28] Quorumchain consensus. https://github.com/jpmorganchase/quorum/wiki/QuorumChain-Consensus. [Online; accessed 4-August-2017].

[29] Quorum faq. https://github.com/jpmorganchase/quorum/wiki/FAQ. [Online; accessed 4-August-2017].

[30] Raft-based consensus for ethereum/quorum. https://github.com/jpmorganchase/quorum/blob/master/raft/doc.md. [Online; accessed 4-August-2017].

[31] Vitalik Buterin. Critical update re: Dao vulnerability. https://blog.ethereum.org/2016/06/17/critical-update-re-dao-vulnerability/. [Online; accessed 4-August-2017].

[32] Security alert - critical bug in parity's multisig-wallet. https://blog.parity.io/security-alert-high-2/. [Online; accessed 4-August-2017].

[33] Karthikeyan Bhargavan, Antoine Delignat-Lavaud, Cédric Fournet, Anitha Gollamudi, Georges Gonthier, Nadim Kobeissi, A Rastogi, T Sibut-Pinote, N Swamy, and S Zanella-Beguelin. Formal verification of smart contracts. In *Proceedings of the 2016 ACM Workshop on Programming Languages and Analysis for Security-PLAS'16*, pages 91–96, 2016.

[34] Nicola Atzei, Massimo Bartoletti, and Tiziana Cimoli. A survey of attacks on ethereum smart contracts (sok). In *International Conference on Principles of Security and Trust*, pages 164–186. Springer, 2017.

[35] Laura Shin. Jpmorgan chase to integrate zcash technology to its enterprise blockchain platform. https://www.forbes.com/sites/laurashin/2017/05/22/jpmorgan-chase-to-integrate-zcash-technology-to-its-enterprise-blockchain-platform/. [Online; accessed 4-August-2017].

[36] George Danezis, Cedric Fournet, Markulf Kohlweiss, and Bryan Parno. Pinocchio coin: building zerocoin from a succinct pairing-based proof system. In *Proceedings of the First ACM workshop on Language support for privacy-enhancing technologies*, pages 27–30. ACM, 2013.

[37] George Danezis and Sarah Meiklejohn. Centrally banked cryptocurrencies. *arXiv preprint arXiv:1505.06895*, 2015.

[38] David Chaum. Blind signatures for untraceable payments. In *Advances in cryptology*, pages 199–203. Springer, 1983.

[39] David Chaum, Amos Fiat, and Moni Naor. Untraceable electronic cash. In *Proceedings on Advances in cryptology*, pages 319–327. Springer-Verlag New York, Inc., 1990.

[40] Sarah Meiklejohn, Marjori Pomarole, Grant Jordan, Kirill Levchenko, Damon McCoy, Geoffrey M Voelker, and Stefan Savage. A fistful of bitcoins: characterizing payments among men with no names. In *Proceedings of the 2013 conference on Internet measurement conference*, pages 127–140. ACM, 2013.

[41] Alex Biryukov, Dmitry Khovratovich, and Ivan Pustogarov. Deanonymisation of clients in bitcoin p2p network. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 15–29. ACM, 2014.

[42] David L Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–90, 1981.

[43] Malte Möser. Anonymity of bitcoin transactions. In *Münster bitcoin conference*, pages 17–18, 2013.

[44] Joseph Bonneau, Arvind Narayanan, Andrew Miller, Jeremy Clark, Joshua A Kroll, and Edward W Felten. Mixcoin: Anonymity for bitcoin with accountable mixes. In *International Conference on Financial Cryptography and Data Security*, pages 486–504. Springer, 2014.

[45] Ronald Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. *Advances in Cryptology—ASIACRYPT 2001*, pages 552–565, 2001.

[46] David Chaum, Ronald L Rivest, and Alan T Sherman. Advances in cryptology. In *Proceedings of CRYPTO*, volume 82, pages 279–303. Springer, 1983.

[47] Eiichiro Fujisaki and Koutarou Suzuki. Traceable ring signature. In *Public Key Cryptography*, volume 4450, pages 181–200. Springer, 2007.

[48] Nicolas van Saberhagen. Cryptonote v 2. 0, 2013.

[49] Shen Noether. Ring signature confidential transactions for monero. *IACR Cryptology ePrint Archive*, 2015:1098, 2015.

[50] Ian Miers, Christina Garman, Matthew Green, and Aviel D Rubin. Zerocoin: Anonymous distributed e-cash from bitcoin. In *Security and Privacy (SP), 2013 IEEE Symposium on*, pages 397–411. IEEE, 2013.

[51] Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. *Advances in Cryptology—EUROCRYPT 2001*, pages 93–118, 2001.

[52] Christina Garman, Matthew Green, and Ian Miers. Accountable privacy for decentralized anonymous payments. In *International Conference on Financial Cryptography and Data Security*, pages 81–98. Springer, 2016.

[53] Ahmed Kosba, Andrew Miller, Elaine Shi, Zikai Wen, and Charalampos Papamanthou. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In *Security and Privacy (SP), 2016 IEEE Symposium on*, pages 839–858. IEEE, 2016.

Appendix A - Privacy in a DLT environment

Previous monetary authorities efforts refer to a common issue: privacy. Project Jasper's research paper clearly expresses this requirement as *fundamental* for a wholesale payment system to protect participants sensitive information. Likewise, Bank of Canada's research team concludes that most DLT platforms suffer from excess data transparency [12]. Project Ubin's document reinforces this belief, stating that DLTs offer the potential to improve domestic and cross-border transactions only if it can bypass serious technical shortcomings, like full transparency. One of the solutions proposed by Ubin's researchers is to secure data in blocks with encryption, a technique explored by the Central Bank of Brazil (detailed further in this document). Furthermore, MAS team was the only authority that publicized exploring Quorum platform privacy capabilities, although its latest publication does not offer any detailed explanation of whether they were able to avoid double-spend attacks, and how so. They are currently conducting phase 2 of their research, and their conclusions on the matter are expected to be public in the near future [14].

Since Bank of Scotland's Emerald aims particularly on performance testing and DLT evaluation to discover its limits, there is no research on data confidentiality on this project [15]. Danezis and Meiklejohn's research on RSCoin [37] follows the same rationale, focusing in providing a cryptocurrency framework in which monetary policy is still controlled by a central bank, not by a consensus algorithm. Thus, both leave privacy for a future work.

Unlike other monetary authorities, the Central Bank of Brazil's team focused specifically on the antagonism between blockchain's transparency and the desired confidentiality during the latter part of the experiments, leading to a in-depth research about available data obfuscation technologies, which follows.

One of the first efforts in digital money was developed by David Chaum using a technique called *blind signatures* in the early 80's. It was a major breakthrough that allowed the generation of digital currency *"notes"* by the customer, blindly signed by the bank while debiting the customer account [38]. However, the system relies on a central party to detect and prevent double spending. Several enhancements where developed on top of this clever cryptographic system, including a probabilistic double-spending detection and identification of the offending party[39].

As described in its seminal work [1], Bitcoin's original design include source and destination addresses in every transaction, in clear form. It relies, therefore, on the fact that blockchain addresses carry no information about its owners. At this moment, most blockchains follow this design, thus making them *pseudonymous* networks: once an observer is able to identify an user's address, it may trace his entire transaction history. In [40], for instance, authors were able to analyze the Bitcoin transaction history, use heuristics to uncover evidences to cluster addresses by ownership, identifying balances held by some major institutions. In [41], authors go further along and present a series of technical steps that allow attackers to determine the network address of a node in the Bitcoin P2P network, effectively deanonymizing bitcoin addresses of transactors.

The first solution to this problem are *Mixes*. They were first proposed by Chaum [42], in order to assure anonymous communication between two parties by encrypting a number of input messages and sending the decrypted output to receivers, effectively decoupling input and output addresses. Cryptocurrency mixes work by the same principle: users achieve anonymity by sending a value to a shared wallet, and receive it back in fresh addresses (possibly decreased by a service fee) [43].

In Mixcoin [44], a third party is used to anonymize cryptocurrency by receiving coins from several parties and transferring them to fresh, unidentifiable addresses, conceivably by chaining mixes. In this protocol, mixes sign receipts of the coins they receive, so that clients may keep them honest, and charge randomized fees from clients in order to keep output untraceable. However, although the protocol guarantees anonymized output, attackers may still link addresses by analyzing mixing timestamps and spending behavior.

By delaying the payout until a substantial number of inputs have been sent to the shared walled, mixes may yield an adequate level of anonymity for users. Nevertheless, by observing inputs and outputs in the public transaction record of a blockchain, attackers may still infer the correlation between two addresses and, in some cases, successfully identify their ownership. Moreover, this technique requires putting trust on mix service providers, since they would be able to retain users' balances if behaving dishonestly, and they need to keep (at least temporarily) records on both inputs and outputs to do their work, therefore being susceptible to leaks that would compromise anonymity. Authors in [43] analyzed the obtained anonymity degree of three popular Bitcoin mix services in 2013, and found out that only one of them worked at adequate levels.

Ring signatures were first proposed by Rivest et al [45]. In this authentication scheme, individuals in a group may generate digital signatures which can prove that a message was originated by someone in that group, but can't be traced to a specific participant. Although they resemble group signatures as proposed by Chaum [46], they do not rely on any central node, thus guaranteeing full anonymity of the signature's author. Traceable ring signatures are an extension by Fijisaki and Suzuki [47] that detect if the same participant uses the signature twice for the same object. Using this technique, van Saberhagen proposes modifications in the Bitcoin blockchain in order to make transactions unlinkable [48]: transactions would be sent to one-time public keys, generated using the recipient's public address. Using his own information and data in the transaction body, the recipient is able to recover a one-time private key that unlocks the transactions; he selects a random subset of other users and, using their public keys and the recovered private key to generate a ring signature, is able to sign and spend the incoming transaction. On this scheme, attackers can't recover the signer's identity from the signature because they don't have the private information to calculate her public key, and double spending is prevented, since the (traceable) ring signatures may be used only once for each unspent transaction. However, by design, signing groups are restricted to transactions of the same amount, thus transactions of less common values may possibly be linked to a small group of users. Monero [49] extends this scheme to overcome this drawback, and adds features for allowing hidden amounts, origins, and destinations for transactions.

Miers et al [50] propose a distributed e-cash system that add an anonymization feature to a blockchain without the need of a trusted third party by using zero-knowledge cryptographic proofs, *Zerocoin*. In this scheme, users mint new coins by transferring their own coins to a public escrow, and receive a private secret that allow them to produce a zero-knowledge proof of her deposit (i.e. a *certificate of deposit*). Using this proof, users may spend values equivalent to their deposit from any of the coins in the escrow, which are effectively unlinked from their own addresses, thereby obfuscating their origin. Hence, Zerocoin may be defined as a decentralized mix technology. By modifying the bitcoin source code, authors proved that the scheme is feasible, but would increase substantially the network's computational cost at its current state. In [36], this performance issue in the protocol was later mitigated by replacing cryptographic algorithms. Moreover, authors note the protocol could be further modified in order to provide anonymity while allowing user accountability by using, for instance, anonymous credentials [51].

Zerocash goes a step further, providing a fully anonymous decentralized payment scheme with strong guarantees and enhanced performance. It uses *zero-knowledge Succinct Non-interactive ARguments of Knowledge* (zk-SNARKs) to create a layer of *shielded transactions* on top of existing blockchain technologies, by allowing users to mint new coins (in a similar fashion to Zerocoin) deposited in shielded addresses. Users may then transfer these coins to other shielded addresses (and, alternatively, to public addresses) of its base blockchain. Although general data (such as timestamps) may be seen in the public blockchain records, third parties may not extract any information about *shielded transaction* (neither origin and destination addresses, or value), with the obvious exception of destination information on transfers to public addresses. Aiming regulatory purposes, [52] proposes a set of extensions to guarantee compliance and allow selective user and tainted coin tracing.

All the aforementioned zero-knowledge schemes, however, may be applied exclusively for purely monetary transactions, they don't allow the programmability in a blockchain (neither scripts, such as Bitcoin's, nor smart contracts). Hawk [53] proposes a decentralized smart contract system that keeps transactional data private from the public blockchain records: users commit newly minted private coins and commit them to a contract by presenting a zero-knowledge proof; then they send their private inputs to a minimally trusted party (i.e. a *manager*), who performs off-chain computations to validate these inputs, determine payout distribution and generate a zero-knowledge proof of the contract's outcome; after attesting the outcome validity, the blockchain generates output transactions as specified in the contract, as well as redistributes any unused collateral and fees. Although it has access to private inputs, a dishonest manager can't affect outcome and redistribution of a running contract, even when in collusion with a subset of participants, and its misbehavior may be automatically detected and penalized accordingly by the system. Authors describe practical applications of this system in scenarios like second-price auction, crowdfunding and financial swap instruments.