sciendo

BlockChain Platforms in Financial Services: Current Perspective

Pablo Garcia Bringas, Iker Pastor-López
University of Deusto, Bilbao-Spain
Giuseppe Psaila
University of Bergamo, Bergamo-Italy

Abstract

Background: BlockChain technology was invented to support *bitcoin*, currently the most popular virtual currency. **Objectives:** The purpose of this paper is to investigate contemporary BlockChain platforms in financial services. **Methods/Approach:** An unstructured literature review has been used. **Results:** BlockChain in financial services is mostly associated with *bitcoin* exchange. However, this is a partial view of both BlockChain technology and its possible adoption for financial services: in fact, many BlockChain platforms are now available and many different financial services can be effectively supported by BlockChain platforms, even though they are not based on virtual-money exchange. Furthermore, people are attracted by the concept of *smart contract*, i.e., a contract that is automatically executed by computer technology, without human intervention. **Conclusions:** The contribution of this paper is twofold: first of all, we introduce the four BlockChain platforms that are now most popular, discussing how they support the smart contract concept; second, we identify some typical categories of financial services, matching each of them with the platform that provides the best support for each category.

Keywords: BlockChain history, BlockChain characterization, smart contracts, perspective BlockChain in financial services.
JEL classification: L86
Paper type: Research article

Received: Feb 25, 2020
Accepted: Jul 06, 2020

Citation: Garcia Bringas, P., Pastor-Lopez, I., Psaila, G., (2020), "BlockChain Platforms in Financial Services: Current Perspective", Business Systems Research, Vol 11 No 3, pp. 110-126.
DOI: https://doi.org/10.2478/bsrj-2020-0030

Introduction

The acronym DLT, which stands for *Distributed Ledger Technology*, has become quite popular. However, what is its meaning? The ledger is the registry on which notaries transcript transactions of buildings and any other kind of goods between two parties. Its role is to immutably certify the ownership: the notary must look for records in a ledger to certify the ownership of the good to sell, in order to approve the transaction.

Therefore, the idea of DLT is that the ledger is distributed among many parties, in order to ensure its reliability; not only, the consensus to transactions is cooperative, i.e., many parties have to approve the transactions, based on their copy of the ledger. This is the idea behind the acronym DLT, the absence of a third party responsible for storing the ledger and for giving consensus to transactions (replaced by a plethora of anonymous counterparties) is the key idea to achieve resilience to tampering, because the probability that many parties are corrupted is much lower than the probability that one single party (the unique responsible for consensus, in a centralized schema) is corrupted.

The technology to effectively deal with distributed ledgers is called *BlockChain*, because they are stored as *chains of blocks*: this solution, associated with hashing techniques, ensures the immutability of the ledgers.

The first platform that has introduced BlockChain technology is *Bitcoin*, the platform that supports the *bitcoin* virtual currency. The absence of a third central party that gives consensus to money exchange is (probably) one of the keys of the success of *bitcoin*. Its fame has become worldwide, and its popularity leads people usually think that financial services can take advantage of BlockChain technology only if they are based on the exchange of virtual currencies. However, this view is quite limited and does not consider the current scenario as far as available BlockChain platforms are concerned since its birth, BlockChain technology has significantly evolved and several platforms are now available, each of them providing a specific interpretation of the concept of distributed ledger and different approaches for its application.

During its evolution, BlockChain technology has been extended to support the concept of *Smart Contract*, i.e., a contract between two parties that is automatically executed by and within the platform, without need of human intervention. This idea has opened the way to apply BlockChain technology to a plethora of application contexts that were unexpected, at the beginning; however, it is necessary to properly comprehend how each platform supports this concept.

The goal of this paper is to provide readers that operate in the financial market with an introduction to (nowadays) most popular BlockChain platforms. To do so, we have to discuss the main features that characterize a BlockChain platform: in fact, depending on the features provided by each single platform (for example, the way it supports smart contracts, one of the buzzwords of DLT) several categories of services can be implemented.

The paper is organized as follows. First, the research methodology is presented: we start by reporting a brief history of BlockChain technology; then, we explain the difference between *permissionless* (classical) and *permissioned* platforms; we continue by explaining the concept of *smart contract* and the different approaches to support it. After the investigation methodology is introduced, we analyse the four most popular platforms, i.e., *Bitcoin*, *Ethereum*, *Corda* and *HyperLedger Fabric*. Next, a discussion section will address the typical issues concerned with the adoption of DLT for supporting financial services. Finally, we conclude the paper with some final remarks.

Methodology

Brief History of BlockChain

Haber and Stornetta (1990) developed a cryptographically protected chain of blocks in which no one could manipulate the timestamps of the documents. But it was only in 2008 that Satoshi Nakamoto described the first BlockChain system in Nakamoto (2008) named *Bitcoin*, that supports the virtual currency named *bitcoin*. In *Bitcoin*, the

BlockChain constitutes the underlying protocol of any crypto-currency and is a novel peer-to-peer methodology to link a sequence of transactions or events that ensures their immutability.

A few months later, a new open source application implementing the *Bitcoin* protocol was released and the first block of the chain, called *Genesis*, was generated. By installing this application, anyone can become a part of the *Bitcoin* peer-to-peer network.

Even if *bitcoin* is the most famous application of BlockChain technology, many different applications could significantly benefit by its adoption. To this end, in 2013 V. Buterin started working on a BlockChain platform capable of providing advanced functionalities, such as *smart contracts*, executing them directly within the peer-to-peer network (see Buterin, 2014).

Ethereum was presented in Wood (2014) as a new public BlockChain platform that overtakes the simple support to a crypto-currency (named *Ether*), by evolving into a platform to develop decentralized applications as well. This is made possible by natively supporting the concept of *smart contract*, (thus, actually it has implemented the ideas in Buterin (2014)).

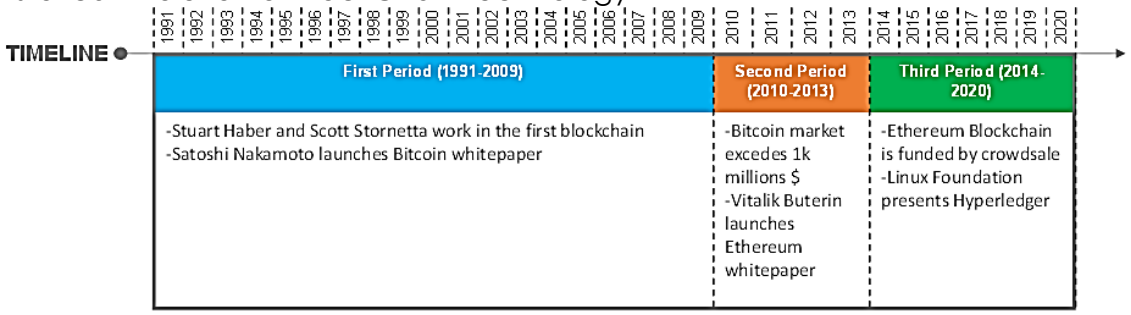
The concept of smart contract was originally introduced by Szabo (1997): it combines computer protocols with user interfaces to execute the terms of a contract. Furthermore, in 2014, when BlockChain was clearly emerging, Fairfield (2014) proposed the use of smart contracts to carry out with the transaction processes by automatically executing contracts in a cost-effective, transparent and secure manner.

However, this does not end the history of BlockChain: this is just the beginning of its evolution. In fact, the history continues with the *HyperLedger* project (see Dhillon et al. 2017), by Linux Foundation. It aims at developing a family of BlockChain platforms based on the same basic architecture, whose goal is to support information systems; the most famous platform belonging to this family is *HyperLedger Fabric* (see Sousa et al. 2018). Furthermore, within the financial world, the platform named *Corda* (presented in 2016 in Brown et al. 2016) is gaining a lot of interest, because it supports smart contracts in a specific way that tries to reconcile technical aspects and juridical aspects.

Figure 1 illustrates how, after a long latency period, the births of *Bitcoin* and *Ethereum* have been the disruptive events that have caused the subsequent birth of the *HyperlerLedger* project, which represents the current evolutionary trend as far as new developments are concerned.

Figure 1

Historical Evolution of BlockChain technology



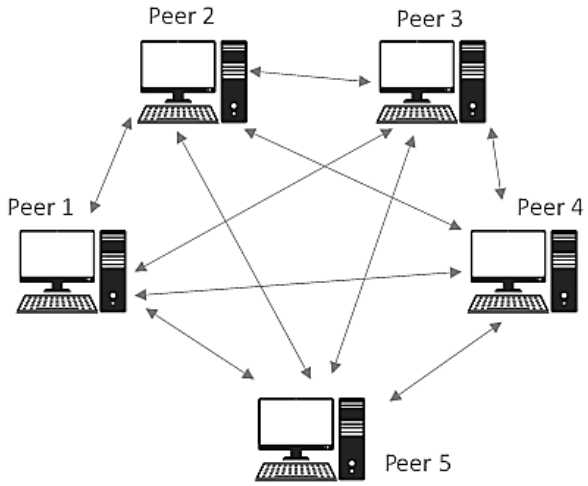
Source: Authors' work

General Concepts Concerning BlockChain Technology

Before carrying on our analysis, we present some basic concepts that underlay BlockChain platforms, so that novices can fully understand our argumentations.

A BlockChain system is a *peer-to-peer network*, where each peer is a computer connected to the other computers involved in the network (see Schollmeier (2001) and Pourebrahimi et al. (2005), for extensive presentations of concepts concerning peer-to-peer networks). Peers are also called *nodes* of the network. They are called *peers* because no node dominates other nodes, i.e., there is not a master that controls slaves; each of them plays an equal role. Figure 2 shows a sample topology of a peer-to-peer network, which clarifies why peers are also called nodes.

Figure 2
Sample of Peer-to-Peer Network



Source: Authors' work

On each node, an instance of the software that provides access to the peer-to-peer network is running.

The term *Distributed Ledger Technology* (DLT) means that a BlockChain platform stores multiple copies of the database (or ledger): the greater the number of copies of the database, the higher the capability of the network to resist to attacks.

Any peer in the network can receive transactions, i.e., requests to change the data. Transactions are performed only if there is consensus by the network, i.e., the overall network must agree.

The name BlockChain originates from the fact that the database is structured as a *chain of blocks*, where each block records a pool of transactions issued to the system and the current state of modified data. Blocks are never removed; in contrast, they are continuously added, so that each block points backward to the top-most block added before itself. The chain implements, in effect, the concept of *immutable ledger*, because the whole history of transactions is stored within the chain, i.e., within the database.

The mostly-used consensus mechanism is called *Proof ofWwork* (see Beccuti & Jaag, 2017 and Garcia-Bringas et al., 2019); we shortly explain it. All transactions issued in a given time period to a pool of peers are validated (i.e., it is verified that the spent amount of money is truly available) and collected into a block, which should be added to the chain: the consensus mechanism is aimed to validate this action. The mechanism is based on the fact that, based on the content of the block, it is possible to generate a hash code to identify the block; such a hash code must respect specific

constraints, that make it very hard to find it. Specific peers called *miners* have the goal of finding the cryptographic key able to obtain a hash code with the desired properties. If the hash code is found, this gives the consensus to add the block to the chain. Why does it work? Because it makes difficult, for a malicious peer, to force some wrong transactions, as well as it prevents the *double spending* attack, i.e., two transactions that are simultaneously issued to different peers, that spend the same amount of money. Since these two transactions are likely to be stored into two different blocks, generated more or less at the same time, they point to the same previous block. However, only one of them can be inserted into the chain; transactions in the other block must be validated again and this allows for discovering that the amount of money has been already spent. In this scheme, miners can be either all the peers in the network or a specialized subset of them; clearly, the larger the number of miners, the higher the speed of the network to validate a block.

Classical vs Permissioned BlockChain platforms

What is the level of trust that each participant to the BlockChain has in relation to other participants? It depends on the application context.

Usually, people think about virtual currencies, like *bitcoin*: in this context, it is necessary to avoid double spending of the same amount of money, in an environment where *nobody trusts anybody*. However, in different application environments, this assumption is not always true. To understand, we classify possible application environments.

No trust. When no trust is possible, i.e., *nobody trusts anybody*, the best warranty that transactions can be performed is given by the largest possible number of nodes. In fact, a large number of nodes (parties), involved both to validate transactions and to store blocks, makes very difficult to attack and corrupt the system.

Partial trust. In a controlled environment, where many parties co-operate to get a common goal, such as an integrated supply chain (see Korpela et al., 2017), in principle each party trusts other parties a little bit. However, they do not fully trust each other, for several reasons: a centralized approach, where a central entity provides the IT support for everybody, could be prone to system faults, programming errors and external attacks. In contrast, having several nodes that provide consensus to transactions as well as that store multiple copies of the ledger significantly increases the reliability of the system.

Full trust. This scenario is the classical approach to the development of information systems to provide a service to many parties. A central authority is (or must be) fully trusted by other parties (they subscribe a service contract, or they are forced by laws). In this context, it is not exactly true that parties really and fully trust the central authority: they have to trust it, even if they do not want.

Clearly, the third scenario (and doubts concerning trust about the central authority) motivated the original design of BlockChain, which is inspired by the first scenario. However, the second scenario, being in the middle, has originated a different approach to BlockChain technology, which led to the definition and the development of two distinct families of BlockChain platforms:

Permissionless BlockChain platforms. This family encompasses classical BlockChain technology, devoted to support virtual currencies. A new node is free to enter the network, provided that its behaviour is compliant with general rules of the platform. In this case, the larger the number of nodes, the higher the warranty that transactions are correct and immutable. This type of BlockChain platforms is good for *No trust scenarios*.

Permissioned BlockChain platforms. This family encompasses platforms such that new nodes cannot freely enter the network; they must be authorized by an administrator. Furthermore, nodes' owners must agree with the business logic supported by the system: this means that only well-defined actions can be performed by transactions, i.e., only those allowed by the contract every party undersigned before entering the network. This type of BlockChain platforms is good for *Partial Trust scenarios*.

Let us explain the rationale behind the two families. When *nobody trusts anybody*, the consensus to transactions is reached by means of the *Proof of Work* mechanism, which we shortly described above. However, this mechanism is very expensive, because a very large number of miners is necessary, each one performing long and energy consuming computations. In practice, *no trust* means *a large (and expensive) effort to achieve trust*.

On the other side, *partial trust* asks for a different approach: it is not necessary to waste as many computational resources as those necessary in permissionless platforms. A very different consensus mechanism can be adopted. An example is the *Proof of Knowledge* approach (see Mazumdar & Ruj, 2018). In this approach, consensus is managed by building a total order among transactions, based on dependencies among *read sets*, i.e., data affected by a transaction, and *write sets*, i.e., new data produced by transactions. If it is not possible to build a total order among transactions on every node involved in the BlockChain, this means that a conflict has been detected and conflicting transactions are aborted. This consensus mechanism works with a small number of nodes.

Smart contracts

Originally, BlockChain technology was thought to support money exchange based on a virtual currency. However, a ledger can be used for many application contexts, not necessarily for money exchange. Consequently, the idea of using BlockChain for application contexts without a simple exchange of possibly virtual money is straightforward.

How to foster the adoption of BlockChain technology? If transactions are not totally free money transfers, but ruled operations that can be performed on the basis of an agreed contract, a new and immense scenario opens. This is the concept of *smart contract*; originally, it was introduced in Szabo (1997), "to describe agreements between two or more parties, which can be automatically enforced without a trusted intermediary" (from Atzei et al., 2018). The idea was ignored for a few years; then, the advent of BlockChain technology has made it actually applicable.

A smart contract can be seen as a contract state, i.e., the set of properties that characterize it, provided with some transformation methods, i.e., procedures that determine how the contract state can change. A transaction consists in asking to change the contract state by invoking transformation methods. When a transaction is issued, the contract state is changed, according to the invoked transformation method.

This concept has incredible potentialities: once two parties have agreed to start the contract, its behaviour can become automatic, there is no need for a third party that handles the contract.

This behaviour can be summarized by the following sentence taken from Clack et al. (2016): "A smart contract is an automatable and enforceable agreement. Automatable by computer, although some parts may require human input and control. Enforceable either by legal enforcement of rights and obligations or via tamper-proof execution of computer code."

Anyway, smart contracts are supported by different BlockChain platforms in different ways.

In-platform code. This approach to smart contracts is characterized by the fact that transformation methods are shared within the BlockChain platform, ready to be used when necessary. A transaction is a triple *‹os, ns, r›*, where *os* is the old state, *ns* is the new state and *r* is the request that generated the new state. As far as the possibility of using transformation methods by parties is concerned, three different scenarios are available. (1) ***Contract-Specific Code***: the code of transformation methods is associated to one specific contract, without any form of sharing. (2) ***Contract-family code***: the code of transformation methods is shared among contracts belonging to the same family. (3) ***Global Code***: transformation methods are global, in the sense they can affect many objects stored within the ledger.

External Code. This category encompasses smart contracts whose business logic is not within the platform. This is the case of *Bitcoin*: the first BlockChain platform in the world has not been designed to host smart contracts; nevertheless, it is used for many smart contract-based applications (see Atzei et al., 2018). This is made possible by implementing protocols based on cryptographic-message exchange: transactions are registrations of messages; involved parties receive messages, in such a way only involved parties can decipher them, and, consequently, act (see the description of *Bitcoin* in further sections).

It is clear that, in this scenario, the business logic of smart contracts is handled outside the BlockChain platform: each party must implement it, hoping to be conformant with specifications.

Results: Analysing principal BlockChain platforms

Based on the investigation methodology previously introduced, we now introduce and analyse the most popular BlockChain platforms.

(1) Bitcoin has been the first BlockChain platform. Born to support the *bitcoin* virtual currency, in fact, it validated the approach, proving the effectiveness of the idea. It is a typical permissionless platform: a new node (party) can freely enter the peer-to-peer network; remember that this approach is typical in the context of virtual-money transfer, that we characterize as the typical context where *nobody trusts anybody*.

As far as the support to smart contracts is concerned, its design strongly limits the possibility to add smart contracts to the platform in a native way; in fact, remember that only in a subsequent time the concept of smart contract has been associated with BlockChain platforms (consequently, *Bitcoin* has not been designed to natively support smart contracts). For this reason, smart contracts in *Bitcoin* must be necessarily based on external code: parties involved in the contract exchange ciphered messages that can be read only by involved parties; code for automatic execution of contracts is outside the platform, but this approach can create significant problems as far as trust in executing smart contracts is concerned. To address this intrinsic problem, Atzei et al. (2018) proposes an algebra for specifying semantics of contracts, to be executed by different remote systems connected to the platform.

(2) Ethereum (see Wood, 2014) is the BlockChain of the *Ether* virtual currency. Since it is designed to support virtual-money exchange, it is still a permissionless platform, thus it is based on the *Proof of Work* consensus mechanism, as *Bitcoin* is.

However, unlike *Bitcoin*, it natively supports the execution of smart contracts: they are designed to deal with exchange of money, even though contracts that do not exchange money could be developed as well. A contract is identified by an address and has a state, whose changes are stored within the ledger.

The contract is created with a creational transaction, which also registers the transformation code. Then, a party acts on the contract by sending a transaction request to the address that identifies the contract. Then, the transformation code is executed and the new contract state is stored in the ledger (see Luu et al., 2016 for a complete description).

Ethereum contracts are written in the *Solidity* programming language (introduced in Dannen, 2017), which is similar to *JavaScript*. Later, we will see an example of contract code.

Contract code can be introduced anytime by anybody; each contract has its own code. Due to the *Proof of Work* consensus mechanism, the transformation code is executed on a large number of nodes, causing an excessive use of computational power; however, this is necessary, because we are still in the *no trust* context.

(3) HyperLedger Fabric (see Sousa et al., 2018) is a *permissioned* platform that gives a different perspective to the adoption of BlockChain technology: a BlockChain is a database that immutably logs all changes (transactions) performed on the database; this approach ensures that the current state of the database can be rebuilt, by re-executing change requests stored within the ledger.

The first effect of this database view is that not everybody can enter the network: only authorized parties are admitted (in fact, it is a *permissioned* platform).

The second consequence is that smart contracts are global procedures, called *chain code*, which actually perform changes on data; a transaction is the invocation of a procedure by a party. Chain code can be written in three different programming languages: *Java*, *JavaScript* and *Go* (introduced in Pike, 2009).

Furthermore, chain code cannot be added freely by parties: it is uploaded by administrators of the chain, because its role is similar to stored procedures in relational database technology. Thus, admitted parties are not free to do anything they want: they are allowed to execute only predetermined procedures.

In terms of computational resources, a *Proof of Knowledge* consensus mechanism is adopted, that is called *Byzantine Fault Tolerant* consensus mechanism, explained by Sousa et al. (2018). This mechanism is able to limit the number of nodes involved both in the execution of chain code and in the validation of transactions, thus minimizing the necessary computational power, if compared to permissionless platforms.

(4) Corda is "a distributed ledger platform for recording and processing financial agreements" (from Brown et al., 2016, first sentence of Section 4). It is a permissioned platform, thus only authorized parties can enter the network.

Corda is designed to support legal aspects related to smart contracts: contracts are accompanied by a *legal-prose description* of the contract itself; furthermore, when a transaction is performed, a legal-prose version is generated.

A smart contract, or *smart agreement*, has a state, that is accessible only by involved parties, as well as it has transformation code and validity rules. *Java* and *Kotlin* (presented by Panchal & Patel, 2017) are supported as programming languages.

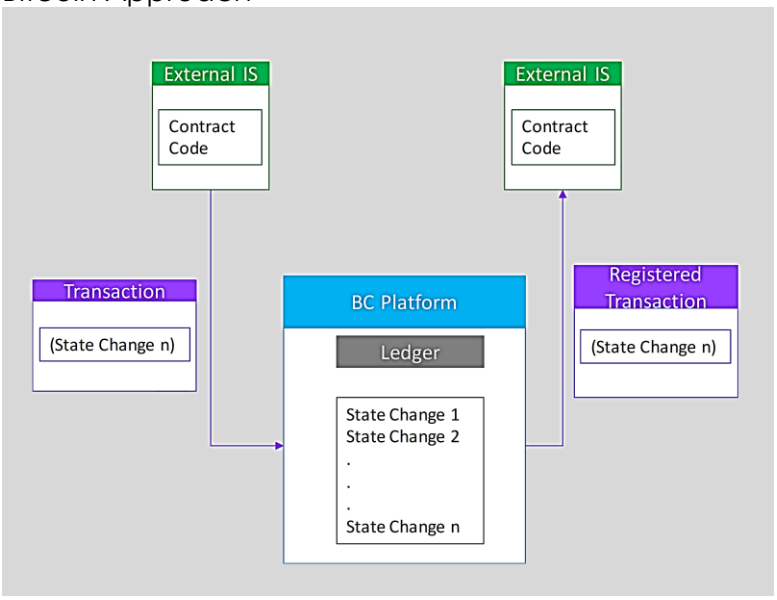
An interesting concept provided by *Corda* is the notion of *Contract Template* (see Clack et al., 2016): parties pre-load templates of contracts, where details (e.g., interest rate and duration) are not specified; when two or more parties agree, the actual contract is derived from the template, by specifying missing details; this way, all contracts derived from the same template share the same code.

Contract execution is performed only by parties involved in the contract. Then, a pool of *Observer* nodes guarantees the correct sequence of transactions (state changes) by validating timestamps. The effect is the limited amount of computational resources necessary to perform transactions and execute transformation code, because only few nodes are involved.

Anyway, the most distinctive feature provided by *Corda* is the legal-prose version of contracts: a correspondence mechanism ensures that legal prose has a code counterpart, in order to ensure legal validity to contracts.

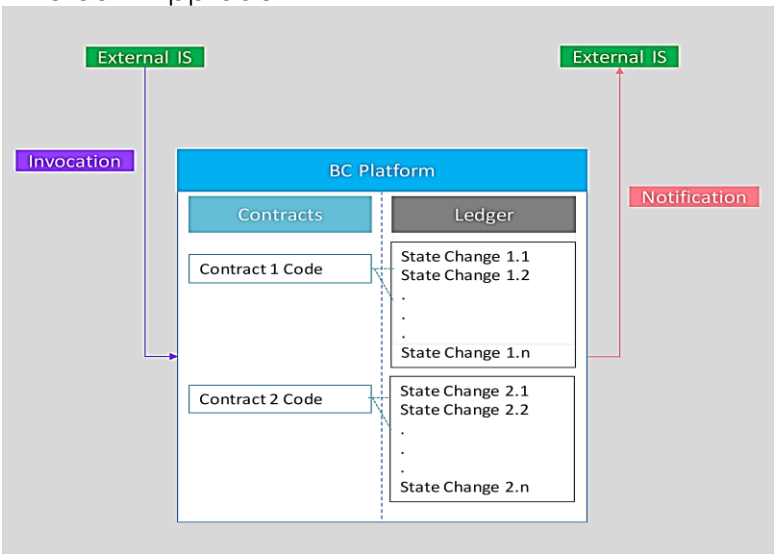
The interested reader can find a detailed comparison of *Ethereum*, *HyperLedger Fabric* and *Corda* in Valenta & Sandner (2017).

Figure 3
Bitcoin Approach



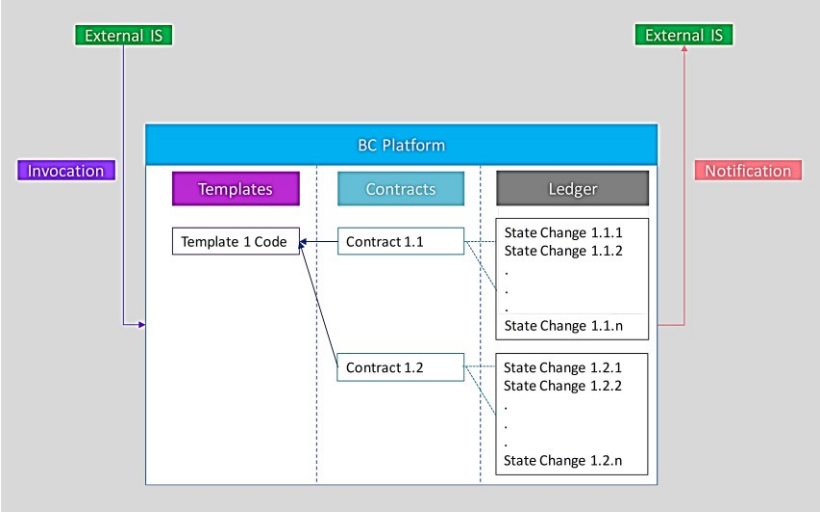
Source: Authors' work

Figure 4
Ethereum Approach



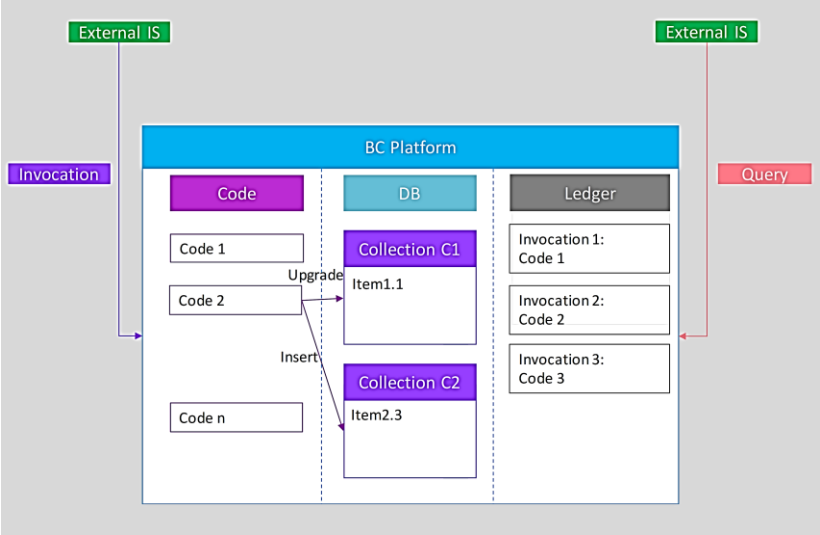
Source: Authors' work

Figure 5
Corda Approach



Source: Authors' work

Figure 6
HyperLedger Fabric Approach



Source: Authors' work

To further clarify, consider Figures 3-6, that illustrate the different approaches adopted by the four discussed BlockChain platforms, as far as smart contracts are concerned.

(1) Figure 3 shows the *Bitcoin* approach. The code is not inside the platform; in contrast, it resides on external information systems (denoted as *External IS*). When the code is activated, it sends a transaction to the platform: its content is the description of the state change of the contract. Involved external ISs are notified by the change. The ledger stores all state changes.

(2) Figure 4 shows the approach adopted by *Ethereum*. The code is stored within the platform. When an external IS invokes the code, it is executed, and makes a change to the contract state. Other involved external ISs are notified; the ledger stores all state changes. Note that there is no code sharing: contracts that behave in the same way have their own copy of the code.

(3) Figure 5 illustrates the approach followed by *Corda*. The approach is similar to *Ethereum*, i.e., the code is within the platform. However, it is associated to templates.

When a contract is instantiated, it refers to a template; this way, contracts referring to the same template share the same code. When an external IS invokes the contract, the state change caused by the contract code is registered into the ledger; other involved external ISs are notified.

(4) Figure 6 shows the approach adopted by *Hyper- Ledger Fabric*. This platform creates a shared and distributed database among external ISs. A database contains data items. The code is global for the database: a transaction is an invocation of a procedure. The invoked procedure changes the state of data items in the database; the ledger stores all code invocations, in this way, the current database state can be rebuilt from scratch, if necessary. External ISs can query the database, as it were a traditional database.

Table 1
Features of popular BlockChain platforms and financial service type

	Bitcoin	Ethereum	HyperLedger F.	Corda
Features				
Permissionless	X	X		
Permissioned			X	X
Contract-specific code		X		
Contract-family code				X
Global code			X	
External code	X			
Financial Service types				
Virtual currency	X	X		
Asset property			X	
Fraud detection			X	
Contract between Financial Op.				X

Source: Authors' work

Smart Contracts in Ethereum

To help the reader understand how a smart contract looks like, we provide a simple example for *Ethereum*, written in the *Solidity* programming language. We report the code hereafter and, then, we will explain it.

```
pragma solidity >=0.4.21;

contract Wallets {
   address public owner;
   mapping (address => unit) public balances;

   function Wallets() public
  {      owner = msg.sender;  }

   function load(unit amount) public
   {  if (msg.sender != owner) return;
      balances[msg.sender] += amount;  }

   function transfer(address receiver, unit amount) public
   {  if (balances[msg.sender] < amount) return;
      balances[msg.sender] -= amount;
      balances[receiver] += amount;  }
}
```

Function *load* is exploited by the owner of the contract to load money to the pool of wallets, namely to the owner's wallet. Notice that, first, it is necessary to check if the issuer is the owner: if so, the owner's wallet is loaded with the amount of money received as parameter *amount*.

The last function is named *transfer*. It is called by any user that possibly has a wallet managed by the contract to transfer money to another user's wallet. Let us explain the function. The function receives two parameters, i.e., the identifier of the receiver and the transferred amount. First, it is necessary to verify if the sender user has enough money to transfer: if not, the function terminates with no effect. This check encompasses also the case in which a user previously unknown to the contract tries to transfer money: the user simply does not exist in the map and the 0 value is obtained as current balance of the wallet.

The second instruction of the function subtracts the transferred amount from the wallet of the sender, while the last instruction adds the transferred amount to the wallet of the receiver.

The reader can see that it is not hard to comprehend the contract and it is not hard to write it, being familiar with object-oriented programming.

Discussion

In the classical centralized scheme, parties performing transactions (have to) trust the intermediary. However, this starting hypothesis of unwavering confidence in central entities, and in their information systems, it is not always clear: can we really trust central entities?

This is the key point that has made BlockChain technology disruptive: it eliminates intermediaries, ensuring the maintenance of trust and even increasing it (see Brezo & Bringas, 2012), by making possible to build networks with a decentralized validation mechanism in untrusted contexts.

Having clarified this premise, in this section, we want to discuss potential applications of BlockChain platforms for financial services.

We begin with a question: *what is the best platform for financial services*? The answer is: it depends on the type of service. Hereafter, we discuss some possibilities, summarized in section *Financial Service Types* of *Table 1*.

1) Virtual Currency. If the service to provide is based on a virtual currency, like *bitcoin* or *Ether*, the choice is mandatory: a service based on *bitcoin* must be necessarily deployed on the *Bitcoin* platform; a service based on *Ether* must be necessarily deployed on the *Ethereum* platform.

However, although it is possible to guess that financial operators are attracted by the possibility to operate with virtual currencies, we expect that this will be a small part of all financial services that in the future will be deployed on BlockChain platforms, simply because national states have their own non-virtual currencies.

2) Asset Property. Financial institutions exchange assets of various types, such as equities, bonds, and so on. An important issue to regulate the financial market is transparency as far as ownership of financial assets is concerned.

In such an application context, a platform like *Hyper-Ledger Fabric* could be the right solution: activities to support are established by regulatory bodies, in this case, asset exchange recording.

3) Fraud Detection. Customers of financial operators could try to fraud them, trying to exploit the fact that, in some cases, operators do not exchange information. An example is given by a service offered by banks to account owners: if the account owner presents an invoice to be paid later by a customer, the bank anticipates the money. However, a typical minor fraud is the following: the account owner presents

the same invoice to more than one bank, in order to unduly receive the money more than once. A platform like *HyperLedger Fabric* could be, again, the right solution: once a bank receives an invoice from an account owner, the invoice is registered, in order to tell other banks that the money has been already anticipated.

4) Contract between Financial Operators. When two or more financial operators sign a contract, this could be managed as a smart contract. The goal is to ensure transparency and to avoid misinterpretation because several information systems must deal with the contract. The adoption of a smart contract executed within a BlockChain platform removes duplications and possible inconsistencies. Since the nature of contracts might be very specific and related to a small group of financial operators (two or three, for example), a global approach is not possible. In this case, *Corda* could be the best solution, because parties can agree on a template (shared implementation of the contract) that is instantiated when they sign the detailed agreement. In this context, legal value of contracts is a crucial issue. The legal-prose version of smart contracts and of transactions is essential for agreeing and documenting all state changes of contracts.

The reader can notice that permissioned platforms offer, in our opinion, the best support to traditional financial services. They provide transparency and trust among financial operators. Of course, since they are permissioned, financial operators must be admitted to the network. This ensures a kind of fairness among parties: a party that tries to fraud other parties can be easily blocked and, in the worst case, kicked out from the network; the damage for the fraudulent financial operator would be enormous; thus, this scenario further increases the level of trust.

However, it is not easy to set up a permissioned scenario, because parties must agree in advance: requirements are crucial and must be understood and shared by all parties. Typically, the financial market is global: this means that an international consortium is the only way to build and regulate a BlockChain platform for sharing financial services.

An aspect to consider that is under investigation by researchers is scalability. It is directly related to the speed of transaction processing within a specific BlockChain platform, as well as to the total volume of such transactions per unit of time. Different flavours of BlockChain have already suffered important moments of crisis, with strong bottlenecks that came not only to extremely slow down the service, but even to shoot the costs for users, who were in the position of paying extra fees to raise the priority level of their transactions. A congested BlockChain platform, which cannot process transactions at the rate at which they occur, is no longer interesting for all BlockChain parties. If the network does not work as expected, many users of different types look for other alternatives.

To respond to this crucial challenge of scalability, some variants of BlockChain technology are exploring different technical alternatives, such as working with smaller size signatures, incorporating secondary chains for specific types of transaction, or are experimenting with different block sizes. In particular, the maximum size of the block is a technically long-disputed conditioner, which even today can significantly limit the transaction capacity of the network.

To understand the potential impact of this issue, we refer to various works that made a performance analysis of BlockChain platforms, such as Pongnumkul et al. (2017) and Dinh et al. (2018). It appears that the latency of a transaction can be hundreds of seconds. Such a latency is too high for information systems that have to process a very large amount of transactions.

As far as the four BlockChain platforms we consider in this paper are concerned, *HyperLedger Fabric* manages several chains at the same time, one for each

application context: this way, it is able to serve many distributed information systems, each one with its own database and its own chain; as an effect in terms of scalability, the number of nodes involved in the computation is kept low, in order to reduce the latency of transactions.

Corda is able to execute contracts only on nodes of involved parties; a reduced number of observer nodes, that guarantees the correct order of timestamps, is still beneficial in terms of latency of transactions.

In general, the adoption of the *Proof of Work* consensus strategy is a significant obstacle as far as scalability and reduced execution times are concerned. In fact, the computational effort made by miners is quite high, so it can significantly slow down transaction processing. This is a crucial issue in *Ethereum*, where contracts are executed by a large number of nodes in the network. This is the main reason why *Hyper-Ledger Fabric* and *Corda* do not rely on the *Proof of Work* consensus strategy. This different behaviour is studied by Pongnumkul et al. (2017), where it is shown that *HyperLedger Fabric*'s throughput is significantly higher than *Ethereum*'s trough put. The same is for latency: transactions in *HyperLedger Fabric* have a highly reduced latency, if compared to transactions in *Ethereum*; however, 34 secs in the worst case are still too many, in many applications, such as, registering transactions performed by credit cards.

Furthermore, by considering the general philosophy of the *HyperLedger* project, it is clear that BlockChain platforms are going to replace or backup local databases in information systems; furthermore, they could become sources for NoSQL frameworks able to integrate many data sources for data science applications, like Bordogna et al. (2017, 2018).

As a final remark, we think that this paper is innovative in the scientific literature concerning the adoption of BlockChain technology for financial applications. Indeed, other surveys presenting BlockChain platforms in the financial market are available in literature, such as Bouri et al. (2018) and Corbet et al. (2018). Although it is true that these works consider the financial market, they are focused on studying the dynamics of transactions performed by means of virtual currencies. In contrast, our perspective is quite different: this paper presents basic technological aspects, as well as it classifies application contexts; the goal is to provide readers with the basis to understand, for each single type of financial service to support that are not relying on virtual currencies, which is the best platform to potentially adopt.

Conclusion

This paper has presented a brief overview of most popular BlockChain platforms, by introducing their main features. We relied on an investigation methodology that highlights the different ways platforms support the concept of smart contract. Then, we focused on financial services, i.e., we investigated the best platform for supporting different types of financial service, by motivating our choices.

We can summarize the main outcomes of our analysis. (1) The most recent platforms, such as *HyperLedger Fabric* and *Corda*, definitely divide BlockChain technology from virtual currencies. (2) *Corda* addresses the problem of giving a legal description of smart contrasts, thus giving them the same legal validity as traditional contracts; this was a critical issue in the financial market. (3) *HyperLedger Fabric* opens the way to effectively integrate information systems of financial institutions, as an effective form of fraud prevention.

We can now make hypothesis about the future work on this topic. This survey has suggested us to investigate the problem of designing complex services and complex data models for permissioned platforms: we will investigate this in the near future.

Finally, to conclude our vision of the future, we could think about hybrid approaches, in which a BlockChain platform and a NoSQL data store system able to store large volumes of data in cloud environments (usually called Big Data) could cooperate to provide a service able to deal with large volumes of data ensuring, at the same time, their temporal integrity; as far as financial services are concerned, this approach could open the way to support a plethora of financial services by means of BlochChain technology, that now are considered not feasible.

References

1. Atzei, N., Bartoletti, M., Cimoli, T., Lande, S., Zunino, R. (2018), "SoK: Unraveling bitcoin smart contracts", In Bauer, L., Küsters, R. (Eds.), 7th International Conference Principles of Security and Trust, 14-20 April, ETAPS, Thessaloniki, pp. 217-242.
2. Beccuti, J., Jaag, C. (2017), "The bitcoin mining game: On the optimality of honesty in proof-of-work consensus mechanism", Swiss Economics Working Paper 0060, Swiss Economics SE AG, Zürich August 2017.
3. Bordogna, G., Capelli, S., Ciriello, D. E., Psaila, G. (2018), "A cross-analysis framework for multi-source volunteered, crowdsourced, and authoritative geographic information: The case study of volunteered personal traces analysis against transport network data", Geo-spatial Information Science, Vol. 21 No. 3, pp. 257-271.
4. Bordogna, G., Capelli, S., Psaila, G. (2017), "A big geo data query framework to correlate open data with social network geotagged posts", In Bregt A., Sarjakoski T., van Lammeren R., Rip F. (Eds.), Annual International Conference on Geographic Information Science, 9-12 May, Springer, Cham, pp.185-203.
5. Bouri, E., Gupta, R., Roubaud, D. (2019), "Herding behaviour in cryptocurrencies", Finance Research Letters, Vol. 29, pp. 216-221.
6. Brezo, F., Bringas, P. G. (2012), "Issues and risks associated with cryptocurrencies such as bitcoin", in Berntzen, L. (Ed.), 2nd International Conference on Social Eco-Informatics SOTICS2012, 21-26 October, International Academy, Research, and Industry Association Venice, pp. 20-26.
7. Brown, R. G., Carlyle, J., Grigg, I., Hearn, M. (2016), "Corda: An introduction", availbale at: https://www.researchgate.net/profile/Ian_Grigg/publication/308636477_Corda_An_Introduction/links/57e994ed08aed0a291304412.pdf (Feb 25, 2020)
8. Buterin, V. (2014), "A next-generation smart contract and decentralized application platform", available at: https://cryptorating.eu/whitepapers/Ethereum/Ethereum_white_paper.pdf (Feb 25, 2020)
9. Clack, C. D., Bakshi, V. A., Braine, L. (2016), "Smart contract templates: Foundations, design landscape and research directions", available at: https://arxiv.org/abs/1608.00771 (Feb 25, 2020)
10. Corbet, S., Meegan, A., Larkin, C., Lucey, B., Yarovaya, L. (2018), "Exploring the dynamic relationships between cryptocurrencies and other financial assets", Economics Letters, Vol. 165, pp. 28-34.
11. Dannen, C. (2017), Introducing Ethereum and Solidity, Apress, Berkeley.
12. Dhillon, V., Metcalf, D., Hooper, M. (2017), "The hyperledger project", In Blockchain Enabled Applications: Understand the Blockchain Ecosystem and How to Make it Work for You, Apress, Berkeley, pp. 139-149.
13. Dinh, T. T. A., Liu, R., Zhang, M., Chen, G., Ooi, B. C., Wang, J. (2018), "Untangling blockchain: A data processing view of blockchain systems", IEEE Transactions on Knowledge and Data Engineering, Vol. 30 , pp. 1366-1385.
14. Fairfield, J. A. (2014), "Smart contracts, bitcoin bots, and consumer protection", Washington and Lee Law Review Online, Vol. 71 No. 2, pp. 35-50.
15. Garcia-Bringas, P., Pastor, I., Psaila, G. (2019), "Can BlockChain technology provide information systems with trusted database? The case of HyperLedger Fabric", In

International Conference on Flexible Query Answering Systems, 17-19 June, Springer, Cham, pp. 265-277.
16. Haber, S., Stornetta, W. S. (1990), "How to time-stamp a digital document", In Menezes, A. J., Vanstone, S. A. (Eds.), Conference on the Theory and Application of Cryptography, 8-11 January, Springer, pp. 437-455.
17. Korpela, K., Hallikas, J., Dahlberg, T. (2017), "Digital supply chain transformation toward blockchain integration", 50th Hawaii International Conference on System Sciences, 4-7 January, University of Hawaii, Manoa, pp. 4182-4191.
18. Luu, L., Chu, D.-H., Olickel, H., Saxena, P., Hobor, A. (2016), "Making smart contracts smarter", 2016 ACM SIGSAC Conference on Computer and Communications Security, 24-28 October, Association for Computing Machinery, New York, pp. 254-269.
19. Mazumdar, S., Ruj, S. (2018), "Design of anonymous endorsement system in hyperledger fabric", available at: https://arxiv.org/pdf/1811.01410.pdf (Feb 25, 2020)
20. Nakamoto, S. (2008), "Bitcoin: A peer-to-peer electronic cash system", available at: https://bitcoin.org/bitcoin.pdf (Feb 25, 2020)
21. Panchal, R. K., Patel, M. A. K. (2017), "A comparative study: Java vs kotlin programming in android", International Journal of Innovative Trends in Engineering & Research, Vol. 2 No. 9, pp. 4-10.
22. Pike, R. (2009), "The go programming language", avaiilable at: https://talks.golang.org/2012/splash.article (Feb 25, 2020)
23. Pongnumkul, S., Siripanpornchana, C., Thajchayapong, S. (2017), "Performance analysis of private blockchain platforms in varying workloads", 26th International Conference on Computer Communication and Networks, 31 July-3 August, IEEE, Piscataway, pp. 1-6.
24. Pourebrahimi, B., Bertels, K., Vassiliadis, S. (2005), "A survey of peer-to-peer networks", 16th Annual Workshop on Circuits, Systems and Signal Processing, ProRisc (Vol. 2005), 17-18 November, STW, Utrecht, pp. 263-270.
25. Schollmeier, R. (2001), "A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications", in 1st International Conference on Peer-to-Peer Computing, 27-29 August IEEE, Washington, pp. 101-102.
26. Sousa, J., Bessani, A., Vukolic, M. (2018), "A byzantine fault-tolerant ordering service for the hyperledger fabric blockchain platform", In 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks DSN, 25-28 June, IEEE, Piscataway, pp. 51-58.
27. Szabo, N. (1997), "Formalizing and securing relationships on public networks", First Monday, Vol. 2 No 9.
28. Valenta, M., Sandner, P. (2017), Comparison of Ethereum, Hyperledger Fabric and Corda, Blockchain Center, Franfurt.
29. Wood, G. (2014), "Ethereum: A secure decentralised generalised transaction ledger", available at: https://files.gitter.im/ethereum/yellowpaper/VIyt/Paper.pdf (Feb 25, 2020)

About the authors

Pablo García Bringas is actually working as Vice-Dean of Deusto Engineering, at University of Deusto. He has been dedicated to Research, Technology and Innovation for 20 years, from positions as Head Researcher of DeustoTech Computing - S3Lab, and as General Director of DeustoTech - Deusto Institute of Technology. García-Bringas has combined teaching (in several BSc, MSc, PhD, and in-company courses) with research since 1998, mainly focused in Data Mining applied to the fields of (a) Information Security, (b) Industrial Processes, and (c) Genomics and Proteomics. He has over 15 years of experience in R&D management, with many projects, journal papers, and PhD supervised dissertations. The author can be contacted at **pablo.garcia.bringas@deusto.es.**

Iker Pastor-Lopez holds a PhD in Computer Science with the highest qualification and with Cum Laude Mention, since 2013. Master in Information Security, since 2010. Computer Engineer, since 2007. Program in Big Data and Business Intelligence, since 2016. He works in the Faculty of Engineering of the University of Deusto, and focuses him scientific interests in the areas of Big Data Analytics, Opinion Mining and Computer Vision. He is the author of several scientific articles in conferences and indexed journals. In addition, he is a member of the scientific committee of several congresses and reviewer of JCR journals. The author can be contacted at **iker.pastor@deusto.es.**

Giuseppe Psaila received his PhD at Politecnico di Torino (Italy). He is associate professor at University of Bergamo (Italy). He works on many topics concerning data management, such as data mining, XML processing, query languages and soft computing, information retrieval, Big Data and Open Data. The author can be contacted at **giuseppe.psaila@unibg.it.**