Corda: An Introduction

Richard Gendal Brown, James Carlyle, Ian Grigg, Mike Hearn

August 11, 2016

Abstract

A distributed database made up of mutually distrusting nodes would allow for a single global ledger that records the state of deals and obligations between institutions and people. This would eliminate much of the manual, time consuming effort currently required to keep disparate ledgers synchronised with each other. It would also allow for greater levels of code sharing than presently used in the financial industry, reducing the cost of banking for everyone. We present Corda, a platform which is designed to achieve these goals. This paper provides a high level introduction intended for the general reader. A forthcoming technical white paper elaborates on the design and fundamental architectural decisions.

Contents

1 Introduction

At R3, we believe that distributed ledger technology has the potential to transform the financial services industry to the benefit of its clients and participant firms alike. We envision a future where financial agreements are recorded and automatically managed without error, where anybody can transact seamlessly for any contractual purpose without friction. We believe markets will move towards models where parties to financial agreements record them once and collaborate to maintain accurate, shared records of these agreements. Duplications, reconciliations, failed matches and breaks will be things of the past. Isolated islands of asset representations will be no more.

We aspire to define a shared ledger fabric for financial services use-cases that can be deployed within existing legal frameworks and which relies on proven technologies. Our philosophy can be broken down into three categories: engineering for the requirements of institutions, a focus on non-functional requirements, and extensibility.

This paper introduces the design features of the Corda platform which we believe make it an attractive choice for regulated financial institutions.[1]

2 Context

Banks were amongst the earliest adopters of information technology and, contrary to popular belief, they have done a good job in automating previously manual processes and in digitizing previously physical processes. However, there are significant opportunities to improve the cost and efficiency of the architectures that emerged.

In particular, each bank maintains its own ledgers, which record that firm's view of its agreements and positions with respect to its customer set and its counterparts. Its counterparts, in turn, maintain their views. This duplication can lead to inconsistencies, and it drives a need for costly matching, reconciliation and fixing of errors by and among the various parties to a transaction. To the extent that differences remain between two firms' views of the same transaction, this is also a source of risk, some of it potentially systemic.

A plurality of financial institutions drives competition and choice but the plurality of technology platforms upon which they rely drives complexity and creates operational risk. However, until recently, this was unavoidable: except for centralised market infrastructures, there were few effective ways to consolidate technology across firms without also consolidating the firms themselves.[2]

Centralised market infrastructure utilities have gone some way towards increasing the amount of data and business-logic sharing between firms but the degree of integration achieved in the realm of financial transactions still lags far

[1]The authors can be reached via email: Richard Gendal Brown (richard@r3cev.com), James Carlyle (james@r3cev.com), Ian Grigg (iang@r3cev.com), Mike Hearn (mike@r3cev.com)

[2]Examples of this include the Depository Trust & Clearing Corporation (DTCC) and Continuous Linked Settlement Group (CLS).

behind that which has been achieved in the realm of information exchange since the advent of the web.[1]

We believe that the maturation of cryptographic techniques, exemplified in part by "blockchain technology", provides a new opportunity: the possibility of authoritative systems of record that are securely shared between firms. This vision provides the opportunity to transform the economics of financial firms, particularly but not exclusively in post-trade services, by implementing a new shared platform for the recording of financial events and processing of business logic: one where a single global logical ledger is authoritative for all agreements between firms recorded on it, even though the relationships and obligations recorded may remain between those firms. This architecture will define a new shared platform for the industry, upon which incumbents, new entrants and third parties can compete to deliver innovative new products and services.

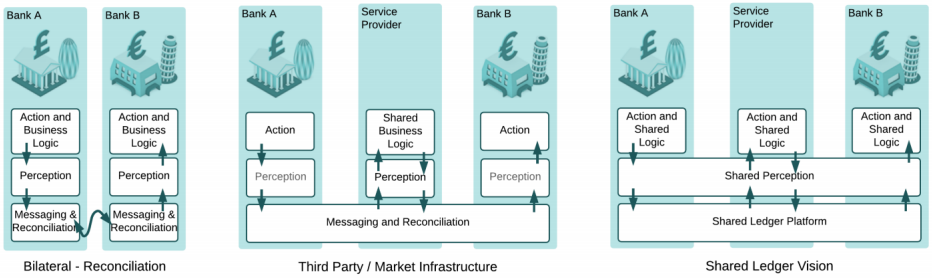

Figure 1: In the diagram above, we show a progression from a world where parties to shared facts record and manage their own records, with associated discrepancies and duplications ("Bilateral - Reconciliation") or one where parties delegate control and responsibility over critical processing to centralised utilities ("Third Party / Market Infrastructure"), to one where they collaborate to maintain a shared record, assured to be consistent between them, consuming the services of existing and new service providers and market infrastructure providers on an open and competitive basis ("Shared Ledger Vision").

We believe that the savings accruing from higher-quality data, fewer discrepancies and quicker agreement of details between firms will be significant. Moreover, deployment of this common architecture across firms will define a new platform on which existing and new providers can compete to serve the needs of clients. Going further, it is possible that such a platform will also find application within firms, where the problem of multiple systems recording details of the same trades is also a major driver of cost and complexity.

3 Vision

In the long-term, one can envision a "global logical ledger" with which all economic actors will interact and which will allow any parties to record and manage

agreements between themselves in a secure, consistent, reliable, private and authoritative manner. We say it is global in the sense that everybody sees the same data that pertains to them and logical in the sense that the physical implementation may be composed differently. As such, a possible end-state is one in which we have moved from authoritative systems-of-record maintained within firms to global authoritative systems-of-record shared *between* firms.

3.1 End-State Principles

Principles underpinning a possible end-state using distributed ledger technology may include:

- Facts recorded by the ledger are by contract accepted as admissible evidence and legally binding by all parties in any dispute.
- Settlement occurs on the ledger, not through the ledger.
- Once all parties to an agreement have assented, facts recorded on the ledger are final and immutable; errors, unwinds, etc., must be processed through a subsequent transaction. Firms will be under pressure to reengineer their internal processes to increase accuracy and quality.
- Any authorized actor may, in principle, connect directly to the ledger and use it to record agreements with its counterparts. No actor is compelled to deal with any other but we may see a decline in "tiered" or hierarchical market models.
- By promoting open standards and inclusive access, existing and new service providers can connect and compete to offer differentiated services, promoting choice and competition.
- The only parties who should have access to the details of a financial transaction are those parties themselves and others with a legitimate need to know.

However, the vision encompasses the notion of interim states, focusing first, perhaps, only on the sharing of business logic. This is intended to acknowledge the reality that todays systems will be with us for the foreseeable future, requiring co-existence, integration and migration paths as a fundamental part of the solution design. Less ambitious interim states can also deliver considerable value whilst legal and other non-technical implications of the longer-term vision are addressed in parallel.

It should be stressed that the long-term vision of a global logical ledger is intended to set a direction of travel but that its realization may be in the form of a multiplicity of ledgers. Perhaps this will take the form of one ledger per asset class which would be autonomous, loosely coupled, providing functional and operational independence between different business services.

Architectural and strategic choices underpinning the vision include:

- Records managed by this system will be accessible only to those actors with a legitimate interest in the assets and agreements they manage.
- The behavior of agreements managed by the system will be described in computer code that explicitly refers to and gains its legitimacy from overarching legal prose.[2]
- Support for contract code upgrades and explicit reference to dispute resolution procedures will be supported in order to provide certainty in the presence of failed contracts. This is because contractual disputes can occur, even in automated settings, as a result of both technical and human factors.
- Successful delivery of this vision will be through reduction of cost, risk and regulatory burden (including capital, liquidity and operational obligations) and through enabling of innovative new products and services.
- To gain wide adoption across the financial community, portions of the system must and will be open: open source, open development process, open standards.
- Although this vision talks in terms of a "platform" or "system", our belief is that the design will actually be multi-layered with different providers potentially competing/collaborating to deliver different pieces. Readers should not assume that we envisage a monolithic vertically integrated approach.
- The vision also includes the possibility that higher-level layers of the stack can contain IP proprietary to individual firms or groups.
- This system will operate under the assumption of an adversarial security environment: the growing threat of cybercrime must be taken as a given.

It is believed that the inventions needed to deliver this vision already exist. These include but are not limited to robust cryptography, global communications networks, standards for the definition of financial instruments and effective algorithms to ensure consistency at a global scale.

What makes this vision possible today is that the recent popular interest in distributed ledger and blockchain systems has created an environment in which such a vision can be openly discussed and that an alliance through which multiple banks can act together has been formed. It assumes an identity infrastructure between the participants in the network but makes no assumption as to its sophistication or mode of operation. Regulatory engagement is a key element of the design process.

From our requirements analysis and assessment of existing distributed ledger platforms, we concluded that no existing platform met our needs. In essence, the threat models underpinning the designs of traditional distributed databases were unsuitable for our use-case of bringing mutually distrusting legal entities into consensus; and the architectures of existing blockchain systems were unsuitable

for our requirement of restricted and carefully specified data sharing at the level of individual legal agreements. As a result we designed and began development of Corda.

4 Corda

Corda is a distributed ledger platform for recording and processing financial agreements, designed to implement the vision contained in this document.

The Corda platform supports smart contracts, matching the definition of Clack, Bakshi, Braine.[3] Our smart contract is an agreement whose execution is both *automatable* by computer code working with human input and control, and whose rights and obligations, as expressed in legal prose, are legally *enforceable*. The smart contract links business logic and business data to associated legal prose in order to ensure that the financial agreements on the platform are rooted firmly in law and can be enforced and that we have a clear path to follow in the event of ambiguity, uncertainty or dispute.

4.1 Principal Features

Corda is specialized for use with regulated financial institutions. It is heavily inspired by blockchain systems, but without the design choices that make traditional blockchains inappropriate for many banking scenarios.

Corda provides a framework to run smart contracts with these key activities and features:

- Recording and managing the evolution of financial agreements and other shared data between two or more identifiable parties in a way that is grounded in existing legal constructs and compatible with existing regulation
- Choreographing workflow between firms without a central controller.
- Supporting consensus between firms at the level of individual deals, not a global system.
- Supporting the inclusion of regulatory and supervisory observer nodes.
- Validating transactions solely between parties to the transaction.
- Supporting a variety of consensus mechanisms.
- Recording explicit links between human-language legal prose documents and smart contract code.
- Using industry-standard tools.
- Restricting access to the data within an agreement to only those explicitly entitled or logically privileged to it.

These features contribute to the design of a platform appropriate for use in complex, financial services organizations. Note that this design does not use a native cryptocurrency or impose a global transaction speed limit.

4.2 Concepts

We begin with the idea of a global ledger: a reliable single source. However, in our model, it is not the case that transactions and ledger entries are globally visible. In cases where transactions only involve a small subgroup of parties we strive to keep the relevant data purely within that subgroup.

The foundational object in our concept is a *state object*, which is a digital document which records the existence, content and current state of an agreement between two or more parties. It is intended to be shared only with those who have a legitimate reason to see it. To ensure consistency in a global, shared system where not all data is visible to all participants, we rely heavily on secure cryptographic hashes to identify parties and data. The ledger is defined as a set of immutable state objects.

We talk and think in terms of the state of agreements and our objective is to ensure that all parties to the agreement remain in consensus as to this state as it evolves. One could argue that this is the essence of the blockchain concept: ensuring that the data held by different actors is, and remains, consistent as operations are applied to update that data, and that this forms the foundation on which reliable transactions are built: from simple monetary payments to sophisticated smart contract transitions.

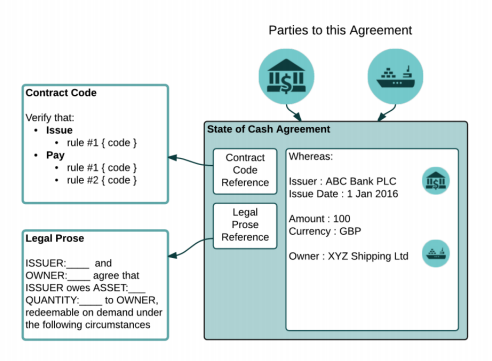

Figure 2: In the diagram above, we see a State object representing a cash claim of £100 against a commercial bank, owned by a fictional shipping company. The state object explicitly refers by hash to its governing legal prose and to the contract code that governs its transitions.

Our focus on "states" of agreements is in contrast to systems where the data over which participants much reach consensus is the state of an entire ledger

or the state of an entire virtual machine. Corda provides three main tools to achieve global distributed consensus:

- Smart contract logic to ensure state transitions are valid according to the pre-agreed rules.
- Uniqueness and timestamping services to order transactions temporally and eliminate conflicts.
- An orchestration framework which simplifies the process of writing complex multi-step protocols between multiple different parties.

4.3 Consensus

In Corda, updates are applied using *transactions*, which consume existing state objects and produce new state objects. There are two aspects of consensus:

1. Transaction validity: parties can reach certainty that a proposed update transaction defining output states is valid by checking that the associated contract code runs successfully and has all the required signatures; and that any transactions to which this transaction refers are also valid.
2. Transaction uniqueness: parties can reach certainty that the transaction in question is the unique consumer of all its input states. That is: there exists no other transaction, over which we have previously reached consensus (validity and uniqueness), that consumes any of the same states.

Parties can agree on transaction validity by independently running the same contract code and validation logic. However, consensus over uniqueness requires a predetermined observer, which in many cases may be required to be independent.

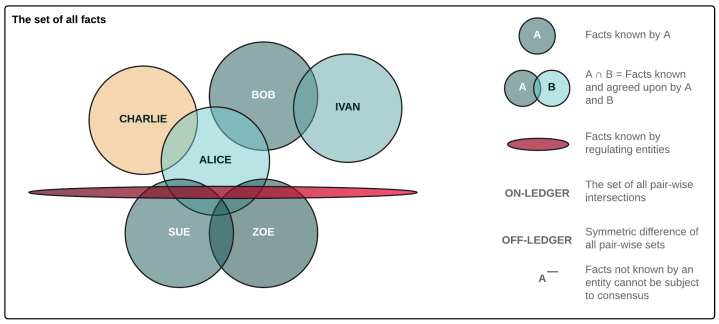

Figure 3: Consensus over transaction validity is performed only by parties to the transaction in question. Therefore, data is only shared with those parties which are required to see it. Other platforms generally reach consensus at the ledger level. Thus, any given actor in a Corda system sees only a subset of the overall data managed by the system as a whole. We say a piece of data is "on-ledger" if at least two actors on the system are in consensus as to its existence and details and we allow arbitrary combinations of actors to participate in the consensus process for any given piece of data. Data held by only one actor is "off-ledger".

Corda has "pluggable" uniqueness services. This is to improve privacy, scalability, legal-system compatibility[4] and algorithmic agility. A single service may be composed of many mutually untrusting nodes coordinating via a byzantine fault tolerant algorithm, or could be very simple, like a single machine. In some cases, like when evolving a state requires the signatures of all relevant parties, there may be no need for a uniqueness service at all.

It is important to note that these uniqueness services are required only to attest as to whether the states consumed by a given transaction have previously been consumed; they are not required to attest as to the validity of the transaction itself, which is a matter for the parties to the transaction. This means that the uniqueness services are not required to (and, in the general case, will not) see the full contents of any transactions, significantly improving privacy and scalability of the system compared with alternative blockchain designs. This design decision represents an important decision as to the acceptable tradeoffs in shared ledger architectures and is explored more fully in the forthcoming technical whitepaper.

4.4 Business Logic

Corda enforces business logic through smart contract code, which is constructed as a pure function that either accepts or rejects a transaction. The functions interpret transactions as taking states as inputs and producing output states through the application of (smart contract) commands, and accept the transac-

tion if the proposed actions are valid. Contracts define part of the business logic of the ledger, and they are mobile: nodes will download and run contracts inside a sandbox without any review in some deployments, although we envisage the use of signed code for Corda deployments in the regulated sphere.

The virtual machine we have selected for contract execution and validation is the Java Virtual Machine[5], as it has a wealth of existing libraries and a large skill base, and reusing an industry standard makes it easier for banks to reuse their existing code inside contracts. However, we augment it with a custom sandbox that is radically more restrictive than the ordinary JVM sandbox, and it enforces not only security requirements but also deterministic execution. Like Ethereum, the choice of standardising a bytecode set rather than a language enables users to innovate in contract language design, or reuse well known languages, according to taste. It also makes it easy to directly use contract code from internal applications, once that contract has been reviewed, which should simplify app development considerably.

4.5 Core Financial Concepts

Corda's architecture was heavily influenced by three *architecturally-significant use-cases*, deemed to be representative of common problems to which it is likely to be targeted. These use-cases are: cash, a security instrument and a derivative contract. In all three cases, we conceive of them as examples of financial agreements:

- A cash balance (e.g., "MegaBank and I agree that MegaBank owes me $1 million").
- A security under custody (e.g., "CustodyBank and I agree that I own 1000 MegaCorp shares").
- A bilateral derivative agreement (e.g., "Banks A and B agree that they are parties to the following Interest Rate Swap (IRS), which means they agree to exchange the following cashflows (netted) at predetermined scheduled times with an agreed payoff formula").

Taking one of these examples, Corda's cash model is based on an analysis of the nature of money. We explicitly model the business reality that there is no such thing as "money in a bank", only a cash claim that an owner has with respect to a named institution.[6] So our core Cash contract is extremely simple, but powerful: we record the legal identity of the cash issuer, the currency, amount, owner (and other information as to the nature of the claim, with an explicit link to the *legal prose* governing the agreement, which is also expected to specify resolution procedures in the event of dispute) and use that to build up all other cash-related concepts (payments, netting and so forth).

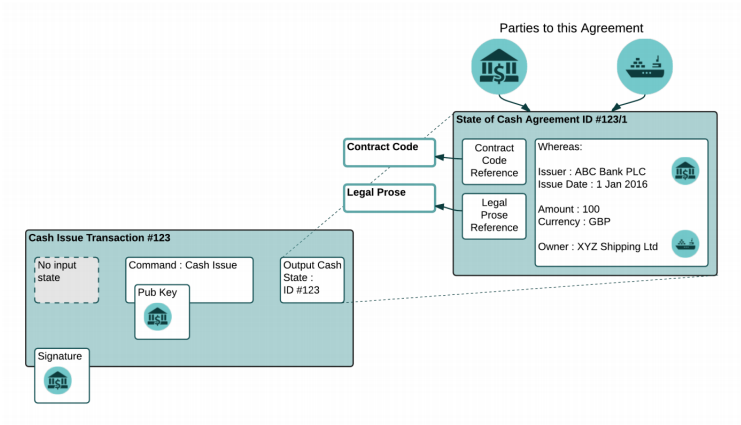

Figure 4: In the diagram above, we see one of the simplest Corda transactions: an issuance transaction. We see the creation of a new Cash state, issued by a commercial bank to a fictional shipping company. The issuing transaction is signed by the issuing bank. From this simple model, significantly more complicated transactions, such as payments, delivery-versus-payment contracts and future-dated obligations can be constructed.

4.6 Summary of the Corda Model

The core concepts in our model are:

- *State objects*, representing an agreement between two or more parties, governed by machine-readable *Contract Code*. This code references, and is intended to implement, portions of human-readable *Legal Prose*.
- *Transactions*, which transition state objects through a lifecycle
- *Transaction Protocols* or *Business Flow*, enabling parties to coordinate actions without a central controller.

Determinism is maximised and the amount of shared state required minimized by selectively and decisively restricting the universe of allowable programming techniques.

The combination of state objects (data), Contract Code (allowable operations), Transaction Protocols (business logic choreography), any necessary APIs, wallet plugins, and UI components can be thought of a Shared Ledger application, or Corda Distributed Application (*"CorDapp"*). This is the core set of components a contract developer on the platform should expect to build.

5 Comparisons with Other Distributed Ledger Platforms

Corda was created from extensive work with financial practitioners and is designed with their requirements in mind. However, its design is also inspired by previous work, including that introduced in the writings of Todd Boyle and Ian Grigg on triple entry accounting[7], and aspects of existing distributed ledger platforms such as Bitcoin[8] and Ethereum[9]. So it may be easier for people unfamiliar with Corda to understand it in terms of these platforms.

5.1 Comparisons to Bitcoin

Corda has some significant similarities to Bitcoin:

- Immutable states that are consumed and created by transactions is the same.
- Transactions have multiple inputs and outputs. Bitcoin sometimes refers to the ledger as the unspent transaction output set (UTXO set) as a result.
- A contract is pure function; contracts do not have storage or the ability to interact with anything. Given the same transaction, a contract's "verify" function always yields exactly the same result.

However, a Bitcoin transaction has a single, rigid data format. A "state" in Bitcoin is always a (quantity of bitcoin, script) pair and cannot hold any other data. Some people have been known to try and hack around this limitation by embedding data in semi-standardized places in the contract code so the data can be extracted through pattern matching but this is a poor approach. By contrast, our states can include arbitrary typed data. Our transactions invoke not only input contracts but also the contracts of the outputs. A Bitcoin transaction's acceptance is controlled only by the contract code in the consumed input states. We use the term "contract" to refer to a bundle of business logic that may handle various different tasks, beyond transaction verification. For instance, currently our contracts also include code for creating valid transactions (this is often called "wallet code" in Bitcoin).

A Bitcoin script can only be given a fixed set of byte arrays as the input. This means there's no way for a contract to examine the structure of the entire transaction, which severely limits what contracts can do. Our contracts are Turing-complete and can be written in any ordinary programming language that targets the JVM. Corda allows arbitrarily-precise time-bounds to be specified in transactions (which must be attested to by a trusted timestamper) rather than relying on the time at which a block happens to be mined. This is important given that many contract types we envisage supporting require precision in timing and because our primary consensus implementations use block-free conflict resolution algorithms.

5.2 Comparisons to Ethereum

Like Ethereum, code runs inside a relatively powerful virtual machine and can contain complex logic. Non-assembly based programming languages can be used for contract programming. They are both intended for the modelling of many different kinds of financial contracts.

However, the term "contract" in Ethereum refers to an instantiation of a program that is replicated and maintained by every participating node. This instantiation is very much like an object in an Object Orientated program: it can receive and send messages, update local storage and so on. In contrast, our implementation of the smart contract in code refers to a set of functions, only one of which is a part of keeping the system synchronised (the *verify* function). That function is pure and stateless (i.e., it may not interact with any other part of the system whilst executing). As contracts do not have any kind of mutable storage, there is no notion of a "message". Ethereum also claims to be a platform not only for financial logic, but literally any kind of application at all. Our platform considers non-financial applications to be out of scope, at least initially.

6 Roadmap

To arrive at this design, we first simulated and prototyped aspects of Corda in code to validate aspects of the concept. This is a list of some, but not all, extensions to the Corda model which are expected to be delivered in the near- to mid-term.

- Transaction Decomposition and Uniqueness Enhancements: Incorporating mechanisms to selectively obscure portions of transactions, including obfuscation from uniqueness services.
- Contract Verification Sandbox: Explicit link-time whitelisting of an aggressively minimal set of Java libraries.
- A plug-in based wallet for position inference.
- Calculation oracles or gateways to proprietary (or other) business logic executors (e.g., Central Counterparties or valuation agents) that can be verified on-ledger by participants.
- Using the Corda model to manage identity of users.
- Interoperability and data integration, specifically with respect to FpML, ISO20022, support for other data formats and integration/interoperability with other platforms.
- Building applications for reference data.
- Privacy enhancements using technology such as address randomization, zero-knowledge proofs, asset reissuance schemes.

- Reference contracts for further financial instruments.
- Native support of portfolio-level business logic, such as aggregations of state objects.

7 Conclusion

In contrast to most existing distributed ledger platforms today, Corda was built with the explicit purpose of recording and enforcing business agreements among registered financial institutions. As such, it takes a unique approach to data distribution and transaction semantics while maintaining the features of distributed ledgers which first attracted institutions to projects such as R3, such as automated, reliable execution of financial agreements in a replicated fashion.

Bibliography

[1] Mai. *IT in banks: What does it cost?* `https://www.dbresearch.com/PROD/DBR_INTERNET_ENPROD/PROD0000000000299039.pdf`, 2012.

[2] Grigg. *The Ricardian Contract.* `http://iang.org/papers/ricardian_contract.html`, 2004.

[3] Clack, Bakshi, Braine. *Smart Contract Templates: foundations, design landscape and research directions.* `http://arxiv.org/abs/1608.00771`, 2016.

[4] European Commission. *The European Commission's Settlement Finality Directive.* `http://ec.europa.eu/finance/financial-markets/settlement/index_en.htm`, 1998.

[5] Lindholm, Yellin, Bracha, & Buckley. *The Java Virtual Machine Specification Java SE 8 Edition.* `https://docs.oracle.com/javase/specs/jvms/se8/jvms8.pdf`, 2015.

[6] McLeay, Radia, & Thomas. *Money creation in the modern economy.* `http://www.bankofengland.co.uk/publications/Documents/quarterlybulletin/2014/qb14q1prereleasemoneycreation.pdf`, 2014.

[7] Grigg. *Triple Entry Accounting.* `http://iang.org/papers/triple_entry.html`, 2005.

[8] Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System.* `https://bitcoin.org/bitcoin.pdf`, 2008.

[9] Buterin et al. *A Next-Generation Smart Contract and Decentralized Application Platform.* `https://github.com/ethereum/wiki/wiki/%5BEnglish%5D-White-Paper`, 2014.