design your future

Blockchain and Payments

Jonathan Mayeur

Industriële wetenschappen en technologie

Bachelorproef Elektronica-ICT

academiejaar 2018-2019

vives katholieke hogeschool
associatie KU Leuven

Acknowledgments

Many people have contributed in their different ways to this paper. First, I would like to thank my parents, for supporting me in this.

Many thanks to IntellectEU, for giving me the chance to learn and gain experience during my internship and while writing this paper.

I would like to thank Thomas Bohner, who was my mentor during this internship and thought me a lot about blockchain and the financial services industry. Thank you for your guidance and comments along the way.

Also, many thanks to thank Belcham, for their support in arranging my visa and assisting me during my stay abroad. Many thanks to Hanna Zubko, for her guidance during my stay in New York.

Lastly, I wish to thank Valeriy Filippov and Igor Ivaniuk, for taking their time to share their technical knowledge with me. Also, thank you to Ruben Buysschaert and Hans Naert, for having thought me a lot of technical skills the last two and a half years.

Summary

Regardless of industry or geography, we see the same difficulty in today's business environment: institutions invariably maintain their own ledgers, which record each firm's view of an agreement and positions in respect to its customers and trading partners. This way of duplication can lead to inconsistencies, and a need for costly matching and fixing of errors for the various parties in the transaction.

With the introduction of blockchain technology, we can have a secure and incorruptible way to share information and automate processes between multiple entities in full trust.

Blockchain technology allows us to have a secure way to trade an asset in the network, together with the assurance that each required party sees the same view of that trade.

While there are blockchains that offer cryptocurrency as payment method, we currently see that many corporates are still hesitant of using cryptocurrencies for settlements because of their volatility in price and regulation.

This project will be the first blockchain application that has a delivery vs payment, with payment going through the traditional correspondent banking route using SWIFT gpi and a change of ownership of the asset on a blockchain.

IntellectEU is the delivering party that built the blockchain application. I studied how the used blockchain platform and payment rail works, got myself Corda-certified and used that knowledge to co-develop the blockchain application.

As a result, the project is on track to be deployed to different Tier 1 banks participating in the demo. This project will be showcased at Sibos in September 2019, an annual banking and financial conference organized by SWIFT.

Table of Contents

List of Abbreviations

AMQP	Advanced Message Queuing Protocol
API	Application Programming Interface
BIC	Business Identifier Code
CorDapp	Application that runs on the Corda platform
DLT	Distributed Ledger Technology
DTO	Data Transfer Object
GPI	Global Payments Innovation
IBAN	International Bank Account Number
JVM	Java Virtual Machine
LEI	Legal Entity Identifier
PoC	Proof-of-concept
RPC	Remote procedure call
SWIFT	Society for Worldwide Interbank Financial Telecommunication
TLS	Transport Layer Security
UETR	Unique End-to-End Transaction Reference

Introduction

In January 2019 SWIFT announced that it will launch a proof of concept to showcase how SWIFT Global Payments Innovation (gpi) can work together with R3 Corda's blockchain platform. The proof of concept (PoC) addresses the need for Distributed ledger technology (DLT) based commerce to be supported by global, fast, secure and transparent settlement using fiat currencies. A corporate using the Corda platform will be able to authorize payments through gpi Link, settling the gpi payments with the users' banks and having the credit confirmation reported back to the trade platforms via the gpi Link when they are completed.

IntellectEU is the party that build the blockchain application. There we choose to develop a "factoring" application. Invoice factoring is the practice where a company ("Corporate A") sells invoices at a discount to a factoring company, financial institution, or other corporate ("Corporate B"). The invoice is sold at a discount and the buyer gets paid when it collects from the customers (typically 30-90 days).

On the Corda network corporates will be able to buy and sell invoices. The invoice exists as an asset on the Corda blockchain and gets transferred between corporate A and corporate B once the transaction is settled. This means a "Credit Confirmation" coming in from gpi Link.

The PoC exists of a front-end, a backend to serve the front-end and the CorDapp. The CorDapp is referred to as marketplace-corda and is the part that will be explained in this paper.

1 Chapter I - Blockchain

As early as 1981, inventors were attempting to solve the internet's problems of privacy, security, and inclusion with cryptography. No matter how the process got reengineered, there were always leaks because third parties were involved.

In 2008 a paper got released called Bitcoin: A Peer-to-Peer Electronic Cash System. This introduced the term chain of blocks. This word has evolved over the years into the word blockchain.

The Bitcoin paper introduced a new protocol, that established a set of rules in the form of distributed computations, that ensured the integrity of data exchanged among these billions of devices without going through a trusted third party. This protocol is the foundation of a global number of distributed ledgers called blockchains.

Blockchain technology allows for a trusted environment to share information between multiple parties. It enables the shift from centralized repositories of information to more decentralized, robust and fault tolerant networks.

To understand blockchain we must look at distributed systems. As blockchain at its core is basically a decentralized distributed system.

1.1 Distributed Systems

Distributed systems are a computing model where multiple nodes work with each other in a coordinated fashion in order to achieve a common outcome. This is modeled in such a way that end users see it as a single logical platform.

The main challenge in distributed system design is the coordination between different participants in the system and fault tolerance. A theorem known as CAP theorem has proven that a distributed system cannot have Consistency (C), Availability (A) and Partition tolerance (P) at the same time.

Consistency is the property that ensures that all participants have a single latest copy of data. Partition tolerance means that the system will continue working even if there is a faulty participant in the system. This is also called Fault tolerance.

While the CAP theorem proves that the above three properties cannot be achieved at the same time. Blockchain manages to do so by using replication in order to achieve fault tolerance. To ensure that all participants have the same set of data, consensus algorithms are used. This provides consistency in the system. Consistency (C) on the blockchain is not achieved simultaneously with Partition tolerance (P) and Availability (A), but it is achieved over time. Blockchain is basically a method to achieve state machine replication across different participants.

1.2 Types of Blockchain

Blockchain is a secure and incorruptible way to share information and automate processes between multiple entities without the need to trust each other. Blockchains can be set up to have different levels of accessibility. This can bring different benefits or drawbacks.

Public blockchains allow anyone to join as a trust-less participant. This means anyone can read and write data to the blockchain. Anyone can download the protocol and verify transactions, by participating in the consensus process. Public blockchains are secured by cryptoeconomics. This is the combination of economic incentives and cryptographic verification using consensus mechanisms such as proof of work or proof of stake. These blockchains are generally considered to be fully decentralized. Examples of public blockchains are Bitcoin or Ethereum.

Consortium blockchains, also called shared permissioned blockchains, allows only verified participants to participate in the network. The consensus protocol is controlled by a selected set of nodes. This enables the use of different consensus mechanisms. Therefore, we can achieve higher transaction throughput compared to public blockchains. These blockchains may be considered partially decentralized. Known examples are Hyperledger Fabric, Corda and Quorum.

1.3 Distributed Ledger Technology

Blockchains are one form of distributed ledger technology. Distributed ledger technologies drastically reduce the cost of trust. It allows to keep organization specific databases in sync with each other and bringing potentially distrusting parties into consensus about shared facts.

2 Chapter II - Corda

Corda is a distributed ledger software for recording and processing shared data. It was developed by R3 in collaboration with over 200 technology and industry partners. And backed by investment of over 120 million dollars from more than 45 firms. Corda was introduced in a white paper in April 2016 and was released as an open source project in October of 2016. While the development of Corda was driven by the needs of regulated financial institutions, field experience has demonstrated that Corda is far more broadly applicable.

Regardless of industry or geography, we find the same difficulty in today's business environment: institutions invariably maintain their own ledgers, which record each firm's view of an agreement and positions in respect to its customers and trading partners.

Effort is needed to keep organization specific databases in sync with each other. Corda ensures that the data held by different actors is, and remains, consistent as operations are applied to update the data.

2.1 Platform Architecture

Corda's architecture is designed to manage real transactions between identifiable parties, with privacy and legal certainty. And do so across an open network on which multiple applications can execute and seamlessly interoperate.

2.1.1 The Corda Ledger

The ledger is a subjective construct from each peer's point of view. In Corda there is no central ledger. Each network participant maintains a separate vault of facts. When participants share a fact, they both store identical copies.

Communication occurs on a point-to point basis only. There is no global broadcast or gossip network. Peers communicate using AMQP/1.0 over TLS.

2.1.2 States

States are immutable objects that represent facts such as an agreement or contract at a specific point in time. A state object is intended to be shared only with those who have legitimate reason to see it.

For a shared fact to evolve, a new state is created by copying the current state and making updates as required. The previous state is then marked as historic. Historic states are not considered 'on-ledger', but they remain accessible.

The lifecycle of a shared fact or agreement over time is represented by a state sequence. The sequence head reflects the current state of the shared fact. The head is always current, all prior states are historic.

The ledger from each peer's point of view consists of all the state sequence heads tracked in the vault.

2.1.3 Transactions

Transactions are the mechanism in which states evolve over time. Transactions are basically proposals to update the ledger and are only visible to the peers where the state is relevant to. Transactions are committed and update the ledger only when signed by all required peers. This process is atomic: either all the transaction's proposed changes are accepted, or none are.

Digital signature enforces output state immutability. Attempts to mutate the output states of committed transactions are easily detected by verifying the digital signatures appended to the transaction.

New transactions reference prior transactions by hash, building up an immutable chain-like structure. Any on-ledger state can be referenced by a transaction id & index coordinate. This reference is used to resolve prior states to new states.

There are three types which transactions can facilitate:

- Issuance: Issue states on the ledger.
- Updates: Change properties in the state. By consuming the prior state and creating a new state.
- Exits: Consume state and don't create a new state.

2.1.4 Contracts

The whole point of Corda is to facilitate bringing potentially distrusting parties into consensus about shared facts. State objects evolve in a way that is validated by contract code. Each state in Corda must be paired with a contract. Contract code verifies that the proposed transaction meets the rules.

The verification function is defined in the contract code. The function takes a transaction as a parameter and either throws an exception if the transaction fails verification or returns nothing if the transaction verifies.

To verify a transaction, the contract code must be executed by all required peers on a need-to know basis. This means that Corda must guarantee that the contract code produces the same output for all peers, each time it is executed. This means that the contract code cannot contain any non-determinism.

Corda executes the contract code within a 'sandboxed' environment that guarantees determinism. The contract code does not have any storage or the ability to interact with anything more than the supplied transaction.

A contract has two items associated with it:

- Contract code
- Legal prose, this references a legal contract underpinning the agreement. Corda does not subscribe to 'Code is law'.

2.1.5 Legal prose

Whilst the contract code is a powerful means to verify transactions, it cannot handle all real-world events. Therefore, Corda contracts support the inclusion of legal prose, which can be relied on when contract code alone is insufficient. Legal prose states the rules governing the evolution of the state over time in a way that is compatible with traditional legal systems.

The legal prose is defined as a template and parameters. The templates are the contract blueprints. The parameters create instances of the template. A hash of the legal prose is referenced in a contract.

The legal prose remains constant for the duration of a state's lifecycle unless required peers opt to novate.

2.1.6 Commands

States can evolve in multiple ways. Corda needs more information than available only in the states to determine how to verify a transaction. Commands parameterize transactions hinting to their intent and specify the required signers via a list of public keys.

Commands may also contain arbitrary data signed by an oracle service.

2.1.7 Time windows

Transaction time windows specify a time within the transaction is asserted to have occurred. This is expressed as windows, because in larger distributed systems there is no true time, only many desynchronized clocks.

The purpose of a time window is to communicate the transaction's position on the timeline to the contract code for the enforcement of contractual logic.

Time windows may be open ended in order to communicate that the transaction occurred before or after a certain time. How much before or after is unimportant.

To enforce time windows, notaries must sign the transaction within some time tolerance, acting as a timestamping authority.

2.1.8 Flows

Corda networks use point-to-point messaging instead of a global broadcast. This means that coordinating a ledger update requires network participants to specify exactly what information needs to be sent, to which counterparties, and in what order.

Flow framework is what you use to write the core business logic of your CorDapp. Flows are check-pointed state-machines. A sequence of steps that tells a node how to achieve a specific ledger update. Communication between nodes only occurs in the context of these flows and is point-to-point. Once a given business process has been encapsulated in a flow and installed on the node as part of a CorDapp, the node's owner can instruct the node to kick off this business process at any time using an RPC call.

The flow framework allows nodes to have many flows active at once. These flows may last days, across node restarts and even upgrades.

This is achieved by serializing flows to disk whenever they enter a blocking state (e.g. when they're waiting on I/O or a networking call). Instead of waiting for the flow to become unblocked, the node immediately starts work on any other scheduled flows, only returning to the original flow later.

2.1.9 Consensus

To determine whether a proposed transaction is a valid, peers reach consensus via two types of consensus: Verification consensus and Uniqueness consensus.

Verification consensus involves peers reaching certainty that a transaction is signed by all required peers and satisfies the constraints in the contract code. For newly proposed transactions verification consensus is performed only by the peers involved.

When a peer is presented with a transaction containing input state references, the prior transaction which created the reference output needs to be verified. This need to happen recursively for all prior transactions back to the issuance transaction for the input state reference in the proposes transaction. If any historic transactions are invalid, then the proposed transaction cannot be accepted.

Uniqueness consensus involves peers reaching certainty that the output states created in the transaction are the unique successors to the input states referenced by that transaction. This is because peers want certainty that an input state has not

been previously been marked historic. The ''Double spend'' problem for on-ledger assets is solved with uniqueness consensus.

2.1.10 Notaries

Notary services provide uniqueness consensus. When a peer wished to commit a transaction, signatures are required from all peers with public keys listed in the commands. In addition, a signature from the notary service specified for the input state reference is required. This is referred to as notarization.

The point of transaction finality is reached when the specified notary service signs the transaction. Where the output states are the unique successors for all referenced input states.

In simple terms, notaries maintain a map keyed with input state references. When a proposed transaction is sent to a notary service, the notary checks if any of the input state references are already in the map. If so, the notary will throw an exception and notes that there is a conflict. If not, the notary adds each input state to the map and signs the proposed transaction.

Corda does not order transactions using a blockchain. Instead each state points to a notary, which guarantees it will sign a transaction only if all the input states are unconsumed.

There are two types of notaries in this. Verifying and non-verifying notaries. Non-verifying notaries only check whether a state has been previously used and marked as historic. They do not perform verification over the transaction and do not see the content of state objects.

Verifying notaries perform the same workflow, but they add an additional step to perform transaction verification.

2.1.11 Oracles

The contract code defines a pure function which cannot depend on any sources of non-determinism. Often it is necessary to condition a transaction with some off-ledger fact being true or false. As all calculations on the ledger must be deterministic. Everyone must be able to check the validity of a transaction and arrive at the same answer every time. Corda addressed this issue by introducing oracles. Services that create digitally signed data structures which assert facts.

Oracles are an authority that attest to and may also provide off-ledger facts needed to verify transaction proposals.

Flows are used to communicate with oracles. Oracles are available during proposal and verification of the transaction. Multiple oracles can be used per transaction.

For privacy purposes, the oracle does not require to have access on every part of the transaction and the only information it needs to see is their embedded command(s), related to this oracle.

2.1.12 Node

A Corda node is a JVM run-time environment with a unique identity on the network that hosts Corda services and CorDapps.

A CorDapp stand short for a Corda Distributed Application. This is the combination of state objects, contract code, the flow framework, any necessary APIs, a vault, plugins and UI components. The CorDapp provider is where new CorDapps are installed to extend the behavior of the node.

The node has two interfaces with the outside world. A network layer, for interacting with other nodes. And an RPC interface, for interacting with the node's owner.

Corda is designed for semi-private networks. Admission to the network requires obtaining an identity signed by a root authority. The network map service publishes the IP addresses through which every peer on the network can be reached, along with the identity.

3 Chapter III - Correspondent Banking

The transfer of value, or payment is one of the oldest and most widely used customs in the world. Modern economics depend on transfer of value within the banking systems. The payer and payee respectively experience the debits and credits materializing in their accounts. But behind these movements lies a complex exchange of instructions, confirmations, checks and reports, netting and settlement processes.

Correspondent banking is an informal network of banks around the world, connected by a series of contractual agreements. This can be defined, in general terms, as an arrangement under which one bank (correspondent) holds deposits owned by other banks (respondents) and provides payment and other services to those respondent banks.

Correspondent banking is an essential component of the global payment system, especially for cross-border transactions. Through correspondent banking relationships, banks can access financial services in different jurisdictions and provide cross-border payment services to their customers.

At the most basic level, correspondent banking requires the opening of accounts by respondent banks in the correspondent bank books and the exchange of messages to settle transactions by crediting and debiting those accounts.

The accounts held between correspondent banks, and the banks to which they are providing services, are referred to as nostro and vostro accounts. An account held by one bank for another is referred to by the holding bank as a nostro account. The same account is referred to as vostro account by the counterparty bank. Both banks in a correspondent relationship hold accounts for one another for the purpose of tracking debits and credits between parties.

A correspondent bank is a financial institution that is authorized to provide services on behalf of other financial institutions.

3.1 SWIFT

SWIFT is a global member-owned cooperative and the world’s leading provider of secure financial messaging services. SWIFT plays a pivotal role in the correspondent banking system by providing a safe, secure and confidential platform for the exchange of these instructions and pieces of information between different parties.

According to a report by the US Treasury on the fundamentals of the funds transfer process, SWIFT messages direct the transfer of nearly $5 trillion worldwide each day and is the primary method for international funds transfer messages.

3.1.1 Payment Messaging

Correspondent banking transactions are channeled and settled through a chain of bilateral relationships between respondent and correspondent banks. SWIFT allows channeling of a correspondent banking transaction through the SWIFT network.

The SWIFT messaging platform connects more than 11.000 banking and securities organisations, market infrasturctures and corporate customers in more than 200 countries and territories.

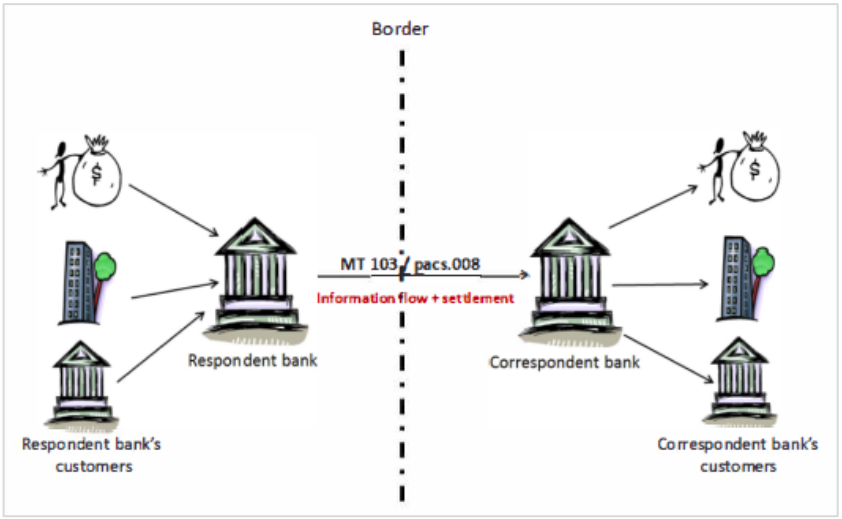


Figure 1: A simple cross-border correspondent transaction *(source: bis.org)*

This figure shows a simple cross-border correspondent transaction. However, in many cases the respondent bank originating the payment does not have a direct bilateral account relationship with the correspondent bank receiving the payment. In these cases, it is necessary to find a chain of one or more intermediary banks to transmit the funds from originating bank to the receiving bank.

3.1.2 Global Payments Innovation (gpi)

When transferring money cross-border, customers currently may still encounter frictions, cost and delays. SWIFT gpi was developed to resolve this. Enabling cross-border payments to be fast, transparent, trackable across multiple banks and deliver confirmation when the beneficiary account is credited.

With SWIFT gpi, the correspondent banking community, together with fintech's, corporates , and others is collectively removing frictions and reducing the costs associated with cross-border payments.

A total of 300 billion dollars' worth of payments are sent every day using the new gpi standard. Payments are made quickly. Overall over 50% of SWIFT gpi payments are credited to end beneficiaries within 30 minutes, and almost 100% of payments within 24 hours.

3.1.3 GPI Link

SWIFT gpi Link opens gpi payment capabilities to e-commerce and DLT platforms. It is a gateway to interlink e-commerce and trading platforms with SWIFT gpi. With gpi link, banks will be able to provide rapid, transparent settlement services to trade ecosystems.

The gateway enables the monitoring and control of payment flows. This includes payment initiation, end-to-end payment tracking, payer authentication and credit confirmation.

4 Chapter IV – Catalyst

Catalyst is IntellectEU's blockchain integration platform that easily connects traditional infrastructures with any distributed ledger network.

In this project, Catalyst will act as a proxy to gpi Link. The Corda blockchain application will interact with Catalyst, which will relay the requests and responses to and from gpi Link to the CorDapp.

The future view of Catalyst is to enable connections to numerous payment rails, to enable off-ledger settlements for DLT and non-DLT technologies.

More info on Catalyst can be required through catalyst.intellecteu.com or info@intellecteu.com

5 Chapter V - Corda-Marketplace

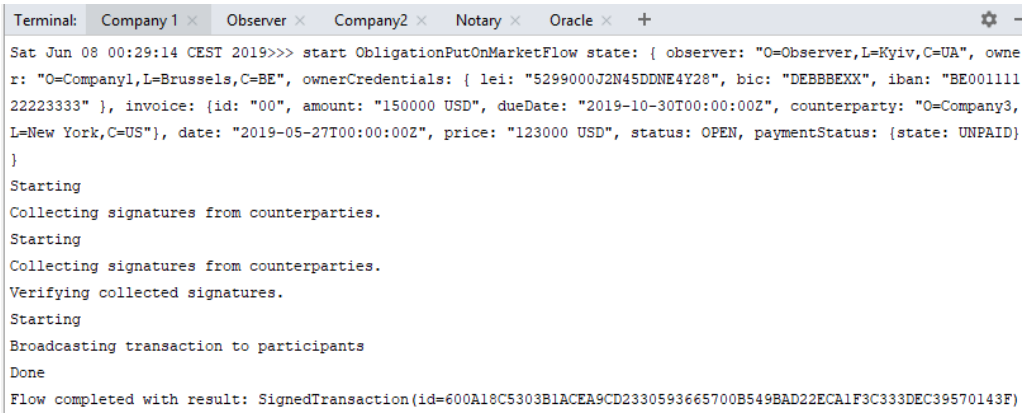
The marketplace-corda implements the different business flow to facilitate the process of delivery vs payment in invoice factoring. The flows are made accessible via API endpoints. Together with API endpoints to query the ledger for invoices.

To view the code please contact jmayeur@outlook.com or info@intellecteu.com

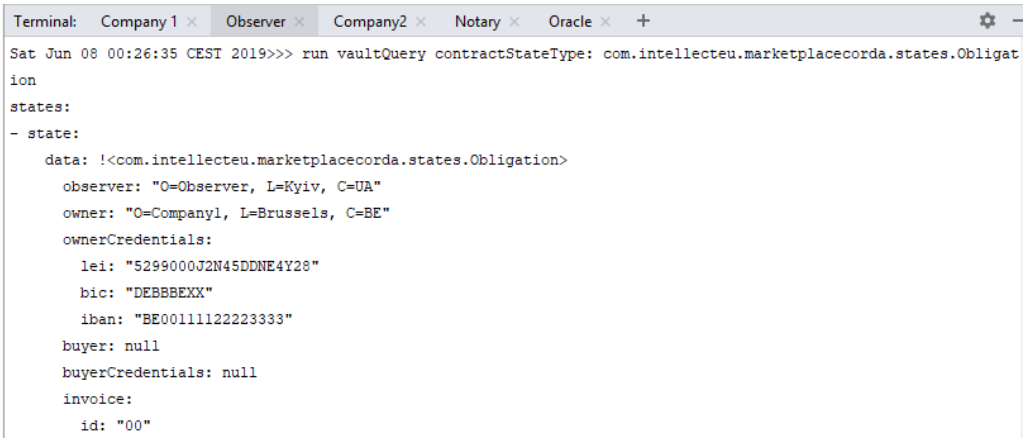
5.1 Local deployment demo

We can start a corda node by running the corda.jar file for that node in a shell. Below we will demo the working of the CorDapp, deployed on three nodes, company1, company2 and observer node. Company1 will put an obligation with an invoice on the market. The BuyFlow is run from the observer node and sets company2 as the buyer. Company2 can then pay for the obligation, with payment initiation going through Catalyst to gpi Link. After company2 signed the payment, it can call the settle flow, which will check for the payment status of the payment. Depending on the returned result, the output state is changed.

First, we initiate the putOnMarketFlow from the Company1 node.



We can see the state we put on the market in the observer node's vault. We check the linearId of the state and use it to reference the input in the BuyFlow.



```
    paymentStatus:
      state: "UNPAID"
      uetr: ""
      signatureStatus: "NOT_SIGNED"
    participants:
    - "O=Company1, L=Brussels, C=BE"
    - "O=Observer, L=Kyiv, C=UA"
    linearId:
      externalId: null
      id: "bdaf510e-87ca-4fee-8dc0-18384c1fe342"
```

We start the BuyFlow from the Observer node, using the linearId of the obligation.

When we check the vault of company2 we can find the state now also.

```
Terminal:  Company 1  Observer  Company2  Notary  Oracle  +
Sat Jun 08 00:36:10 CEST 2019>>> start ObligationBuyFlow obligationId: "bdaf510e-87ca-4fee-8dc0-18384c1fe342", buy
er: "O=Company2,L=London,C=GB", buyerCredentials: {lei: "6299300D2N76ADNE4Y55", bic: "CREDGB2L", iban: "BE99888877
776666"}
Starting
Collecting signatures from counterparties.
Starting
Collecting signatures from counterparties.
Starting
Collecting signatures from counterparties.
Verifying collected signatures.
Starting
Requesting signature by notary service
Requesting signature by Notary service
Validating response from Notary service
Broadcasting transaction to participants
Done
Flow completed with result: SignedTransaction(id=40DB46E01AAB4801AE8592428AE14E125015FF8F2D6798C72E3A339F40315670)
```

The status field is now changed to BOUGHT.

```
Terminal:  Company 1  Observer  Company2  Notary  Oracle  +
Sat Jun 08 00:27:30 CEST 2019>>> run vaultQuery contractStateType: com.intellecteu.marketplacecorda.states.Obligat
ion
states:
- state:
    data: !<com.intellecteu.marketplacecorda.states.Obligation>
      observer: "O=Observer, L=Kyiv, C=UA"
      owner: "O=Company1, L=Brussels, C=BE"
      ownerCredentials:
        lei: "5299000J2N45DDNE4Y28"
        bic: "DEBBBEXX"
        iban: "BE00111122223333"
      buyer: "O=Company2, L=London, C=GB"
      buyerCredentials:
        lei: "6299300D2N76ADNE4Y55"
        bic: "CREDGB2L"
        iban: "BE99888877776666"
      invoice:
        id: "00"
        amount: "150000.00 USD"
        dueDate: "2019-10-30T00:00:00Z"
        counterparty: "O=Company3,L=New York,C=US"
      date: "2019-05-27T00:00:00Z"
      price: "123000.00 USD"
      status: "BOUGHT"
      paymentStatus:
        state: "UNPAID"
        uetr: ""
        signatureStatus: "NOT_SIGNED"
      participants:
      - "O=Company1, L=Brussels, C=BE"
      - "O=Observer, L=Kyiv, C=UA"
      - "O=Company2, L=London, C=GB"
      linearId:
```

```
    externalId: null
    id: "bdaf510e-87ca-4fee-8dc0-18384c1fe342"
```

Now we can start the PayFlow. This will communicate with the oracle. The initiation of the handlers in the oracle can be shown via the log files from the oracle. When we check the state in the vault the paymentStatus state says PAYMENT_CREATED. Signature status NOT_SIGNED.

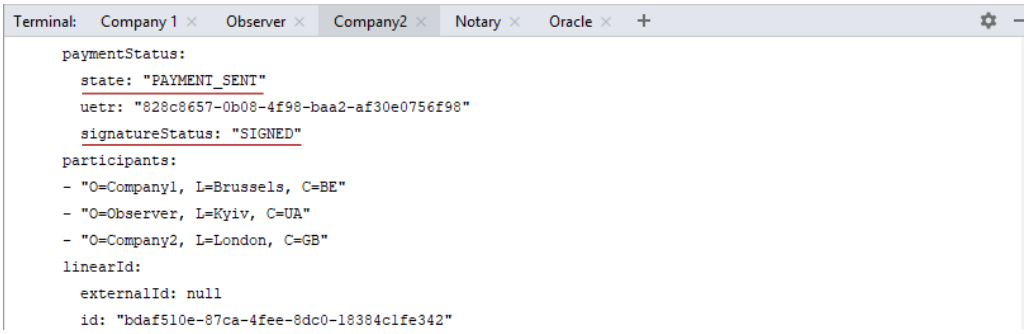
```
Terminal:  Company 1  Observer  Company2  Notary  Oracle  +
Sat Jun 08 00:37:09 CEST 2019>>> start ObligationPayFlow linearId: "bdaf510e-87ca-4fee-8dc0-18384c1fe342", oracle:
 "O=Oracle,L=London,C=GB", endToEndIdentification: "Invoice1234", creditorName: "creditor", debtorName: "debtor"
Starting
Collecting signatures from counterparties.
Starting
Collecting signatures from counterparties.
Starting
Collecting signatures from counterparties.
Verifying collected signatures.
Starting
Requesting signature by notary service
Requesting signature by Notary service
Validating response from Notary service
Broadcasting transaction to participants
Done
Flow completed with result: SignedTransaction(id=274C6EAB4CC0F376E3DFB6801ACD8AE2D6D1460166011780320841A1836A0DEB)
```

```
\Oracle\logs
Id                Flow name                              Initiator          Status
-------------------------------------------------------------------------------------------------------------------
2b37ec43-c414-4e74 Initiate Payment Handler              O=Company2, L=Lon No return value
b7497la8-7950-45c2 Sign Payment Initiation Handler       O=Company2, L=Lon No return value
```

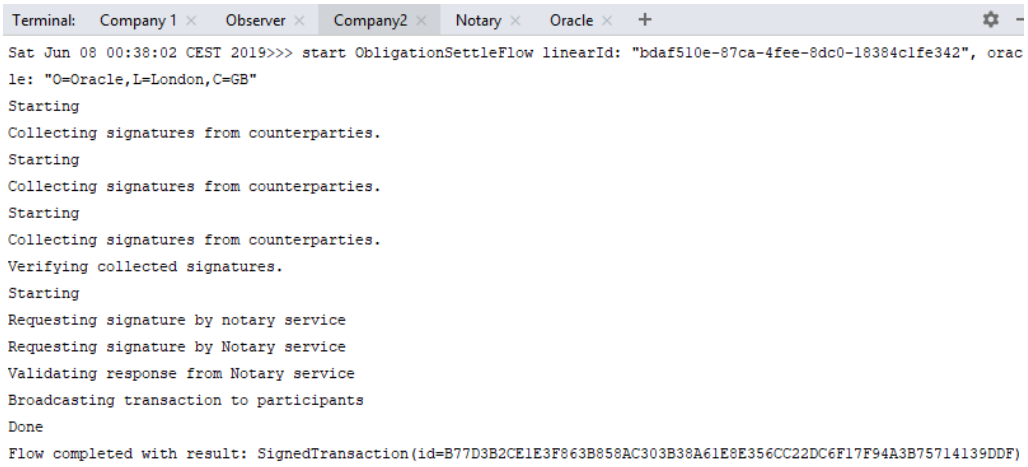
```
Terminal:  Company 1  Observer  Company2  Notary  Oracle  +
    price: "123000.00 USD"
    status: "BOUGHT"
    paymentStatus:
      state: "PAYMENT_CREATED"
      uetr: "828c8657-0b08-4f98-baa2-af30e0756f98"
      signatureStatus: "NOT_SIGNED"
    participants:
    - "O=Company1, L=Brussels, C=BE"
    - "O=Observer, L=Kyiv, C=UA"
    - "O=Company2, L=London, C=GB"
    linearId:
      externalId: null
      id: "bdaf510e-87ca-4fee-8dc0-18384c1fe342"
```

To sign the payment, we start the SignFlow from the company2 node.

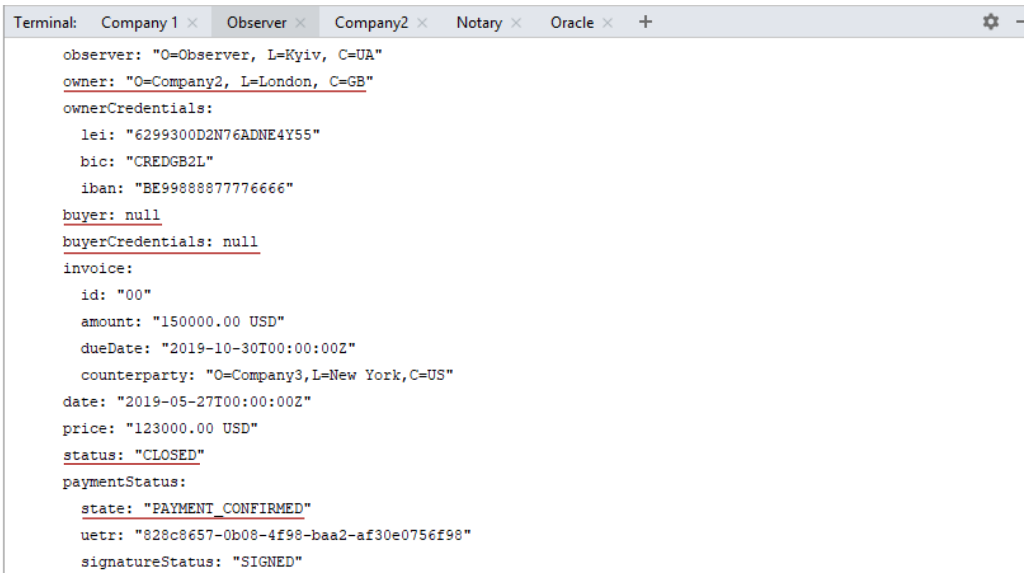
```
Terminal:  Company 1  Observer  Company2  Notary  Oracle  +
Sat Jun 08 00:37:53 CEST 2019>>> start ObligationSignFlow linearId: "bdaf510e-87ca-4fee-8dc0-18384c1fe342", oracle
: "O=Oracle,L=London,C=GB", payload: "cert1237"
Starting
Collecting signatures from counterparties.
Starting
Collecting signatures from counterparties.
Starting
Collecting signatures from counterparties.
Verifying collected signatures.
Starting
Requesting signature by notary service
Requesting signature by Notary service
Validating response from Notary service
Broadcasting transaction to participants
Done
Flow completed with result: SignedTransaction(id=DF51F6AADD45877673980BA4F419B4708468AAEC9C4F1EB224B97196F952F90C)
```



Once payment is signed, we can start the SettleFlow.



After payment was confirmed, we have change of ownership of the obligation. Here we query the state in the vault of the observer.



Conclusion

This paper explains how a blockchain-based application can be integrated with the SWIFT gpi payment system for off-ledger settlement. With this demonstration, we prove it is possible to connect a new distributed world with traditional payment rails, in a secure and seamless manner.

During this internship I learned about blockchain technology, and its potential for the financial services industry. With that knowledge I co-developed a unique application that combines the different fields and showcases the possibilities of blockchain technology in the financial services industry.

As a result of my work, the project is on track to be deployed to different Tier 1 banks that are participating in the demo. And will be showcased at Sibos 2019, an annual banking and financial conference organized by SWIFT.

Bibliography

Bank for International Settlements. (2016, July). *Correspondent Banking.* Retrieved from Bis.org: www.bis.org/cpmi/publ/d147.pdf

Brown, R. G. (2018, May). *The Corda Platform: An Introduction.* Retrieved from Corda: www.corda.net/wp-content/uploads/2018/05/corda-platform-whitepaper.pdf

Buterin, V. (2015, August 6). *On Public and Private Blockchains*. Retrieved from blog.ethereum.org: https://blog.ethereum.org/2015/08/07/on-public-and-private-blockchains/

Digital Asset Holdings. (2016, December). *The Digital Asset Platform.* Retrieved from digitalasset.com: https://hub.digitalasset.com/hubfs/Documents/Digital%20Asset%20Platform%20-%20Non-technical%20White%20Paper.pdf

Docs.corda.r3.com. (n.d.). Retrieved from https://docs.corda.r3.com/_static/corda-developer-site.pdf?cv=1

Don, T., & Alex, T. (2016). *Blockchain Revolution.* Portofolio/Penguin.

Fincen.gov. (n.d.). *Fundamentals of the Funds transfer process.* Retrieved from Fincen.gov: www.fincen.gov/sites/default/files/shared/Appendix_D.pdf

Investopedia.com. (2019, April 20). *What You Should Know About Correspondent Banks*. Retrieved from Investopedia: www.investopedia.com/terms/c/correspondent-bank.asp?ad=dirN&o=&qo=&qsrc=0

O'Neal, S. (2019, February 3). *SWIFT Announces PoC Gateway With R3*. Retrieved from cointelegraph.com: https://cointelegraph.com/news/swift-announces-poc-gateway-with-r3-but-remains-overall-hesitant-about-blockchain

R3. (2018, December 5). *R3 Launches Universal Settler Application to Facilitate Global Payments on Corda*. Retrieved from Corda: https://www.r3.com/news/r3-launches-universal-corda-settler-application/

R3. (2019, May 9). *Corda Enterprise Documentation.* Retrieved from Corda: https://docs.corda.r3.com/_static/corda-developer-site.pdf

R3. (n.d.). *Corda | Technology.* Retrieved from Corda: www.corda.net/discover/technology.html

R3. (n.d.). *R3 Corda Master documentation*. Retrieved from Corda: https://docs.corda.net/

Ray, S. (2018, February 19). *The Difference Between Blockchains & Distributed Ledger Technology*. Retrieved from towardsdatascience.com:

https://towardsdatascience.com/the-difference-between-blockchains-distributed-ledger-technology-42715a0fa92

Sepaforcorporates.com. (2018, October 8). *SWIFT gpi UETR*. Retrieved from SEPA for Corporates: https://www.sepaforcorporates.com/swift-for-corporates/explained-swift-gpi-uetr-unique-end-to-end-transaction-reference/

Steemit.com. (2017). *Here's Why Blockchains Will Change the World*. Retrieved from Steemit.com: https://steemit.com/blockchain/@dubl.inc/here-s-why-blockchains-will-change-the-world

Steemit.com. (n.d.). *Here's Why Blockchains Will Change the World*. Retrieved from Steemit.com: https://steemit.com/blockchain/@dubl.inc/here-s-why-blockchains-will-change-the-world

Swift.com. (2019). Retrieved from SWIFT: https://www.swift.com/about-us/media-centre?AKredir=true

Swift.com. (2019). Retrieved from SWIFT: https://www.swift.com/news-events/news/swift-to-bring-benefits-of-gpi-to-dlt-and-trade-ecosystems

Swift.com. (2019, March). *SWIFT gpi.* Retrieved from SWIFT: www.swift.com/file/60026/download?token=1zpr-lyU

Swift.com. (2019, January 30). *SWIFT to open gpi to e-commerce and DLT platform.* Retrieved from SWIFT: www.swift.com/resource/pr-swift-open-gpi-e-commerce-and-dlt-platforms

Swift.com. (2019). *swift-gpi-brochure*. Retrieved from Swift.com: https://www.swift.com/resource/swift-gpi-brochure

Swift.com. (n.d.). *Corporate Brochure.* Retrieved from swift: www.swift.com/node/23621