DCB SERVICES & WHOLESALE CBDC CONCEPT EUROchain Hackathon

BANQUE CENTRALE DU LUXEMBOURG
EUROSYSTEM

CONTENTS

This paper describes a prototype of decentralised central bank (**DCB**) Services, allowing the use of a central bank digital currency (**CBDC**), developed during the fourth EUROchain hackathon organised by the Banque de France in Paris in January 2020. This prototype is a blockchain-based central bank system built on Corda that allows securities settlement, collateral management and payments. It was developed for pure testing purposes, and Corda was chosen as a private blockchain as it best reflected present central bank practices.

In addition, this system's interoperability with the public blockchain Ethereum was developed to create a first instance of cross-blockchain transferable wholesale CBDC in order to enable wholesale CBDC tokens to be used as a settlement asset for business cases developed on other blockchain platforms. Ethereum was chosen as it is a public blockchain commonly used in business environments.

This prototype is not indicative of any decision by the Banque Centrale du Luxembourg (BCL) or the Eurosystem to offer blockchain-based services or to issue a CBDC. Any views expressed are solely those of the author(s) and so do not necessarily represent the views or policies of the BCL.

1. EUROchain hackathons

In 2017, the Market Infrastructure and Payments Committee (**MIPC**) of the Eurosystem established a working group called the MIPC DLT network. Its aim was to explore the distributed ledger technology (**DLT**) by defining prototypes in the field of market infrastructures and payment systems and deploying a (test) DLT network amongst Eurosystem central banks. To develop these prototypes, the group organises hackathons[1] on average every 8 months (i.e. "EUROchain hackathons").

The BCL has participated in the last three hackathons and proposed several prototypes that found recognition amongst the jury and were awarded. Examples of such include the "DLT2S" project (a basic T2S contingency module built on Corda), the "CreditChain" prototype (a credit claim management prototype built on Hyperledger) and finally the DCB Services and wholesale CBDC built on Corda and Ethereum.

In the most recent hackathon, the BCL was accompanied by Neofacto[2] , a Luxembourg-based IT consultancy company with extensive expertise in blockchain technology and particularly in Ethereum. In addition, representatives from Banque de France, Deutsche Bundesbank and Lietuvos bankas joined the team during the hackathon. Throughout this paper, subsequent usage of "we" refers to this team.

[1] A hackathon is a «hacking-marathon» where participants develop a workable simple application/proof of concept in a short period of time (2-3 days).
[2] http://neofacto.com

2. DCB Services & wholesale CBDC concept

This prototype consists in replicating existing central bank services in a decentralised fashion and allowing cross-blockchain interoperability using two DLT platforms, namely Corda and Ethereum[3]. It integrates the concept of TARGET services (T2, TIPS, T2S, ECMS) called DCB Services built on a Corda blockchain and encompasses the following:

- a basic instant payment module, allowing participant banks to instruct instant payments in euro central bank money, using the wholesale CBDC developed in the prototype, intending to replicate TIPS and T2 services existing in TARGET services today.

- a basic securities settlement module, based on a previously developed prototype (DLT2S), which replicates the core of existing T2S services in a decentralised technology. It allows delivery versus payment (**DVP**) settlement in euro central bank money.

- a simple collateral management system, allowing participant banks to obtain a credit line and liquidity against eligible collateral.

Besides the DCB Services themselves (i.e. instant / inter-bank payment module, securities settlement and collateralisation), the prototype offers participant banks the possibility to transfer their liquidities held in DCB Services to any other connected blockchain. For this project, the public blockchain Ethereum has been used to test the interoperability between blockchains. This interoperability allows participant banks to use their balances held at the central banks to settle transactions in central bank money outside the DCB Services sphere, as long as the central bank can control the Ethereum smart contracts used. The starting point was that both participant banks and central banks could benefit from this possible use of a wholesale CBDC.

To mimic euro central bank money settlement outside central bank market infrastructures, we have designed and implemented equity option contracts on Ethereum that the participant banks could create, enter into and subsequently exercise. By allowing the wholesale CBDC to be transferred between the DCB Services on Corda to Ethereum, we have enabled the wholesale CBDC to become a settlement asset for the payment of premiums and the exercise of these derivative instruments.

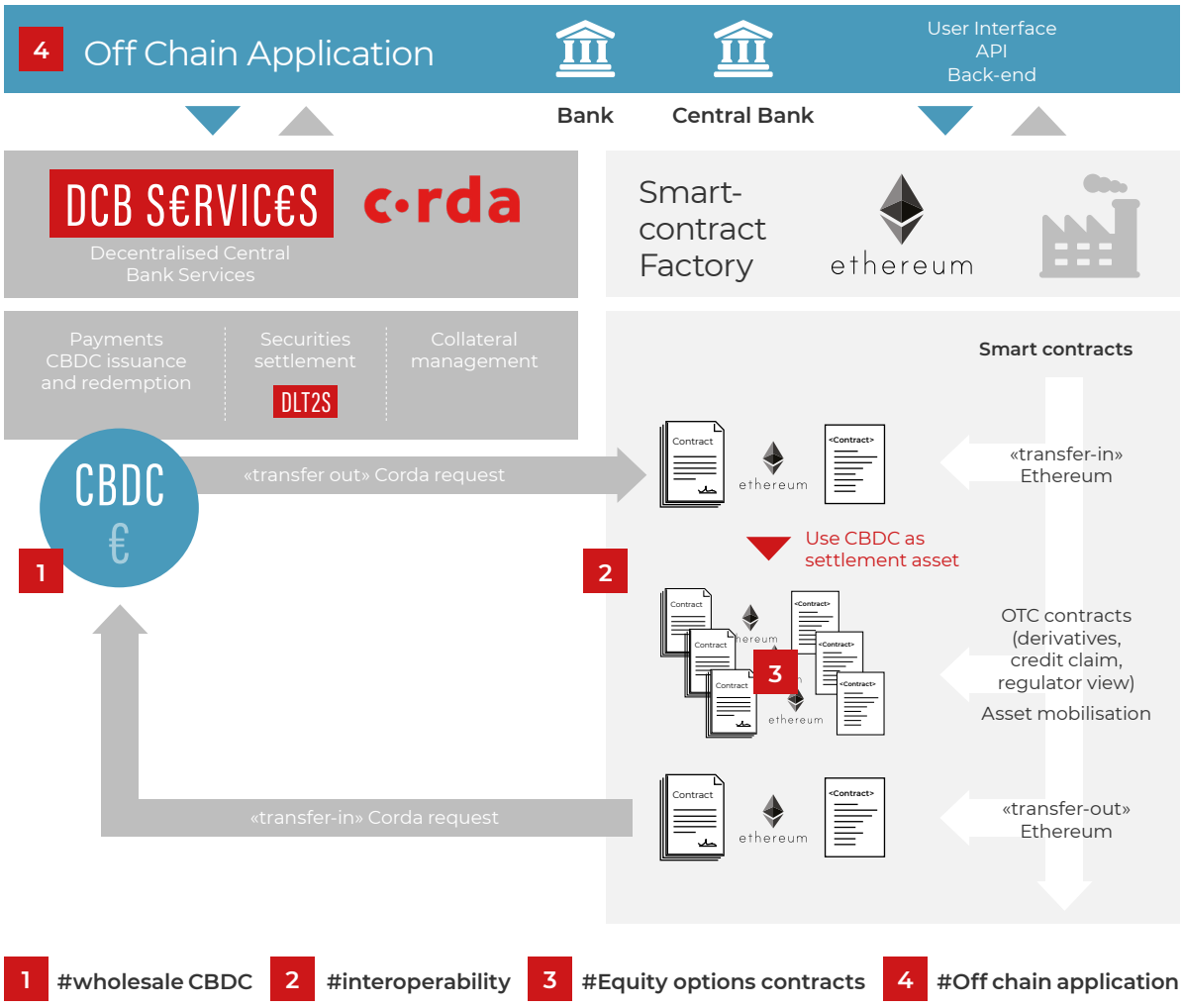
Neofacto helped us design and integrate these contracts with the Corda environment. In order to control the circulation of the wholesale CBDC tokens, we implemented a whitelisting of Ethereum addresses on the DCB Services side, and by means of API calls on the Ethereum blockchain, we could trace the CBDC holdings and ensure consistency in the amount of CBDC between the two platforms.

[3] The test version of Ethereum was used

This feature is important as for a CBDC, a central bank needs to know its creditors. All the contracts created in Ethereum are publicly available in a "pseudonymous" way (only public addresses of banks are shared). In our model, central banks are able to know the name of the issuer and the buyer of the contracts as their public key is first registered in DCB Services by central banks.

To summarise, this proof of concept aimed at:

- creating "TARGET-like" services on a private DLT environment (Corda);
- proving the interoperability between two different DLTs (Corda and Ethereum);
- building a transferable wholesale CBDC; and
- identifying the potential benefits of public DLTs.



1 #wholesale CBDC 2 #interoperability 3 #Equity options contracts 4 #Off chain application

Figure 1 - DCB services overview

3. Functional description

The prototype includes the following set of key functionalities:

Static data creation

Central banks manage the static data creation in an off-chain application, accessible via a graphical user interface available through secured access. They are responsible for creating profiles of participants (including banks and Central Securities Depositories (CSDs)) with all their relevant details (name, BIC, etc.), to be used in Corda and Ethereum. During this phase, an Ethereum smart contract associates an Ethereum wallet public key to the participant bank party. This process allows central banks to follow the wholesale CBDC transfers on Ethereum by whitelisting potential recipients of the wholesale CBDC.

Cash issuance

Central banks run a Corda node and can issue/credit the wholesale CBDC developed in this prototype to the relevant banks.

Payments

Once the participating banks have wholesale CBDC balances reflected in the DCB Services, they can use them in order to make payments to the other participating banks registered in the off-chain application. Payments are settled via central bank operated nodes in Corda. The participating banks can also use their CBDC balances for securities settlement or liquidity transfers to Ethereum.

Securities issuance and settlement

CSDs run Corda nodes and can issue securities and credit initial balances to their participants via the graphical user interface (hosted in the off-chain application). A full securities referential is available including most of the securities common fields (including the ISIN, name, Eurosystem collateral eligibility flag, currency, type, issuer, Eurosystem haircut for use as collateral, etc). Once created or issued on Corda, the securities can be exchanged via DVP with other banks participating in DCB Services. CSD and participating banks can monitor their securities portfolios thanks to the off-chain application providing the graphical user interface (GUI) and relevant access rights to each user. All securities transactions are settled in Corda nodes, within one CSD node for domestic settlements or between two (or more) CSD nodes for cross-border settlements. In our prototype, we created two CSD nodes and successfully performed all the possible scenarios. We also worked on optimisation algorithms to improve the settlement efficiency.

Collateral Management

Participant banks can obtain liquidity by mobilising eligible securities to the central bank. At present, this feature is basic as it was not the focus of the proof of concept. It might be enhanced at a later stage.

CBDC transfer between Corda and Ethereum

Participant banks can transfer all or part of their liquidity held on Corda to their Ethereum public address (created at the time of registration in the DCB services system in Corda). When the bank places a request on the GUI, a smart contract deposit function is triggered on Ethereum to credit its address. The requested amount of wholesale CBDC is then reserved on the Corda side with an intermediary state "ETH reserved status", until the issuance and transfer are confirmed on the Ethereum side and the state of transfer is finalised in Corda. The bank can no longer spend this amount on Corda once this status is set. After a predefined period of time or if the transfer to Ethereum fails, the status is changed back to "Corda available status", following a check that no wholesale CBDC has been created on Ethereum, to avoid liquidity trapping if the transfer does not work as intended.

If the Ethereum issuance and transfer succeed, the status in Corda is changed to a finalised state "ETH finalized status" (the amount is available in Ethereum and still recorded as "unavailable" for control purposes in Corda). The participant can see on the GUI that the CBDC amount is available on its Ethereum address. Similar rollback processes apply for transfers from Ethereum to Corda.

OTC contract creation and lifecycle

Once participant banks hold CBDC on their Ethereum addresses, they can use it to initiate and exercise equity option contracts. There is an equity option creation module allowing these banks to create call or put equity options. They can create a basic equity option including most of the commonly required fields (like call/put, category, type, issuer, counterparty, strike price, expiration date, underlying asset, etc). Once created, the counterparty bank can accept the contract. Upon acceptance of the contract, an Ethereum contract will trigger the wholesale CBDC transfer between the Ethereum addresses of the two banks for the payment of the premium in exchange of the option token. At a later stage, the relevant bank can exercise the option. Based on the predefined conditions of the contract and the underlying market price, the option might be "in the money" when it can be exercised by the owner. If the owner decides to exercise the option, the related amount of CDBC will be transferred from the option writer's account to the option owner's account (if the writer account balance is too low, a "default" event will be triggered). The option will then be considered as "exercised" (unless a default occurred). If the option is not "in the money" at the moment that it can be exercised, it will simply expire.

Monitoring dashboards

The GUI contains monitoring dashboards (see screenshot below) for all the actors, and data is retrieved via API calls on the two blockchains. The banks' dashboards include wholesale CBDC balances in both Corda and Ethereum, securities positions at their CSD(s), history of payments and DVP transactions (in Corda and in Ethereum), the list of open equity option contracts and valuation.

The CSD's dashboards include their participant banks' securities positions and all DVP transactions history. The central banks' dashboards include all the CBDC balances and all the related transactions. Central banks can also see all option contracts details, including total exposures of their participating banks. Finally, central banks can visualise in real time the cost in Ether (ETH) for the all the Ethereum smart contracts (transfers IN/OUT and OTC smart contracts).

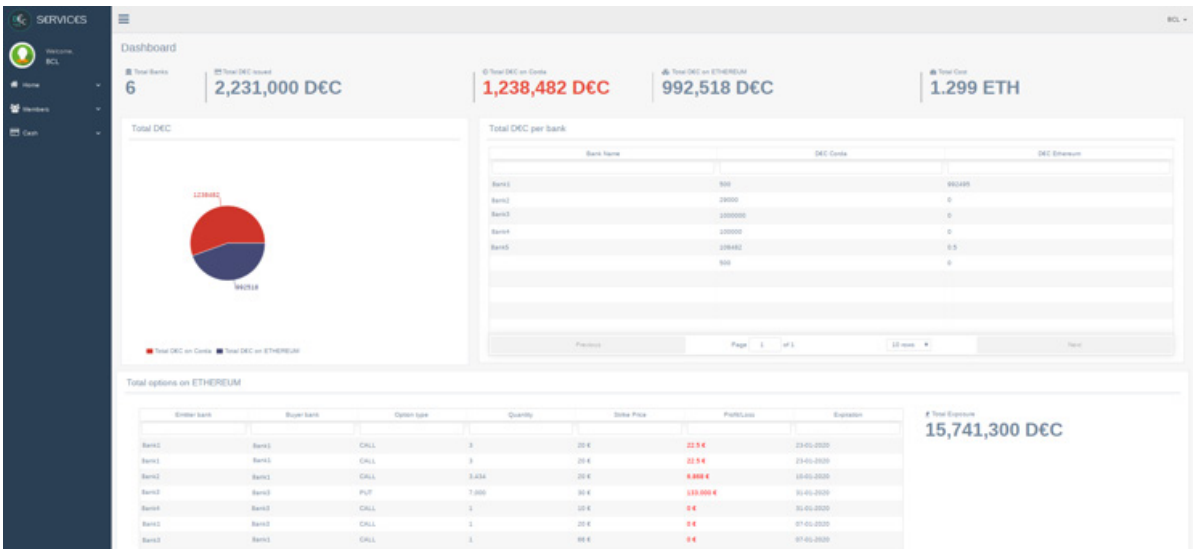


Figure 2 – Central Bank Dashboard (D€C = CBDC in this graph)

4. Technical architecture

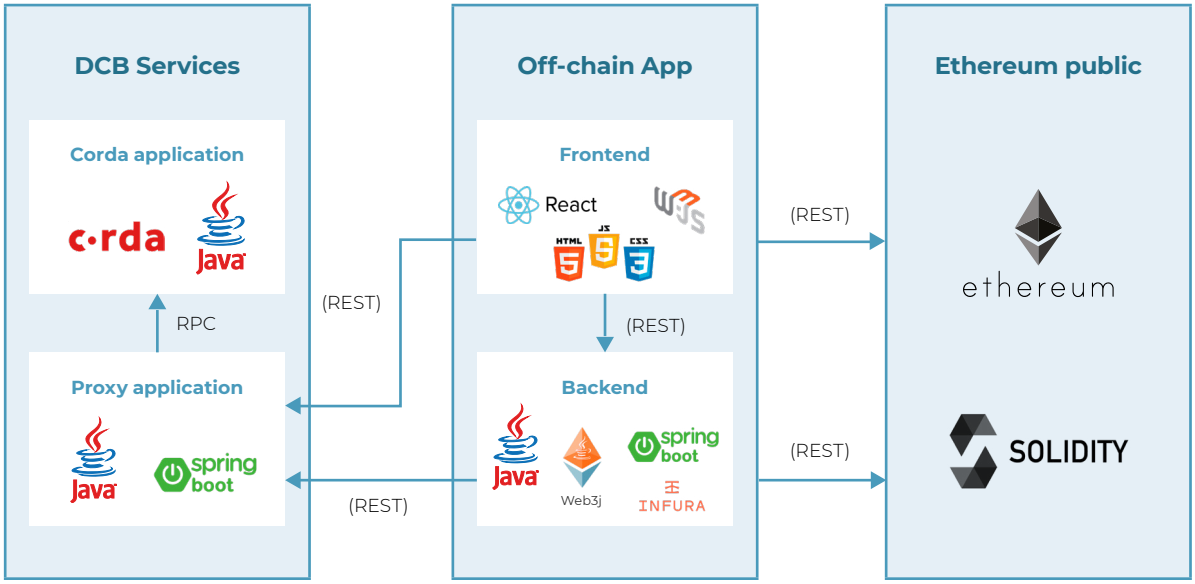


Figure 3 – Technical architecture

4.1 DCB Services on Corda

Corda is a Distributed Ledger Technology that was designed primarily for financial institutions, although Corda makes it possible to create a DLT for any use case. The framework contains three essential components:

- Corda node: the node of an organisation running multiple "Cordapps" (Corda applications) which is incorporating Corda's smart contracts.
- Notary node: this maintains the consensus, avoiding double spending.
- Network map: this permits communication between the various network nodes.

Corda proposes a simplified consensus through the intermediation of the notary function, which is responsible for controlling and signing transactions in the network. In our prototype, the Corda network hosts the DCB Services on three Corda nodes and one notary:

- one central bank node;[4]
- two CSD nodes; and
- one notary node.

For each participant of the network, two applications are deployed on a cloud provider (Amazon Web Services) running on a Java Virtual Machine (JVM). The first application is the node itself coded in pure Java. Corda requires messages to be exchanged between nodes via a defined protocol: AMQP/1.0 over TLS. The second one is a Spring Boot application, which communicates with its node via RPC and exposes services to our off-chain application via a REST API.

[4] Only CSD and CBs have a node on Corda. Participant banks do not run a node on Corda in this prototype. They access the different functionalities via CSD and CB nodes. In practice, 19 NCBs (Eurosystem) and 23 CSDs (in T2S) would have a node, but different layouts could be envisaged.

4.2 Ethereum (or any other guest network) smart contracts

About Ethereum

Unlike DCB Services where there is a need to host four Corda nodes, none of the actors involved in this prototype has to run any node on the public Ethereum network to deploy smart contracts. This network is "permissionless" and any participant can interact with it - directly or indirectly, through an API provider - in order to consult or send transactions over it. However, participant banks can only make transactions through the off-chain application after authentication, which relays their respective instructions to the networks. When a bank is created in the DCB Services system, two separate identities are provisioned on Corda and Ethereum for this purpose. The system operator knows the bank's on-chain and off-chain identities.

About Kovan

We used the Kovan testnet for the Ethereum part and Infura to access by REST API to this network. With this choice, the only costs for this part are the Ethereum gas price and the pricing of Infura. Our smart contracts are developed with the high-level language: Solidity.

About smart contracts

Assets are created according to technology modelling rules (i.e. states in the Corda ledger and entries in Ethereum smart contracts). Their behaviour is programmed in these contracts according to business rules. In order to provide the wholesale CBDC on the "guest" network, the central bank deploys an ERC-20 compatible cash contract that holds all the banks' cash holdings in the Ethereum network, while specifically whitelisting participating banks in order to control the distribution of wholesale CBDC.

Option contracts are created as ERC-721 compatible tokens that are non-fungible and whose lifecycle can be fully traced individually. They are traded as OTC instruments, so there is no need for a CSD representation in the Ethereum ledger. The issuer of an option sells it directly to the buyer through an atomic settlement process driven by an Ethereum DVP smart contract. When the owner exercises his option, the option contract will deliver the adequate wholesale CBDC amount, based on market prices.

4.3 Off-chain application

This application is composed of a backend based on the Spring Boot framework, which aims to authenticate the actors and to make the links with the other systems. In addition, a frontend based on the REACT framework, which brings together information from Corda by calling our REST API and from Ethereum using WEB3JS technology.

5. Interoperability between ledgers

One of the main objectives of this prototype was transferring the wholesale CBDC between the resident network (Corda) and the guest network (Ethereum). Prior to implementing the prototype, we considered several architectures. As this prototype had to be finalised during a 4-day hackathon, we opted for the trusted intermediary transfer solution, the simplest transfer mechanism, which works well, provides enough resilience and is described here below. Nevertheless, other solutions such as hash-locked trusted transfer or hash-locked trustless transfer were envisaged to increase the security and resilience of the transfer of CBDC between networks, but were not implemented.

5.1 Trusted intermediary transfer (implemented)

This transfer scheme is managed by the central bank issuing wholesale CBDC in both ledgers. A participant bank that wants to move funds from Corda to Ethereum, initiates a workflow where the funds are pledged to a transit state controlled by the central bank. Over time, the transit state[5] will accumulate the entire amount of wholesale CBDC on Corda that has been transferred to Ethereum and is therefore out of circulation in the Corda network. The central bank will then issue the same amount of wholesale CBDC on the Ethereum network and credit it to the bank's address, thus enabling it to spend it in the "guest" network.

When a CBDC holder on Ethereum wants to transfer funds back to the Corda network, it initiates a similar workflow that "burns" the funds in the smart contract. Then, the central bank triggers a transaction in Corda network that uses the locked funds in the Transit state to credit the right amount to a bank's state. This way, the transit account reflects the amount of CBDC that exists in the "guest" network at any time.

In its role of transfer operator, the central bank is also in charge of managing any incident that might occur during the transfer. If the creation on the destination network fails, then the central bank has to unwind the transaction. This means moving the funds back to the bank from the transit state. At any time, the central bank can verify that the amount of CBDC transferred from Corda to Ethereum equals the amount of CBDC in circulation on Ethereum. In addition, central banks can always verify who the CDBC holders on Ethereum are.

[5] The CBDC funds are in a "transit state" when transferred between Corda and Ethereum. This limitation could be improved with the model described in 5.2.

5.2 Hash-locked trusted transfer (not implemented)

This system has several advantages, since the participant bank has access to its funds at all times. It can either reclaim the transfer state in the originating system or claim the money in the receiving system. Therefore, it is resilient against any unexpected shutdown of the central bank node. There is no need for timeouts or any other synchronisation measures, since the process is completely asynchronous. The following graph and steps detail the model.

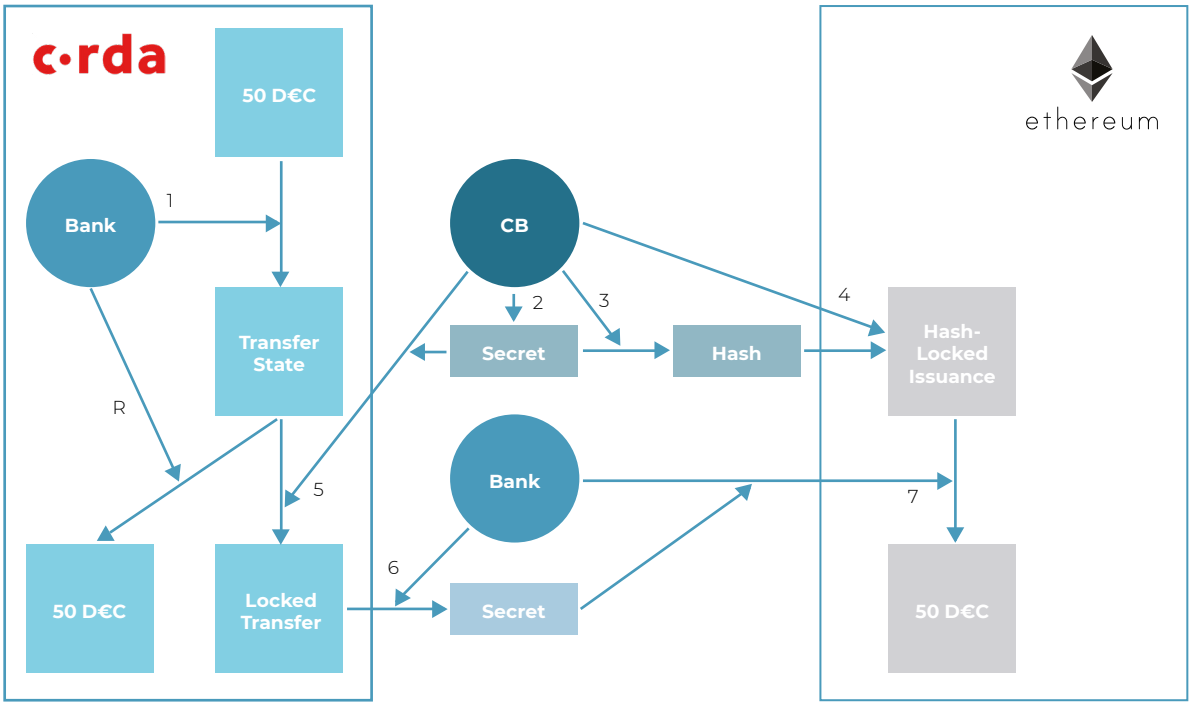


Figure 4 – Hash Locked Trusted Transfer model (D€C = CBDC in this graph)

(1) In Corda, the bank sends the money into an intermediate transfer state which has two functions. The main function can be called by the central bank by providing an additional argument. Calling this function locks the money and the bank cannot reclaim it anymore. The second function, called the (R) reclaim function, can be called by the bank, consuming the transfer state and returning the money to the bank. (2) After the central bank sees the transfer state, it generates a secret "S" and (3) calculates the hash of the secret: H(S). It then (4) issues the corresponding amount of money in the receiving system in a contract, attaching H(S) and with the condition that the money can only be claimed by providing "S". If the reclamation function of the transfer state has been used, the central bank can invalidate this hash-locked issuance. (5) The central bank then executes the locking function of the transfer state in the originating system, adding "S" as the additional argument. By consuming the transfer state, the reclaim function cannot be used anymore. (6) After seeing "S" in the locking transaction, the bank can then (7) claim the money in the receiving system.

5.3 Hash-locked trustless transfer (not implemented)

This is a different interoperability protocol for asset transfer that we evaluated during the hackathon. Instead of having the central bank as transfer operator, this model proposes a synchronisation method of two separate payments on each ledger, brokered by an intermediary party providing the necessary liquidity for this operation (similar to a PvP cross-border transfer featuring the same currency on both sides). Due to the use of two HTLC (hash time locked contracts), this operator cannot "steal" the funds, hence the trustless assumption of this architecture

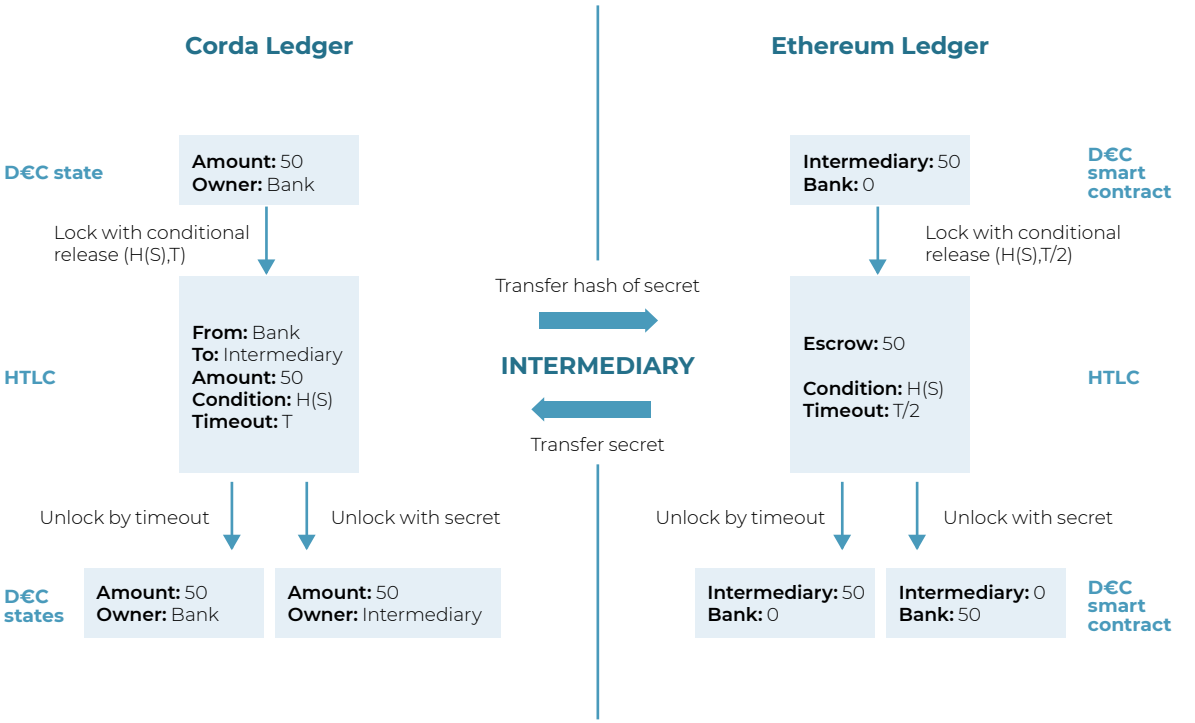


Figure 5 – Hash Locked Trustless Transfer model (D€C = CBDC in this graph)

The bank that needs to transfer CBDC from the Corda ledger to the Ethereum ledger locks the funds in a Corda state with a release condition that will either transfer the funds to the intermediary if a previously generated secret is provided or will return them to the bank after an inactivity timeout. Similarly, the intermediary party locks the same amount of CBDC tokens in an Ethereum smart contract that will conditionally transfer it to the bank if presented with the secret or return it to the intermediary after an inactivity timeout. As soon as the bank claims the funds in the Ethereum contract, the intermediary will use the divulged secret to retrieve the amount locked in the Corda transit state.

This model presents the following advantages: (i) there is no regulated trusted tier (just SLA coerced), i.e. the central bank has no operational role in the transfer, (ii) it is scalable architecture for multiple ledger CBDC distribution and (iii) ledgers can be operated under different regulatory regimes. Nevertheless, there are also drawbacks, as (i) cross-ledger liquidity management is required for the intermediary, (ii) central banks cannot easily monitor the total quantity of issued CBDC (to be calculated off-chain as there is no pivotal transit state).

6. Conclusions

The prototype consisted in (i) using Corda to deploy central bank services including a wholesale CBDC, (ii) deploying smart contracts in a public test version of Ethereum, (iii) building interoperability between these two networks and (iv) developing an off-chain application to manage the rest of the functions as well as the orchestration. We did not intend to compare specific blockchains, but rather gain experience with the different possibilities and benefits of this kind of infrastructure for central banks.

Considering the short timing offered by a hackathon to develop a prototype, we can conclude that all our objectives were reached with success:

- Corda is suitable for deploying basic central bank services and Ethereum offers several functionalities that are compatible with these services; (benefits will be further addressed in the next part, 7 – Findings and constraints);
- it is possible to create central bank financial services using private and/or public DLTs;
- interoperability can be achieved between private and public blockchains in a secured way; and
- portability of wholesale CBDC between Corda and Ethereum was performed. Libra's recent white paper[6] shows that the market is laying the groundwork for such possibilities in the retail CBDC context.

Next steps for the BCL

The BCL will continue collaborating with private companies in order to analyse how a production-grade system would need to deal with the constraints identified for a wholesale CBDC. The BCL will consider the use of both private and public blockchains for this purpose.

[6] Libra.org

7. Findings and constraints

About Corda

Corda is suitable to develop central bank services since Corda is well adapted to the financial industry requirements. Developing smart contracts for wholesale CBDC and securities is very straightforward. In terms of architecture, a Corda node can run on a JVM which is a common requirement in today's banking IT architecture; thus, it was not complicated to deploy. Corda also makes it possible to manage access rights and to define which nodes can access the information. For instance, it was interesting to split the wholesale CBDC and securities-related information between CSDs and central banks. The Corda (mandatory) non-validating notary node guarantees a proper ordering of transactions and prevents "double spending".

About Ethereum

As detailed in the dashboard above, there are several advantages to using the Ethereum public blockchain. The costs of public blockchain to execute a contract is rather low. While the Corda infrastructure costs are quite similar to a classical infrastructure (server costs, maintenance...), the costs are reduced when using Ethereum. We did not run any node or host any infrastructure or service; we only deployed smart contracts on a public Ethereum network. In Ethereum, we only had to bear the costs of deploying and invoking our smart contracts, when needed[7]. We also added a feature to allow central banks to follow the real-time costs[8]. Using the public Ethereum infrastructure may significantly reduce costs, and using public blockchains generally permits rapid deployment of services at a low cost of capital (no need to buy hardware, etc... running an application in the browser is possible to an extent). Nevertheless, the price of Ether[9] could be volatile and costs could fluctuate. The continuous availability of public blockchains is also interesting for 24/7 services and brings resilience to the services.

Finally, Ethereum's transparency offers possibilities for enhancing central banks' monitoring and reporting. In this prototype, when we create banks' static data, we save their public key(s) related to Ethereum. This information allows central banks to follow all usage of the wholesale CBDC by the banks on Ethereum and limit the circulation of CDBC. In the prototype, central banks authorise and deploy the option smart contracts; they can see all CBDC and options transactions, including the total exposures. Today, central banks do not readily have access to this transactional data, which is split between several intermediaries (like CCPs) and depends on the reporting of agents. The use of blockchains could include a harvesting of transactional data directly from the source and create opportunities for improving trade repositories monitoring for regulators[10]. There could be a joint benefit in this model with banks and other financial actors on the one hand benefiting from a safe CBDC token as settlement asset and, on the other hand, "automatic" reporting enhancing automatic cost-efficient monitoring for Central Banks.

---

[7] In the prototype, the Central bank supports all the costs. It is a feeless use of smart contracts for participating banks; nevertheless, other architecture choices could be explored.
[8] Deploying contracts, and running tests, costed around 1 Ether (around 150 euros in January 2020)
[9] Crypto-asset in which the usage of Ethereum is paid.
[10] In complement with the EU regulation n°648 /2012 (SFT-DS project).

Nevertheless, there are still some constraints to overcome. The anonymity has to be further developed to ensure that banks' usage of smart contracts in public blockchains remains only traceable by central banks. In addition, there should be a consensus on the usage of such infrastructures: the aforementioned benefits would only apply if the major stakeholders agree to use it. The last constraint is the number of networks to interconnect: how many blockchains would be accepted and which ones? The prototype was designed with two well-known blockchains, but there are many others. It could become very cumbersome to make several interconnections between different blockchains.

About performance

The speed of operations is deemed promising (see table below), bearing in mind that no major attempts were made to optimise the model for speed. The performance was observed from a user perspective on the graphical user interface.

PROCESS	PERFORMANCE OBSERVED	CONDITIONS
Settle internal payments[11]	0.3 sec	Corda test version
Settle liquidity transfers between corda and Ethereum	17 sec (Corda => Ethereum) 30 sec (Ethereum => Corda)	Corda & Ethereum test versions
Create equity option and settle the premium (ERC 721)	50 sec	Ethereum test version
Option premium payment (DVP)	30 sec	Ethereum test version
Exercise option (DVP)	15 sec	Ethereum test version

About scalability of this prototype

While central bank services are managed on a private Corda blockchain, several smart contracts have been deployed on a public version of Ethereum. It was rather straightforward to create this interaction between the two blockchains, which is why we can consider such a model as a scalable architecture for a multiple ledger central bank digital currency (CBDC) distribution. If scalability were significantly dependent on the scalability of the connected blockchains, any other blockchain could be in theory interconnected with DCB Services on Corda and benefit from the wholesale CBDC. Nevertheless, central banks would remain responsible for assessing the proposed public blockchain and reviewing the proposed smart contracts proposed by any financial actors that would use the CBDC as settlement asset and hold such a claim on the central bank.

[11] In a previous hackathon, we settled up to 1000 securities DVP transactions per minute (i.e. one transaction in less than 0.1 sec) on Corda.

About CBDC aspects to further investigate for production grade

Additional mechanisms should be developed to ensure the integrity of the issuance of the CBDC (no creation or loss of tokens by the guest chains) to preserve the trust and value of the currency. The cross-platform transfer is also a key point providing rollback mechanisms, and different models offering a higher level of safety than the trusted intermediary model (central bank control and maintains the contracts of issuance) or the hash-locked models (where trust is acquired through a digital signature from another platform) exist. In any case, central banks should ensure a controlled distribution according to their participation criteria (whitelist) and find a way to monitor the potential indirect use of CBDC by actors who want to use the tokens but are not entitled to directly hold a central bank account. Thus, a CBDC model should ensure visibility of the central bank on the balances of each CDBC holder, but not necessarily on the content of the underlining transactions. Finally, the CBDC should be transferable and available in different networks (technical agnosticity).

8. Glossary

Ethereum	Ethereum is a global, open-source platform for decentralized applications
Corda	Open-source blockchain platform for business.
Kovan Testnet	Kovan is a Proof-of-Authority (PoA) publicly accessible blockchain for Ethereum.
WEB3JS	WEB3JS is a collection of libraries, which allow you to interact with a local or remote Ethereum node, using a HTTP or IPC connection
WEB3J	WEB3J is a modular, reactive, type-safe Java and Android library for working with Smart Contracts.
RPC	Remote procedure call is a network programming model or inter-process communication technique that is used for point-to-point communications between software applications.
REST	Representational state transfer is a software architectural style that defines a set of constraints to be used for creating web services.
Infura	Development suite provides instant, scalable API access to the Ethereum and IPFS networks.
REACT JS	A component-based JavaScript library for building user interfaces.
Spring boot	Spring Boot makes it easy to create stand-alone, production-grade Spring-based Applications that you can «just run».
TIPS	TARGET instant payment settlement service of the Eurosystem
CBDC	Central Bank Digital Currency
DCB Services	Decentralised Central Bank Services
T2	TARGET2 – Real-time gross settlement payment system and central bank service of the Eurosystem
T2S	TARGET2 Securities – Securities settlement platform of the Eurosystem and European CSDs.
ECMS	Eurosystem collateral management module of the Eurosystem
API	Application programming interface
ERC20	It is the Ethereum token standard which is used for Ethereum smart contracts, ERC-20 defines a common list of rules that an Ethereum token has to implement
ERC721	It is the token standard for non-fungible tokens on the Ethereum blockchain

9. The Team

Banque centrale du Luxembourg
Participants and authors
Alexandre Briand
Simon Riffault
Yohann Tref
Nicolas Grandgirard
Under the supervision of Pierre Thissen

Neofacto
Frédéric Hubin
Thibault Introvigne
Under the supervision of Laurent Kratz

Banque de France
Victor Budau

Deutsche Bundesbank
Markus Zschocke

Lietuvos Bankas
Andrius Adamonis

This team developed the aforementioned prototype during the fourth EUROchain Hackathon which took place in Paris, hosted by the Banque de France Lab, from 28 to 31 January 2020. The project was pitched and designed before the hackathon by the Banque Centrale du Luxembourg with the support of Neofacto, and the rest of the team joined at the beginning of the hackathon.

International Relations and Communications Department
Communications Section
Tel.: (+352) 4774-4265 or 4599
Email: press@bcl.lu
www.bcl.lu