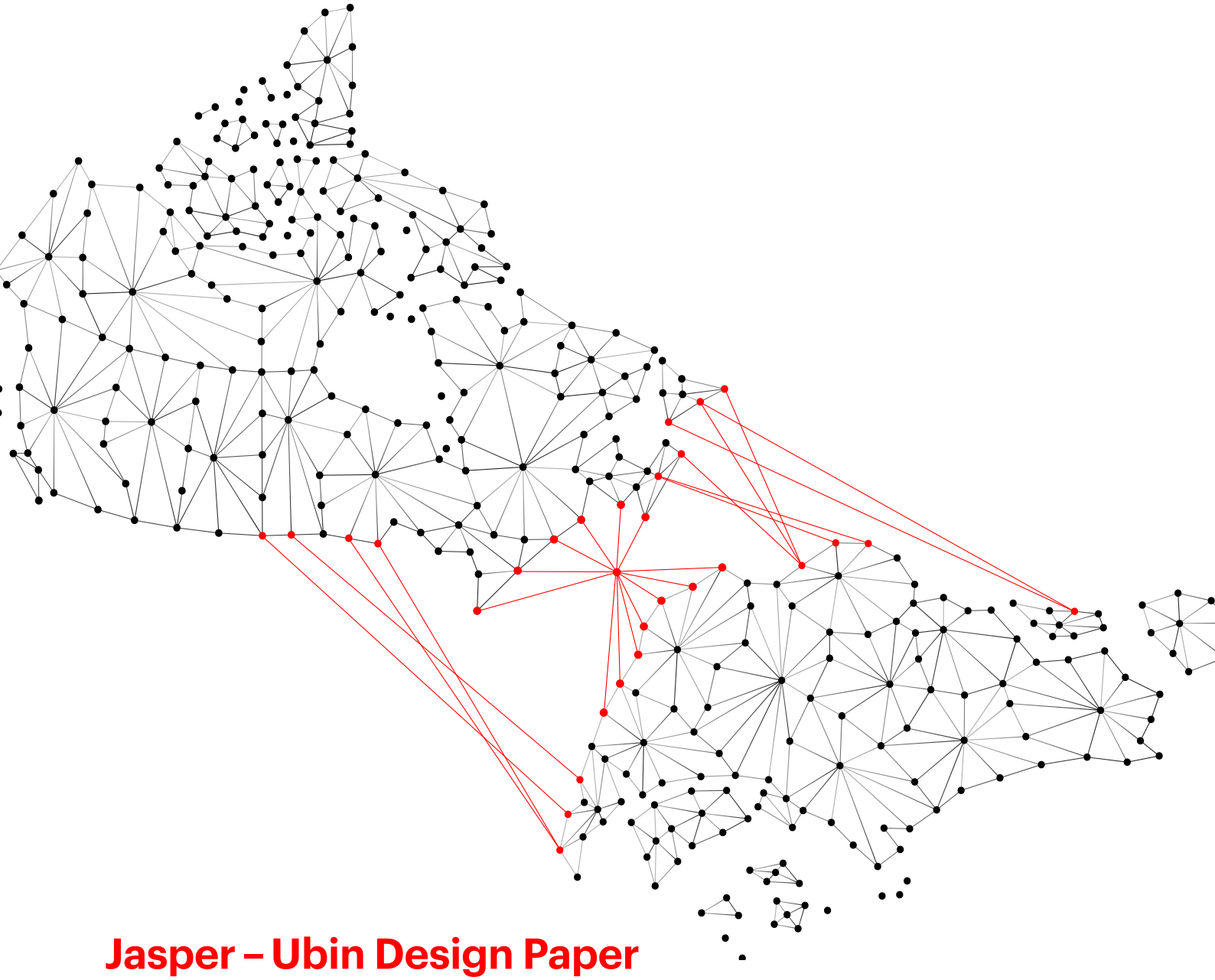
BANK OF CANADA
BANQUE DU CANADA

MAS Monetary Authority of Singapore



Jasper – Ubin Design Paper

Enabling Cross-Border High Value Transfer Using Distributed Ledger Technologies

Powered by: accenture J.P.Morgan

CONTENTS

FOREWORD

The Bank of Canada (BOC) and the Monetary Authority of Singapore (MAS) are pleased to present the report "Jasper-Ubin Design Paper: Enabling Cross-Border High Value Transfer Using Distributed Ledger Technologies".

In 2016, BOC and MAS embarked on Project Jasper and Project Ubin, respectively, to explore the use of Distributed Ledger Technology (DLT) for the clearing and settlement of payments and securities. This report describes how the Jasper and Ubin prototype networks, developed on different blockchain platforms, were able to interoperate, allowing for cross-border payments to be settled on central bank digital currencies, which in turn enables greater efficiencies and reduces risks.

The collaboration between the two central banks has successfully proven the ability for settlement of tokenized digital currencies across different blockchain platforms. In combination with earlier work on Delivery versus Payment (DvP) settlement, we are forging a path forward for blockchain platform interoperability in a future world of heterogeneous distributed ledger platforms.

A fragmented world, with differing standards, processes, norms, and regulations is the key challenge in cross-border payments today. DLT could offer an easier and faster path towards adoption than a centralized approach because it can leave the different jurisdictions involved in control of their portion of the network while allowing for tight integration with the rest of the network.

The Jasper-Ubin project is experimental in nature, and whether we will eventually use blockchain technology for high-value cross-border payments remains to be seen. Technology exploration and experimentation will continue because we see potential in this technology. More importantly, cross-jurisdictional collaborations must continue, as the development of common shared understanding will benefit the global ecosystem regardless of the technology that we eventually choose to use.

We would like to express our appreciation to JP Morgan and Accenture for their contribution to this pioneering work and endeavor.

We encourage central banks, regulators, financial institutions and technology companies to read about the achievements and learnings from the Jasper-Ubin project, and join our efforts in making cross-border payments cheaper, faster, and safer.

Scott Hendry
Senior Special Director,
Financial Technology, Bank of Canada

Sopnendu Mohanty
Chief FinTech Officer,
Monetary Authority of Singapore

EXECUTIVE SUMMARY

The Jasper-Ubin project sets out to determine whether, with recent technological innovations, it is possible to make safe cross-border payments and realize other benefits in a future world of heterogeneous distributed ledger platforms.

This work undertakes a line of enquiry emanating from the paper "Cross-Border Interbank Payments and Settlements,"[1] authored by the Bank of Canada, the Bank of England, the Monetary Authority of Singapore, HSBC and a group of other commercial banks in the United Kingdom, Canada and Singapore. That paper highlights a host of issues in current cross-border payment arrangements: lack of transparency of payment status, limited service availability, processing time, costs and operational risks. It proposes a small set of models that alleviate these issues.

Specifically, two models in that report describe a tokenized form of a wholesale central bank digital currency (W-CBDC) issued on blockchains by the central bank for use by commercial banks. We explore the architecture that supports these models by building a proof of concept to understand some of the technical challenges in implementing these models.

Cross-border payments generally involve a set of actions (updates to multiple separate systems) that are not tightly synchronized, creating the possibility that one action will succeed and another fail. This leaves the payment inconsistent, which essentially creates a risk that one party will gain at another's expense. This specific risk may be eliminated by ensuring all actions succeed or the transaction, in its entirety, is cancelled. One way to accomplish this is to employ a third party acting as escrow to the transacting parties; this third party will ensure commitment of the whole transaction. Another way is to provide a technology-based means of ensuring this commitment without a trusted third party.

The Monetary Authority of Singapore (MAS) and the Bank of Canada (BOC), together with JP Morgan and Accenture, embarked on the Jasper-Ubin Project, a technology-based experiment to realize this all-or-nothing guarantee through an atomic transaction for a Canadian Dollar (CAD) - Singapore Dollar (SGD) payment across two distributed ledger technology (DLT) platforms based on Hash Time-Locked Contracts (HTLC).

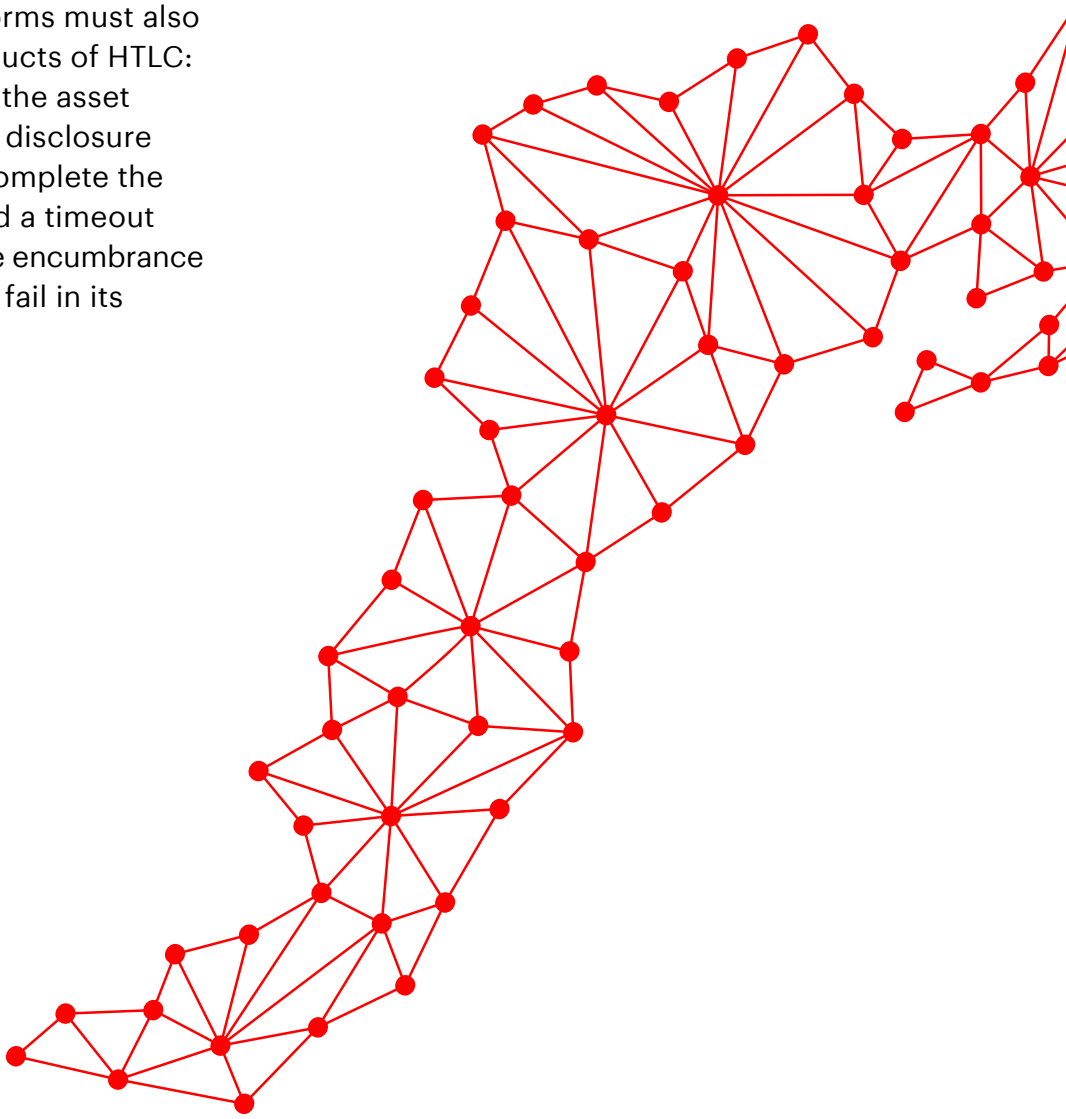
HTLC uses smart contracts[2] to synchronize all the actions making up a transaction so that either they all happen, or none happens.

Furthermore, the Jasper-Ubin project assumes DLT-based domestic gross settlement (RTGS) systems sit on different platforms in each country—in this case, on Corda[3] in Canada and Quorum[4] in Singapore.

The team successfully demonstrated a cross-border, cross-currency, cross-platform atomic transaction without the need for a third party that is trusted by both jurisdictions. In our tests, no other action would proceed if any action fails, thus ensuring the end to end consistency of a transaction. In the correspondent banking method of payment, the sender and receiver trust the correspondent bank. In this DLT-based system using HTLC, trust will still be required, albeit in the technical system rather than in a third party.

HTLC is a reliable way of passing messages between the two systems. Distributed ledger platforms must also support the basic constructs of HTLC: locking or encumbering the asset to be transferred, secret disclosure to the counterparty to complete the acceptance process, and a timeout mechanism to release the encumbrance should the counterparty fail in its acceptance process.

Our work does not constitute an entire solution for cross-border payments. There are open questions to be pursued: How would such a system behaves at scale? What are the complications that will arise with a large number of jurisdictions? How should such a system be governed, for example in updates to the protocol? Are there regulatory or legal aspects to be considered? Can HLTC be used to integrate with non-DLT based Real Time Gross Settlement (RTGS) systems? What problems, highlighted in the "Cross-Border Interbank Payments and Settlements" paper, are also solved using this method (e.g. transparency)?



INTRODUCTION

01

In 2016, BOC and MAS embarked on Project Jasper and Project Ubin, respectively, to explore the use of distributed ledger technology (DLT) for the clearing and settlement of payments and securities.

Both projects unilaterally aimed to develop more resilient, efficient and lower-cost alternatives to today's financial systems based on central bank–issued digital currencies.

Both projects successfully developed DLT-based prototypes for domestic interbank payments, and subsequent experiments have also explored and successfully tested the viability of simultaneous exchange and final settlement of tokenized digital currencies and securities assets. While these experiments have proven the viability of the technology, there are limited, incremental benefits of DLT in a domestic context where there is a trusted central party and where centralized systems perform efficiently.

With an intuition that DLT has merits, we opted to investigate cross-border payments with multiple parties transacting across different DLT networks in different jurisdictions, with no single trusted entity. DLT is hypothesized to be suitable for this use case because (a) traceability of ownership throughout the transaction across different networks is crucial; and (b) there is currently no single trusted central party across jurisdictions. In cross-border payments, central banks act as the trusted central party within their jurisdictions, however, there is no single organisation that can act as the trusted central party across the global payments network.

This technical study is a collaboration between BOC and MAS that uses the learnings from Project Jasper and Project Ubin to test the hypothesis that DLT will enable greater efficiencies and reduce risks arising from errors in coordinating processes across institutions in crossborder transactions. The project builds on earlier work previously undertaken by the Bank of Canada, the Monetary Authority of Singapore, the Bank of England, HSBC and a group of other commercial banks in the United Kingdom, Canada and Singapore to analyze the various business operating models for enabling more efficient crossborder high-value payments. Among the five possible models developed in that earlier work,[1] this project focuses on Model 3a (a W-CBDC that cannot be transmitted beyond its home jurisdiction), Model 3b (a W-CBDC that can be transmitted beyond its home jurisdiction), and variants of Model 3b, which are characterized by the linking up of different cash settlement networks, assumed to be built on different distributed ledger platform technologies.

The Jasper-Ubin technical project, supported by Accenture and JP Morgan, began with a design and analysis of the different possible models of connectivity between the two DLT networks—Quorum[4] and Corda[3]. Workshops were conducted with the Project Ubin consortium to discuss their design considerations and understand the implications of each model across technology, business and operations, and legal and regulatory policies. The team also successfully built a cross-border (Canada and Singapore), cross-currency (CAD and SGD), cross-platform (Corda and Quorum) system for atomic transactions based on HTLC.

This report captures the design considerations and discusses the technical aspects of implementing DLT for cross-border, cross-currency, cross-platform high-value payments. This report outlines the high-level architecture and technical design options and examines the use of Hashed Time-Locked Contracts (HTLC) to enable atomic transactions across DLT networks.

1.1 WHAT IS CROSS-BORDER PAYMENT?

A cross-border payment transaction is one where an entity wishes to send a payment to a recipient in a different jurisdiction. Typically, the sender and end receiver do not have access to the same ledger; hence, transactions between them take place through a series of linked transfers on different ledgers. A common example is where multiple correspondent banks are used in a series of intermediary transactions to reach the receiver.

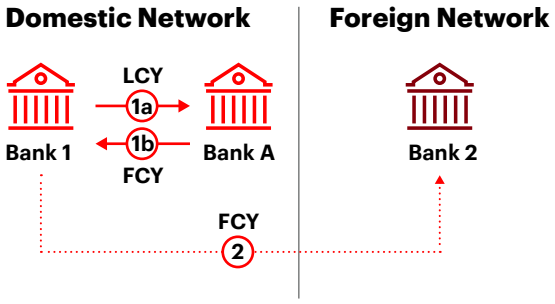
Cross-border payments often involve foreign exchange (FX) since the sender holds local currency (LCY), while the receiver would like to receive funds in their local currency, which is labelled as foreign currency (FCY) from the sender's perspective. The methods of obtaining FCY vary (e.g., the sender may already have FCY from previous transactions). Therefore, the FX funding aspect is distinct from the payment itself. For our experiment we have incorporated the FX funding aspect as part of the overall process.

In such a scenario, the transaction can be considered as two separate logical steps:

- Step 1 is an FX trade of LCY for FCY
- Step 2 is a transfer of the FCY to the receiver.
- Step 1 of an LCY-FCY exchange can be further broken down into 1a, a transfer of LCY from the sender to the FX trade counterparty, and 1b, a transfer of FCY from the FX trade counterparty to the sender.

These steps form the building blocks of a cross-border payment transaction and can be performed in different orders. In the example above, if an intermediary or correspondent bank performs the FX for the sender, there would be a transfer of LCY from sender to intermediary, and a transfer of FCY from intermediary to recipient.

Figure 1



1.2 WHAT ARE SETTLEMENT SYSTEMS?

On the most fundamental level, electronic settlement systems are accounting ledgers where the ownership of assets is recorded, and settlement is the process of updating the record of ownership of the assets being transferred. Payment, or a transfer of funds from sender to receiver, is "settled" by updating the ledger, decreasing the sender's balances and increasing the receiver's balances, whereby any obligations for that payment between the sender and receiver are diminished.

As such, direct transfers can only take place between parties with assets maintained on the same ledger. An example of ledger transfers is domestic interbank settlement systems, where participating institutions transact with each other on the central bank's ledgers. This is referred to as transacting on central bank liabilities, as the balances maintained with the central bank represent "deposits" that are repayable on demand. These balances are recorded as liabilities from the central bank's perspective. It is also possible for parties to transact on a private institution's ledger by maintaining accounts and assets with it.

DESIGN OPTIONS FOR CROSS-BORDER SETTLEMENT SYSTEMS

02

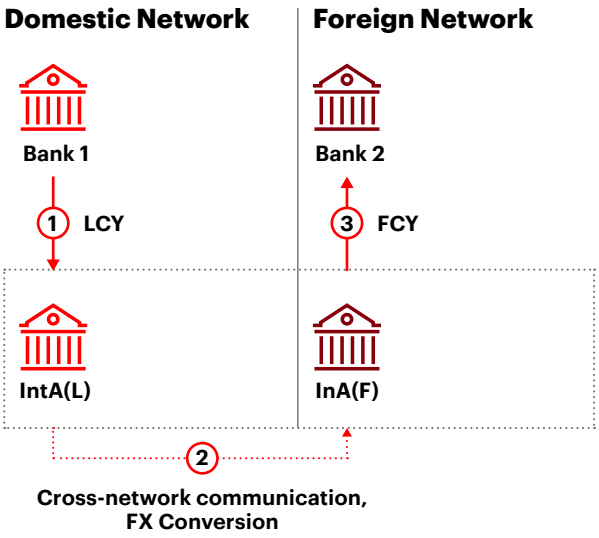
The Jasper–Ubin project builds upon prior technical study conducted by BOC and MAS. Its focus is on operating model underpinned by the interoperability of different DLT-based cash settlement networks, specifically from Project Jasper and Project Ubin. Although this is a technical study and not a policy research, this section outlines the business context for this project.

The models in Project Jasper and Project Ubin are limited by one particular design constraint: parties can transact directly only with other parties that are on the same ledger. In crossborder payments, where there are many transacting parties who do not exist on the same ledger, these parties would be able to transact electronically with each other only by either (a) using intermediaries, or; (b) granting direct access to a central bank's liabilities. Jasper-Ubin referred to these two broad design options.

2.1 USE OF INTERMEDIARIES

The use of intermediaries in the traditional correspondent banking model results in credit default risk (the risk that a party is unable to deliver the currency it sold) and settlement risk (the risk that a party delivers currency it sold without receiving currency it bought) for the transacting parties. One way of eliminating such risks is by removing the need to hold funds with the intermediary.

Figure 2: Cross-network communication



Using the same scenario described earlier, a sender will transfer LCY to an intermediary on the LCY network, and the intermediary will transfer FCY to the receiver on the FCY network. The intermediary facilitates the completion of the transaction without requiring transacting parties to hold funds with it. This differs from the traditional correspondent banking model where funds are held with the correspondent bank, which reduces credit risk exposure to the correspondent bank (or indeed vice versa where the sender receives credit from the correspondent for the payment).

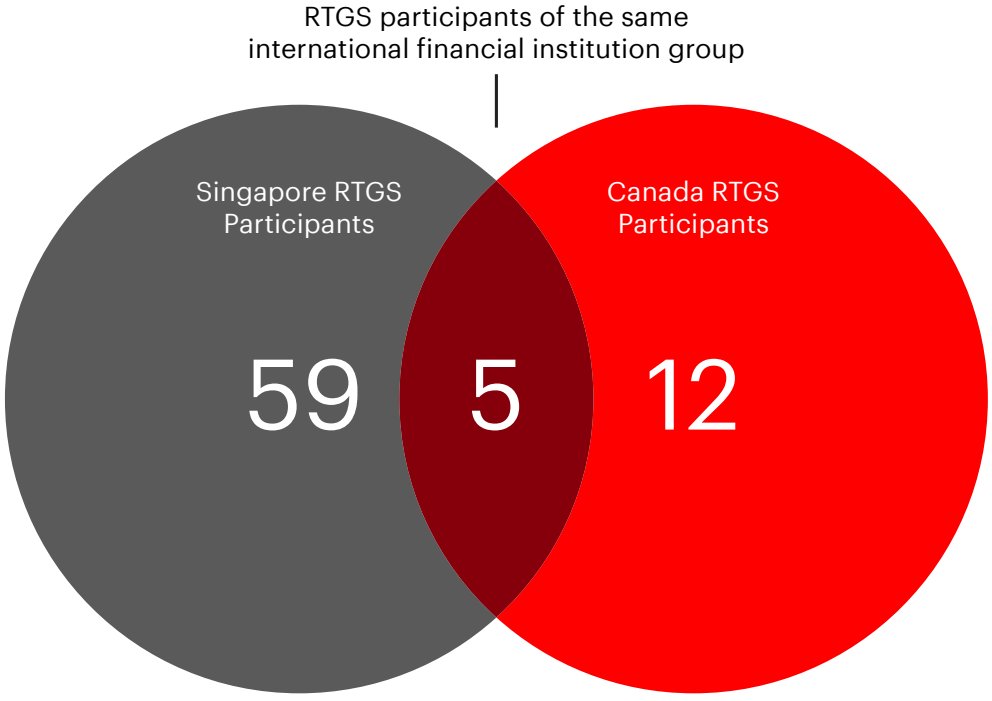
Settlement risk can also be eliminated by ensuring the atomicity of the related or linked transfers. Atomicity refers to the completion of all transfers comprising the transaction where they either succeed together or fail together. In the case of failure, the other linked transfers would automatically fail as well, reverting the funds back to the sender.

In the above example, there are two linked transactions, but there could be scenarios where atomicity must be ensured across multiple linked transfers.

Adopting this intermediaries model minimizes credit risk and settlement risk; in addition, as it is largely similar to the current correspondent banking model, it will be able to rely on existing regulations and processes.

Requiring the intermediaries to exist on both the LCY and FCY networks significantly reduces the number of financial institutions that can play the intermediary role. Based on an analysis of the 64 financial institutions that participate directly in Singapore's MAS Electronic Payment System (MEPS+) and the 17 that participate in Canada's Large Value Transfer System (LVTS), only five global financial institutions have a presence in both MEPS+ and LVTS.

Figure 3: Number of direct RTGS participants in Singapore and Canada



In the ideal case, the intermediary would be a global financial institution with a presence in both networks and thus bears no credit risk. Nonetheless, the intermediary could be a pair of separate financial institutions that are willing to bear the credit risk with respect to each other. In this case, there is still no credit risk for the transacting parties, but the intermediary bank would assume the credit risk of its counterparty intermediary bank. This could also increase the number of intermediaries that could facilitate cross-border transactions.

2.2 WIDENED ACCESS TO CENTRAL BANK LIABILITIES

The alternative to using intermediaries is to grant transacting parties direct access to central banks' liabilities. However, widening access to central banks' liabilities to non-regulated or foreign financial institutions raises a number of considerations and challenges.

There are open questions ranging from regulatory and legal challenges, to economic and monetary policy issues, to commercial costs and benefits for banks.

At the same time, research into this area has been limited because there have not been viable technical models that could enable a central bank's liabilities to be easily transacted beyond a limited group of regulated financial institutions. This report aims to develop technical models and posit their technical viability and, in doing so, create interest and research opportunities from the other perspectives.

PROPOSED TECHNICAL APPROACH FOR CROSS-BORDER PAYMENTS

03

3.1 ENABLING ATOMICITY OF TRANSACTIONS WITH HASHED TIME-LOCKED CONTRACTS

Since cross-border payments usually consist of a series of linked transfers, ensuring the atomicity of these transactions could minimize settlement risk. HTLC offers a possible technical solution to ensuring the atomicity of transactions across multiple DLT-based systems. In most computer systems, including databases, atomicity is guaranteed through the concept of "two-phase commit." A two-phase commit is a protocol that coordinates two or more processes that participate in a transaction to decide to commit or abort (roll back) all the processes of the transaction. The two-phase commit is typically implemented as follows:

Phase 1—Each participant in a transaction writes its data records to a temporary storage and indicates to the coordinator whether the process is successful.

Phase 2—Upon confirmation that all processes are successful, the coordinator sends a signal to all participants to commit, which is to update the records from the temporary storage into the actual storage. If any participant fails, the coordinator sends an instruction to all participants to abort and roll back.

Two-phase commit could work in the systems through the use of intermediary escrow accounts, which act similarly to the temporary storage. As an example, the FX exchange of LCY for FCY requires two separate transactions: (a) transfer of LCY from buyer to seller, and (b) transfer of FCY from seller to buyer. If an intermediate escrow is used, a buyer would first transfer the LCY to escrow and a seller would transfer the FCY to escrow. Once the escrow has ascertained that both legs of the transaction have been performed, it would complete the transaction by sending the LCY to the seller, and the FCY to the buyer. If one leg of the transaction fails, such as the seller failing to transfer the FCY to escrow, the escrow can roll back the transaction by refunding the LCY to the buyer.

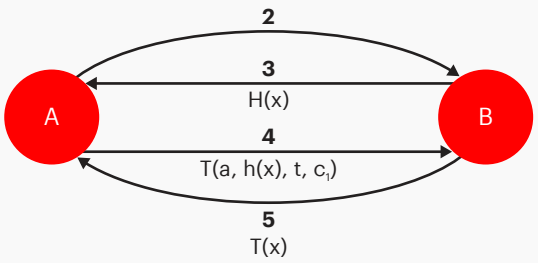
Achieving atomicity of transactions on conventional systems is not new. An entity such as an exchange can act as the trusted party by operating an account and allowing for delivery-versus-payment settlement only after both legs of a transaction have come in and temporary ownership of both cash and securities are assigned to the exchange. However, the problem becomes more complex when there is no single trusted party, as is the case with cross-border payments. This is where HTLC comes into the picture.

The HTLC design has been used in public blockchains to allow for asset swaps to take place across different blockchain networks. While similar in concept to a two-phase commit, HTLC has no need for a trusted third party. Rather, the intermediate escrow account is operated autonomously as a smart contract with predefined rules.

In the context of cross-border payments, where the transaction consists of two parts, one in a home country and one in a foreign country, the HTLC protocol may be used to manage both parts of the transaction. The recipient will generate a secret (denoted by S), and this will be converted into an encrypted output of a fixed length known as a hash (denoted by H(S)) to be included in the transaction. The recipient will need to verify the H(S) of a transaction from the sender within a predefined time frame, T; otherwise, the transaction will be voided.

Figure 4: Hashed Time-Locked Contracts, two-party explanation

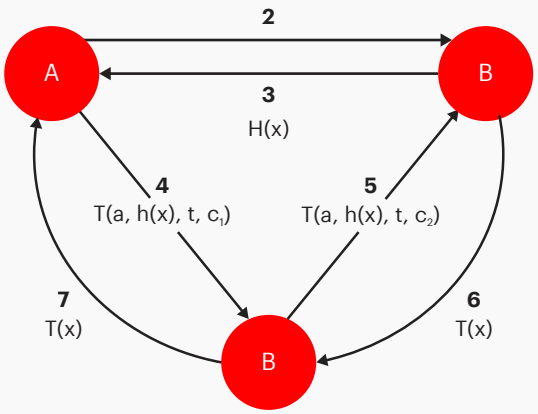
1. Alice opens a payment channel to Bob.
2. Alice wants to buy something from Bob for $1,000.
3. Bob generates a random number and generates its SHA256 hash. Bob gives that hash to Alice.
4. Alice uses her payment channel to pay $1,000, but she adds the hash Bob gave her to the payment along with an extra condition: in order for Bob to claim the payment, he has to provide the data that was used to produce that hash.
5. Bob has the original data that was used to produce the hash (called a pre-image), so Bob can use it to finalize his payment and fully receive the payment from Alice. By doing so, Bob necessarily makes the pre-image available to Alice.



→	On-ledger	a	Amount
→	Off-ledger	t	Time constraint
H(x)	Hash function of "x"	c_n	Currency
T( )	Transaction	u	Time taken to generate second transaction

Figure 5: Hashed-Time-Locked Contracts, multi-party explanation

1. Alice opens a payment channel to Charlie, and Charlie opens a payment channel to Bob.
2. Alice wants to buy something from Bob for $1,000.
3. Bob generates a random number and generates its SHA256 hash. Bob gives that hash to Alice.
4. Alice uses her payment channel to Charlie to pay him $1,000, but she adds the hash Bob gave her to the payment along with an extra condition: in order for Charlie to claim the payment, he has to provide the data that was used to produce that hash.
5. Charlie uses his payment channel to Bob to pay Bob $1,000 and Charlie adds a copy of the same condition that Alice put on the payment she gave Charlie.
6. Bob has the original data that was used to produce the hash (called a pre-image), so Bob can use it to finalize his payment from Charlie. By doing so, Bob necessarily makes the pre-image available to Charlie.
7. Charlie uses the pre-image to finalize his payment from Alice.



→	On-ledger	a	Amount
→	Off-ledger	t	Time constraint
H(x)	Hash function of "x"	c_n	Currency
T( )	Transaction	u	Time taken to generate second transaction

The execution of HTLC for each cross-border-payment approach will be illustrated in detail in the following sections.

3.2 PROPOSED TECHNICAL DESIGNS

Here we propose three broad conceptual design options for cross-border payments where a sender and a receiver are transacting on different ledgers with different currencies. The first option involves using intermediaries, and the second and third involve granting transacting parties access to the central bank's liabilities.

Access to the central bank's liabilities can be achieved through two different designs. The first design achieves direct access by granting transacting parties direct access to accounts or wallets on the network, i.e., allowing a financial institution to hold FCY issued by the central bank even if it is not a financial institution in that particular jurisdiction. The second design allows LCY to flow into foreign currency networks where it can be transacted directly. This can also be viewed as a multi-currency settlement system.

Figure 6 illustrates the three broad technical designs and Table 1 summarizes their characteristics.

Figure 6: Cross-border transaction approaches

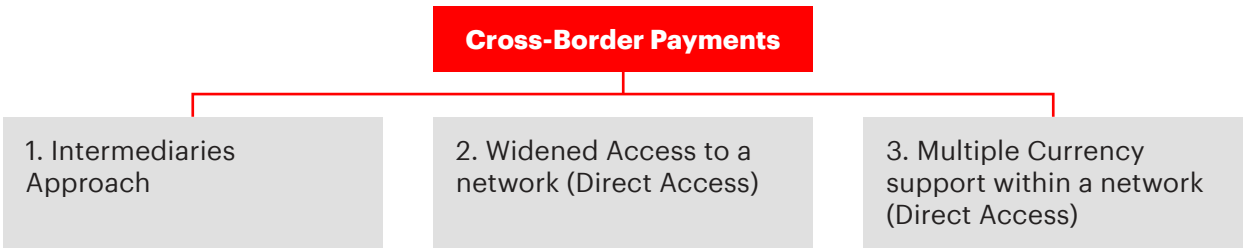


Table 1: Cross-border payments summary

Intermediaries Approach	Widened Access to a network	Multiple Currency Support within a network
• also known as asset swap via intermediary • needs intermediary for foreign exchange and transfer	• also known as direct access • does not involve an intermediary	• also known as asset transfer • allows for multiple currencies within the same network • still need intermediary (which could be the central banks) for transfer

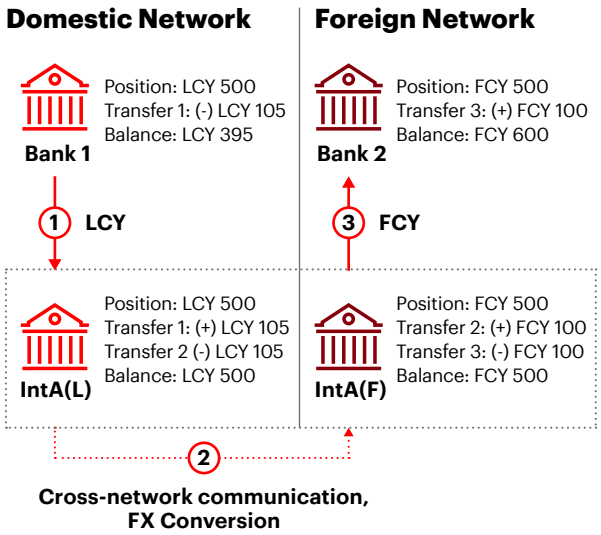
3.2.1. INTERMEDIARIES APPROACH

Conceptually, this method achieves cross-border payments by employing an intermediary to facilitate the settlement. The intermediary is a third party to the payment, acting as a middleman; the intermediary, typically a bank, would have access to both home and foreign networks. Having access to both networks enables the intermediary to receive money from the sender in LCY in its domestic network, and to send money to the receiver in FCY in the foreign network.

Because the intermediary facilitates the payments process, the sender would not need direct access to the foreign network, and, similarly, the receiver would not need direct access to the domestic network of the sender.

The FX conversion and transfer can be provided by the intermediary because each network can operate only its own currency. Therefore, the process of funding is integrated into the transfer process. Figure 7 illustrates this approach.

Figure 7: FX conversion and transfer via intermediary



In this example, Int A(L) and Int A(F) belong to the same international financial group, acting as intermediaries to complete the cross-border transfer. For this scenario we assume the exchange rate is 1.05 LCY to 1 FCY.

1. Bank 1 needs to make a payment of 100 FCY to Bank 2. Bank 1 will pledge 105 LCY to Int A(L).
2. Int A(L) converts the 105 LCY to 100 FCY using its entity in the foreign network, Int A(F). Bank 1 will be charged a transaction fee for this process.
3. Int A(F) transfers the 100 FCY to Bank 2 to complete the transaction.

HTLC sequence flow

An HTLC contract consists of two parts: hash verification and time expiration verification. A secret, denoted as S, will be created first, and then its hash will be generated, denoted as H(S). H(S) and S are key information used to ensure the atomicity of the linked transactions across the two blockchain networks.

The sequence diagram below provides a generalized illustration of how HTLC is executed for an asset swap via an intermediary. Note that HTLC may be implemented differently on different DLT platforms, depending on the capabilities and limitations of each platform.

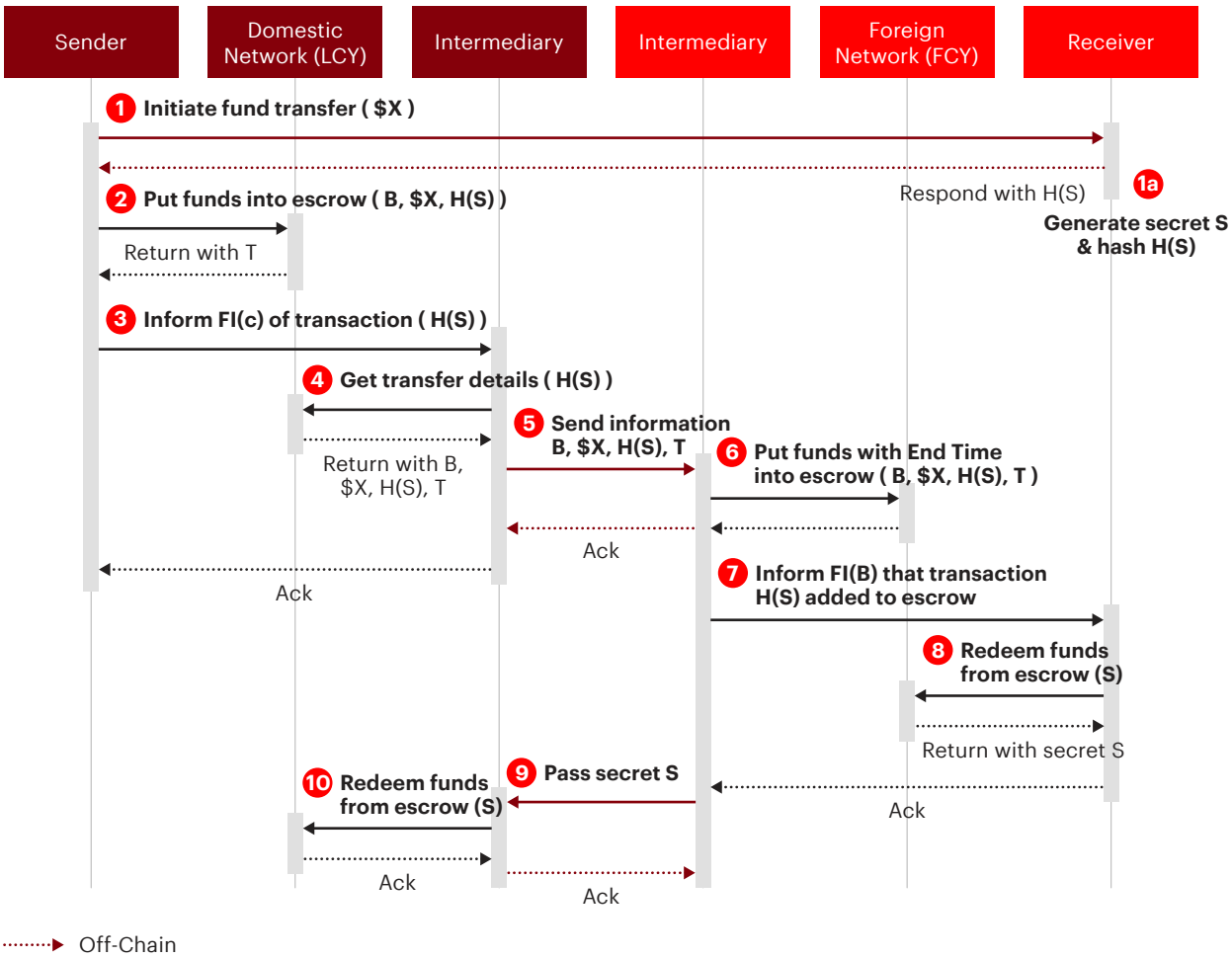
Smart contracts are self-executing computer programs that perform predefined tasks based on a predefined set of criteria or conditions. Smart contracts cannot be altered once deployed, which ensures the faithful completion of contractual terms.

The implementation of smart contracts varies with the platform in use:

In a **Quorum smart contract**, an asset or currency is transferred into a program. The program runs the code and at the same time validates a condition. It automatically determines whether the asset should go to a person or be refunded to the sender.

In a **Corda contract**, the executable code validates changes to state objects in transactions. The state objects are data held on the ledger that contains the information such as sender, receiver and the amount to be paid.

Figure 8: HTLC flow for swap via intermediary



1. The sender from the domestic network initiates a payment transfer request and requests an H(S) from the receiver in the foreign network. The receiver generates an H(S) and a secret for the transaction and acknowledges the sender with an H(S).
2. The sender creates a smart contract in the domestic network with the details below and puts funds in escrow.
   a. B = Receiver
   b. $X = Amount to be sent to receiver
   c. H(S) =Hash tied to the transaction

   The domestic network sets the time lock window, e.g., one hour or two hours for this transaction to be completed, otherwise this transaction expires and be rolled back.
3. The sender informs the intermediary in the domestic network about the transaction and sends the hash of the secret, H(S), to that intermediary.
4. The intermediary requests the smart contract on the transaction details and presents the hash in order to be authenticated as the valid party.
5. The intermediary in the domestic network sends the details on the receiver, amount and H(S) to the intermediary in the foreign network. This is a cross-network communication.
6. The intermediary in the foreign network also creates a smart contract in the foreign network based on the details received from the intermediary in the domestic network, i.e., receiver, amount, H(S) and time window. This time lock window for this smart contract will be half (T/2) of the overall transaction (as per Step 2 in this sequence). Here, the amount parameter may be different but should equal the same value that is received from the intermediary.

   The intermediary in the foreign network then adds funds to an escrow account and locks it. As soon as funds are added to the escrow account, an acknowledgment (Ack) is sent to the entire network.
7. The intermediary informs the receiver that the funds have been added to the escrow. The intermediary sends the hash of the secret to the receiver.
8. The receiver uses the hash to retrieve the correct secret from its repository. The receiver uses the secret to unlock the transaction and redeems funds from the smart contract account. Also, the receiver gives secret S to the intermediary in the foreign network.
9. The intermediary in the foreign network sends the secret S to the intermediary in the domestic network. This is a cross-network communication. It is important to maintain a secure and reliable cross-network communication channel so as to ensure that the intermediary in the home network is able to claim the funds (as described in Step 10 below) before the time period ends.
10. The intermediary in the home network presents the secret to the smart contract. The smart contract hashes the secret and compares it with the original hash. If they are same, the smart contract allows the intermediary to unlock the account and claim the funds.

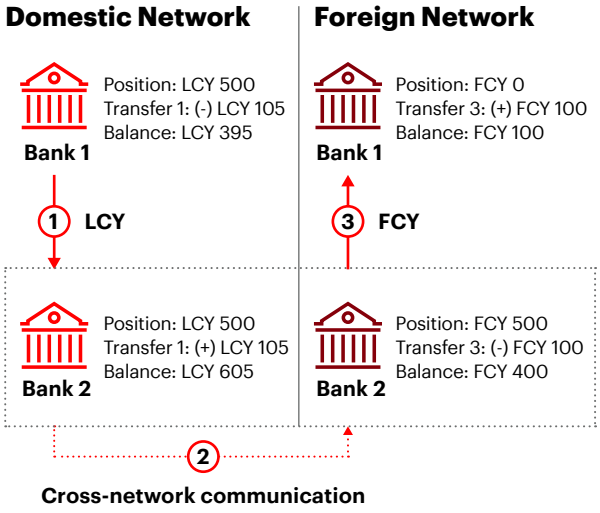
3.2.2 WIDENED ACCESS TO A NETWORK

In this approach, a bank would have access to both home and foreign networks and hold funds in each of these networks. This means that a sender bank would be able to hold a wallet in the foreign network, with FCY in that wallet, and a receiver bank would be able to hold a wallet in the domestic network, with LCY in that wallet. This would represent a change from existing policies where only a subset of domestically regulated financial institutions can gain direct access to the RTGS systems and central bank liabilities. It would require a widening of access policies.

Funding wallet in foreign network:

Figure 9 illustrates Bank 1 funding its FCY wallet via another market participant, Bank 2.

Figure 9: Asset Swap via market participant



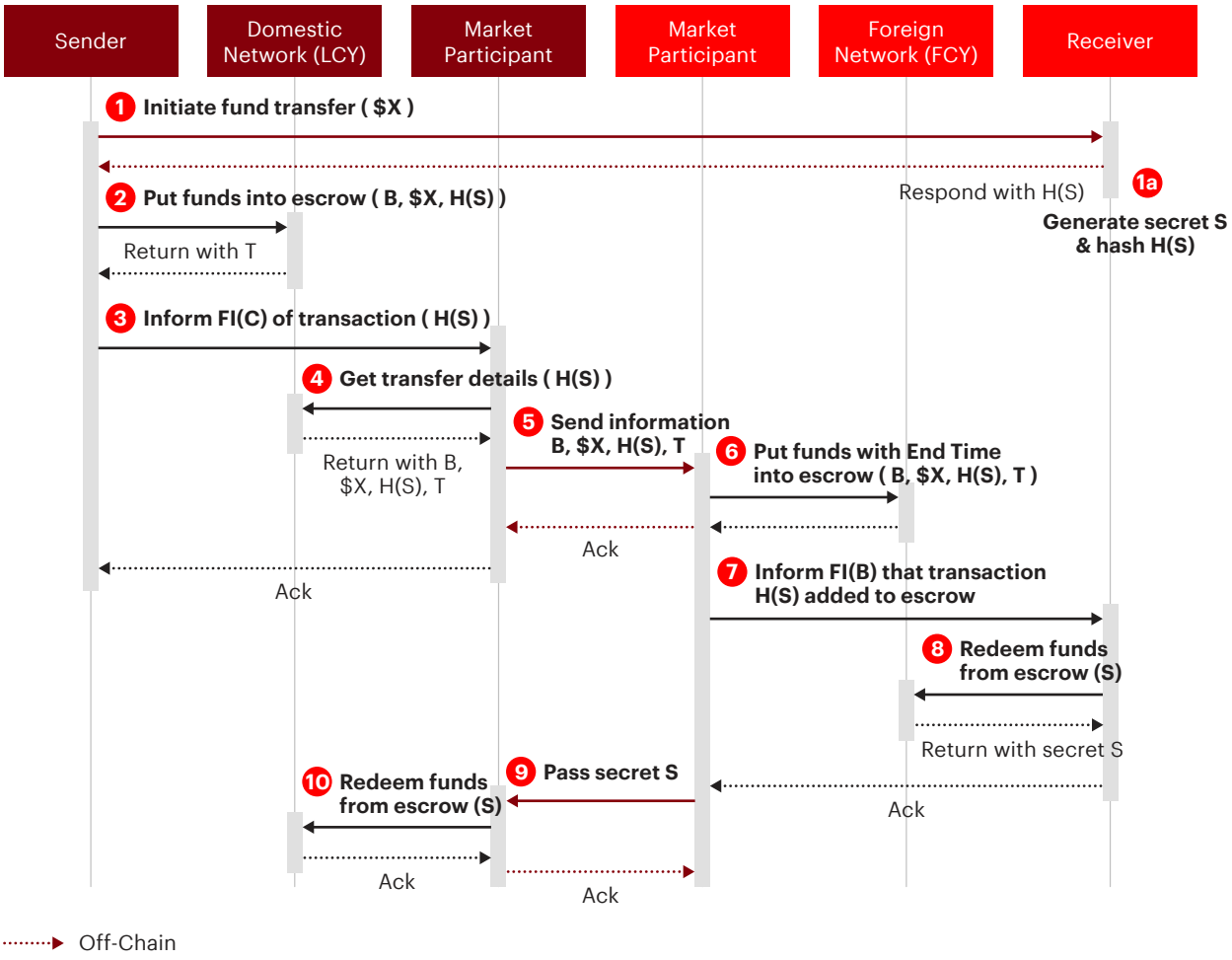
Bank 1 does not have sufficient funds in its FCY wallet. For this scenario we assume the exchange rate is 1.05 LCY to 1 FCY.

1. Bank 1 sends LCY to Bank 2's wallet in the domestic network. It is assumed the two banks have agreed in advance to an FX conversion outside the system.
2. Bank 2 internally manages its LCY and FCY wallets to increase the equivalent amount in its FCY wallet.
3. Bank 2 then transfers FCY to Bank 1's FCY wallet in the foreign network.

HTLC sequence flow

Figure 10 illustrates how HTLC is executed for the funding process of an asset swap. The detailed steps in this figure are similar to those in Figure 8, except that the intermediaries are now the market participants.

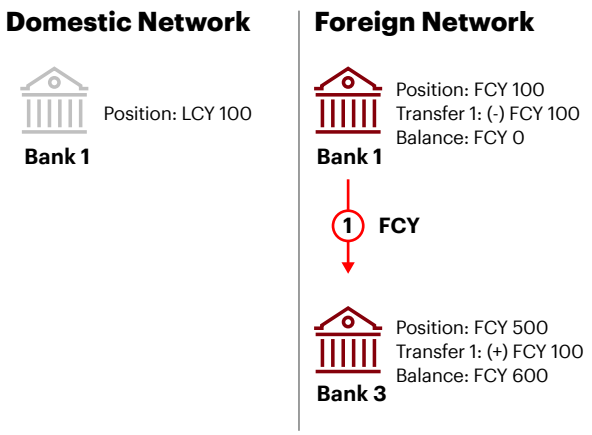
Figure 10: Asset swap through market participant sequence flow



Transfer:

Once Bank 1 has sufficient funds in its FCY wallet, it can make direct transfers to other participants in the foreign network in FCY. Figure 11 illustrates the process flow of a direct transfer from Bank 1 to a recipient bank (Bank 3) in a foreign network.

Figure 11: Direct transfer



1. With sufficient FCY after the initial funding, Bank 1 makes a transfer of 100 FCY to Bank 3 in the foreign network.
2. Bank 3 successfully receives the FCY, and the payment flow is complete.

3.2.3 MULTIPLE CURRENCY SUPPORT WITHIN A NETWORK

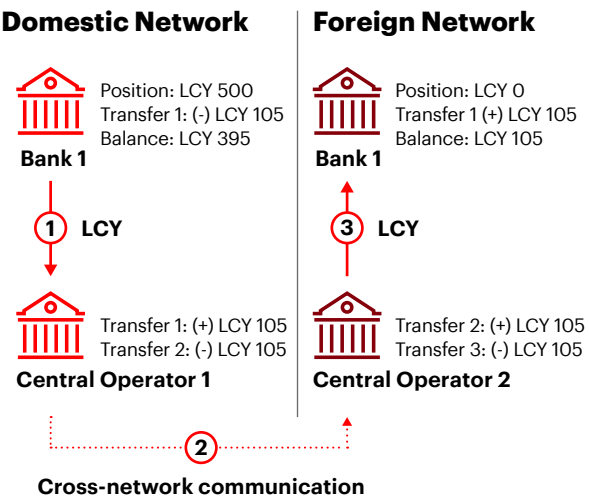
In the previous approach, money is sent from the sender in the domestic network in LCY to the receiver in the foreign network in FCY. The FX conversion and transfer are managed by the intermediary because each network can operate only in its own currency (leaving aside alternative funding arrangements).

This model assumes multiple currencies can be transacted in each network. For example, the sender bank will have both LCY and FCY wallets in its domestic network.

Pledging:

A sender can purchase new funds directly from the issuer. Figure 12 depicts pledging, the process flow for the initial issuance of LCY into an LCY wallet in the foreign network through the central banks.

Figure 12: Initial pledging flow



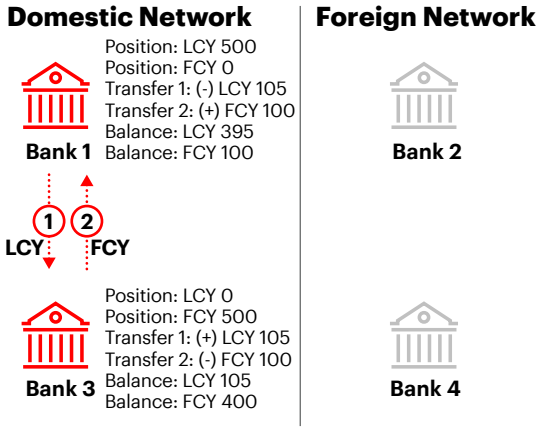
1. Bank 1 pledges 105 LCY to Central Operator 1 for transfer to the foreign network.
2. Central Operator 1 then informs Central Operator 2 that there is a request to transfer 105 LCY to Bank 1 in the foreign network.
3. Central Operator 2 sends the 105 LCY to Bank 1 in the foreign network, while Central Operator 1 redeems the 105 LCY on the domestic network.

Once multiple participants have repeated this process, all of them will have both LCY and FCY balances in each network, enabling direct transactions between them.

Funding:

A sender can purchase new funds directly from other participants. Once FCY is available in the domestic network, participants can transact with each other in FCY within their domestic network. Assume Bank 1 does not have a balance in its FCY wallet. In the funding process, Bank 1 exchanges LCY with other participants in return for FCY within the domestic network. Figure 13 illustrates this process.

Figure 13: Funding within home jurisdiction



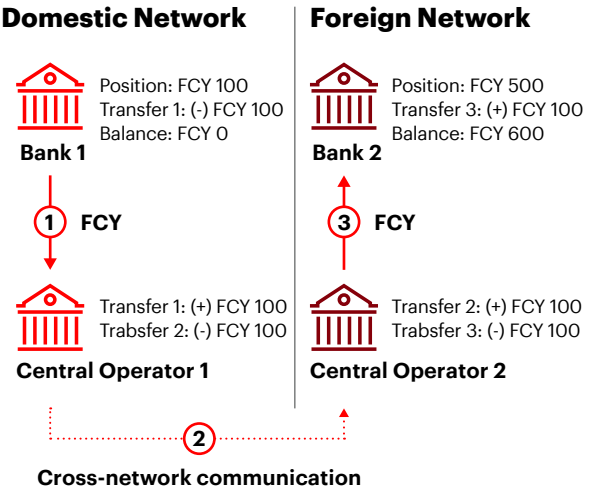
1. Bank 1 sells LCY to Bank 3.
2. Bank 3 transfers FCY to Bank 1.

This process is possible because Bank 3 has sufficient funds in its FCY account to complete this transaction.

Transfer:

Continuing from the previous example, Bank 1 now has 100 FCY in its domestic network wallet. Bank 1 can make a payment transfer to Bank 2 in the foreign network. Figure 14 illustrates the process flow of the payment.

Figure 14: Asset transfer



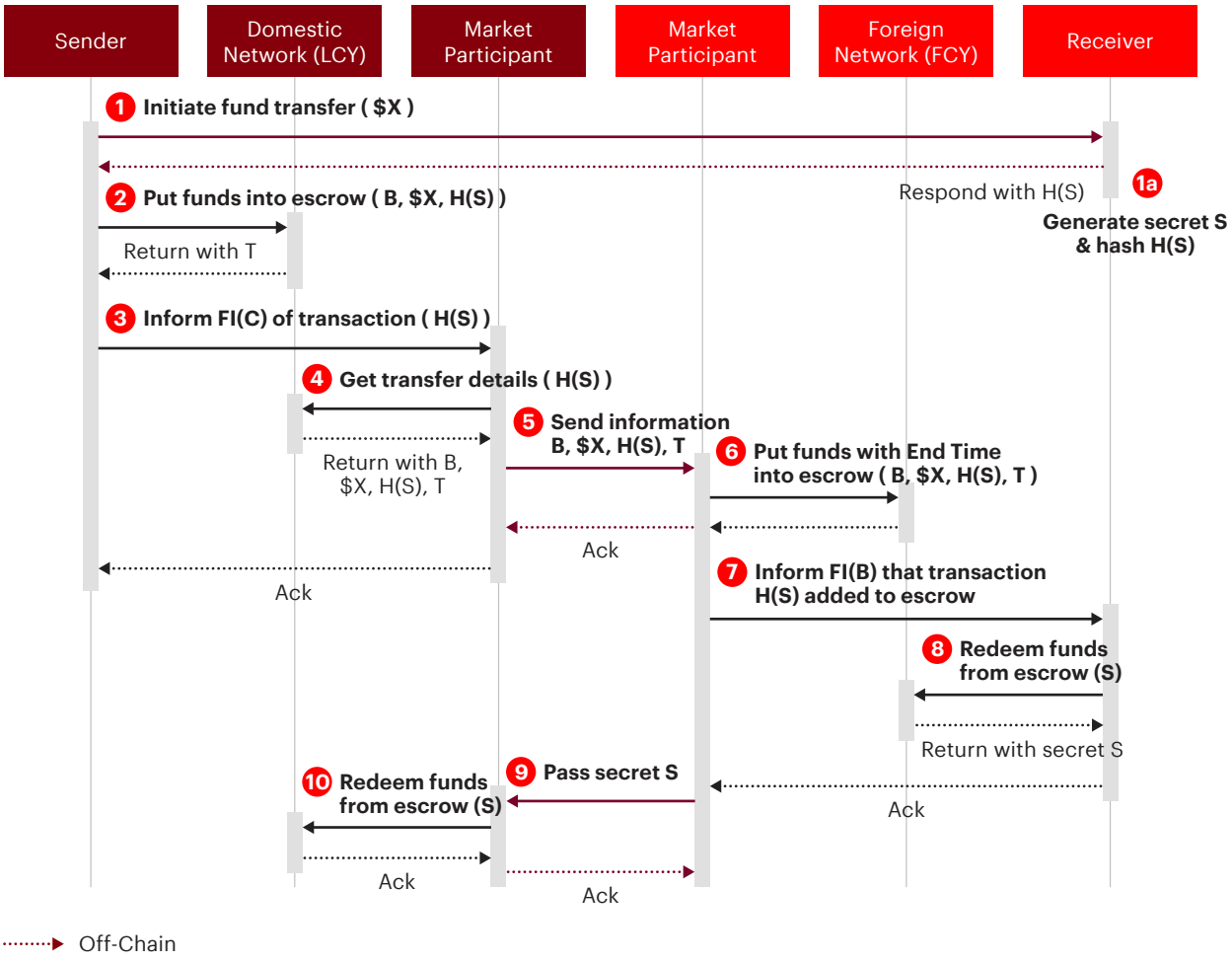
1. Bank 1 transfers 100 FCY to Central Operator 1 for transfer to the foreign network.
2. Central Operator 1 then informs Central Operator 2 that there is a request to transfer 100 FCY to Bank 2 in the foreign network.
3. Central Operator 2 sends the 100 FCY to Bank 2, while Central Operator 1 redeems the 100 FCY on the domestic network to ensure that no extra money is created.

Funds can be transferred between banks using the central operator as an intermediary. Note that commercial banks may also act as the intermediary, sending FCY in the foreign network in exchange for accepting FCY in the domestic network.

HTLC sequence flow

Figure 15 illustrates the HTLC sequence flow for transfer via central operators with the currency identification code included in the message payload.

Figure 15: Asset transfer



TECHNICAL PROOF OF CONCEPT

04

To verify the conceptual designs outlined in section 3.2, we developed a simplified proof of concept using Quorum and Corda. The proof of concept covers only one model—the intermediary approach (see Table 1), which is the least complex approach so as to focus on proving the technical viability of transacting atomically across two dissimilar blockchain networks using HTLC for atomic transactions.[5]

This technical proof of concept has two objectives:

- To implement HTLC contracts across Quorum and Corda DLT platforms
- To establish secure communication between Quorum and Corda to transfer transaction details, including secret hash H(S) and secret S

4.1 SET-UP FOR THE PROOF OF CONCEPT

In this section we illustrate the asset swap model between a bank in Singapore and a bank in Canada using intermediaries. We assume each of these jurisdictions has its own DLT-based payment networks, based on different DLT platforms, i.e., Quorum in Singapore and Corda in Canada. The atomicity of the cross-DLT transaction is achieved by implementing an HTLC protocol in both networks.

In the Singapore network, local Bank A and Intermediary A each uses two different Quorum nodes. In the Canada blockchain, Intermediary A and local Bank B in Canada will use two different Corda nodes. Intermediary A has a presence in both networks and acts as an intermediary. As part of this example, local Bank A in Singapore will transfer SGD$105 to local Bank B in Canada with the FX rate of 1 SGD to 0.95 CAD. At the end of the transaction, local Bank B in Canada will receive CAD$100.

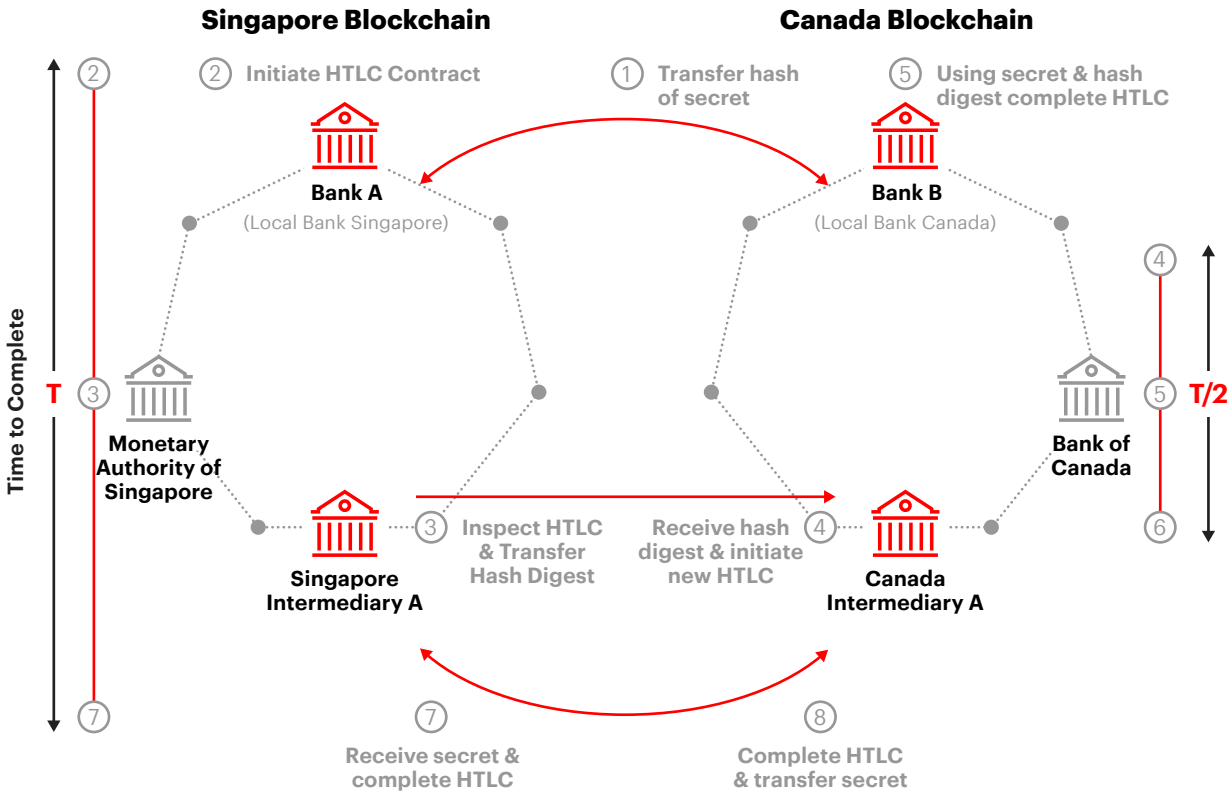
The set-up of the proof of concept is illustrated in Figure 16.

In the Singapore network:

Initiate HTLC transaction (with time to expiry T)

1. Bank A in Singapore and Bank B in Canada share the hash of secret H(S) off chain via a secure communication channel. Bank B generates secret S and creates H(S). Bank A uses H(S) to lock the contract.
2. Bank A initiates the HTLC transaction and completes the following actions as part of the HTLC initiation:
    i. Bank A locks the amount in the designated escrow account with the Intermediary A in Singapore as the recipient.

Figure 16: Overview of an HTLC



ii. The expiry time for the contract is set to T, which will be the overall duration for completing the payment processing across both DLT platforms.

iii. Since Intermediary A in Singapore is an intermediary bank in this contract, it receives the hash digest information.

Inspect HTLC

3. Using the hash digest H(S), which is part of HTLC, Intermediary A in Singapore can review the contents of the contract and validate that the appropriate amount is locked in the escrow account. As an intermediary bank in this payment process, Intermediary A in Singapore performs the following checks:

   i. Verifies that the amount of locked funds is correct. Locked funds can be claimed by Intermediary A in Singapore only after receiving the original secret from the corresponding DLT network.

   ii. Retrieves the contract expiry time T.

   iii. Sends the hash digest H(S) and contract expiry time (T/2) to Intermediary A in Canada.

In the Canada network:

Initiate HTLC (with time to expiry T/2)

4. Intermediary A in Canada receives the hash digest from Intermediary A in Singapore to start and lock the new contract in the Canada DLT network.

   i. Intermediary A in Canada starts a new HTLC contract with an expiry time of T/2 and the same hash digest H(S).

   ii. Intermediary A in Canada locks the amount in the escrow account with the recipient as Bank B.

iii. Since Bank B is the beneficiary in the above contract, it receives the hash digest information H(S).

Inspect and complete HTLC

5. Bank B, the beneficiary bank, inspects the contract on the Canada blockchain.
   i. Bank B verifies that the locked amount is correct.
   ii. Bank B completes the contract using the original secret to claim the funds from the escrow account and in the process releases the secret to Intermediary A in Canada.
6. Intermediary A in Canada shares the secret with Intermediary A in Singapore.

In the Singapore network

Complete HTLC

7. Intermediary A in Singapore receives the secret S and will be able to complete and redeem the locked funds from the escrow account.

The basic flow described above remains the same for the transactions initiated in the Canada network.

4.1.1 EXCEPTION SCENARIOS

A key aspect of the HTLC protocol is the off-chain transfer of secrets and hash digests between the participating and intermediary banks to facilitate the initiation and completion of transactions. As a result, the following exception scenarios may occur during the process:

1. Transfer of secret hash H(S) from Bank B in Canada to Bank A in Singapore—If Bank A loses H(S), it will not be able to initiate the transaction. Bank B will have to regenerate H(S) and send it to Bank A.
2. Loss of secret S and completion of the second leg of the transaction by Bank B in Canada—If Bank B loses the original secret S after sending H(S) to Bank A in Singapore or is unable to complete the second leg of the transaction in T/2 time, then the transaction will expire in both networks, and the funds will eventually be returned to Bank A.
3. Transfer of secret hash H(S) from Intermediary A in Singapore to Intermediary A in Canada—If Intermediary A in Singapore is unable to send H(S) to Intermediary A in Canada, then no transaction will be initiated in the Canada network. As a result, the transaction in the Singapore network will expire after T time, and the funds will automatically be returned to Bank A.
4. Transfer of secret S from Intermediary A in Canada to Intermediary A in Singapore—If Intermediary A in Singapore is unable to receive the original secret S from Intermediary A in Canada, then the transaction in the Singapore network will expire after T time, and funds will automatically be returned to Bank A. In this scenario, the intermediary bank loses the funds since it has paid Bank B in Canada but has not received the funds from Bank A in Singapore. This can be prevented by ensuring a reliable communication channel and/or a different rollback mechanism as discussed in the section "Advantages and limitations of HTLC".

4.2 SINGAPORE NETWORK DESIGN

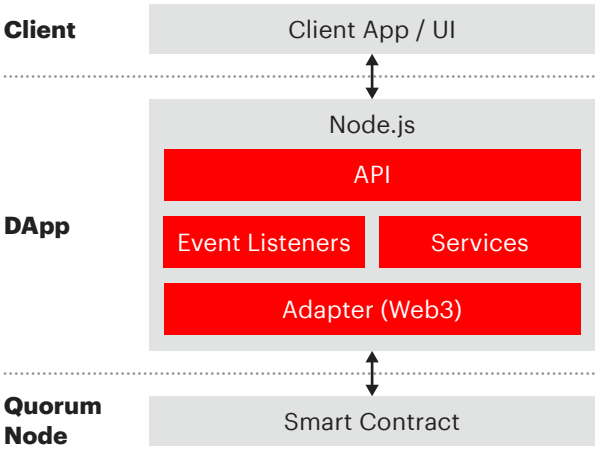
The Singapore network was built using Quorum, which is a blockchain platform developed by JP Morgan for the financial services sector. For this technical proof of concept, we used Solidity as the programming language for smart contracts and Node.js for the application layer. We used React for the user interface layer. Refer to the Appendix (Quorum Framework) for a detailed technical description of the Quorum platform.

This section provides an architecture and design overview of the Singapore network solution in the proof of concept.

4.2.1 ARCHITECTURE

Figure 17 depicts the logical architecture of the prototype. Details of each layer are explained in the subsequent sections.

Figure 17: Quorum logical Architecture



A sample user interface using React JS to demonstrate the actions taken by various participants during the life cycle of an HTLC contract was created. The client application communicates all user actions to the decentralized application (DApp) of the DLT network via HTTP REST calls and displays the up-to-date balances, transaction history and active contract details on the respective networks.

The DApp functions as the orchestrator of the payment process and acts as a bridge between the public and private contracts. It contains all orchestration flow and logic. The DApp accepts requests from a client through a RESTful application programming interface (API) and invokes the appropriate sequence of smart contract calls. The DApp utilizes the Web3 library to interact with smart contracts via JSON-RPC through HTTP and listens to contract events, which will subsequently trigger calls to the appropriate services. The DApp also calls an off-chain binary for the transfer of the secret and hash digest.

Following functional APIs were used:

- To set up a new bank in the network
- To add a bank balance
- To reduce a bank balance
- To get the current balance of the bank
- To get all HTLC transactions for a bank
- To initiate an HTLC transaction
- To reclaim the transfer amount if the transaction fails or expires
- To initiate an HTLC transaction on another DLT
- To complete an HTLC transaction

Cross-chain functional APIs used

The following information is used to send an encrypted secret hash to a bank on a corresponding network for redemption:

- The bank identifier code (BIC) code of the sending intermediary bank
- The BIC code of the receiving intermediary bank
- The BIC code of the contract originating bank
- The BIC code of the contract beneficiary bank
- A unique identifier for the transaction. This is created during first-leg contract initiation.
- Encrypted value of the secret hash used to initiate the contract

To send an encrypted secret to a bank on the corresponding network to initiate the second leg of the contract, the following information is used:

- The BIC code of the sending intermediary bank
- The BIC code of the receiving intermediary bank
- The BIC code of the contract originating bank
- The BIC code of the contract beneficiary bank
- A unique identifier for the transaction. This is created during first-leg contract initiation.
- Timeout used for first-leg contract (in seconds)
- Amount to be transferred (in the currency of the network of origin)
- The encrypted secret used to initiate the contract

The DApp uses events emitted by the smart contract in two ways:

1. Returning values when performing call transactions—The DApp will invoke smart contract methods by sending transactions and will receive responses through events that are emitted in the form of returning promise values.
2. Listening to events that are emitted as a result of actions taken by other parties on the network. A bank receives notification that another party has acted by listening to events that get emitted. For example, when a sender submits a fund transfer, an event is emitted that notifies the receiver.

The DApp is stateless and relies on the data stored in smart contracts to carry out operations.

The DApp's services layer contains functions that call the smart contract methods when an action is invoked using the API. The event listeners also respond to the emitted events by calling functions in services.

Services are broken down into:

- bank-related functions, such as pledge, redeem balances, etc., and
- HTLC-related functions.

For this HTLC proof of concept, we used the following private smart contracts:

- **HTLC**—This represents the core HTLC contract that is created between the initiating and counterparty banks for every cross-chain transfer function. Each HTLC contract owns an escrow contract in which the transferred funds are locked using the secret digest. This contract also tracks the HTLC expiration time based on the network timeout value.

- **Stash**—This is the store contract that holds the individual balances for each bank in the network. All debit and credit of funds is performed by this contract.
- **Escrow**—The escrow account plays a critical role in HTLC implementation. This is an extension of the stash contract that keeps track of the locked balance held between the initiating and counterparty banks.

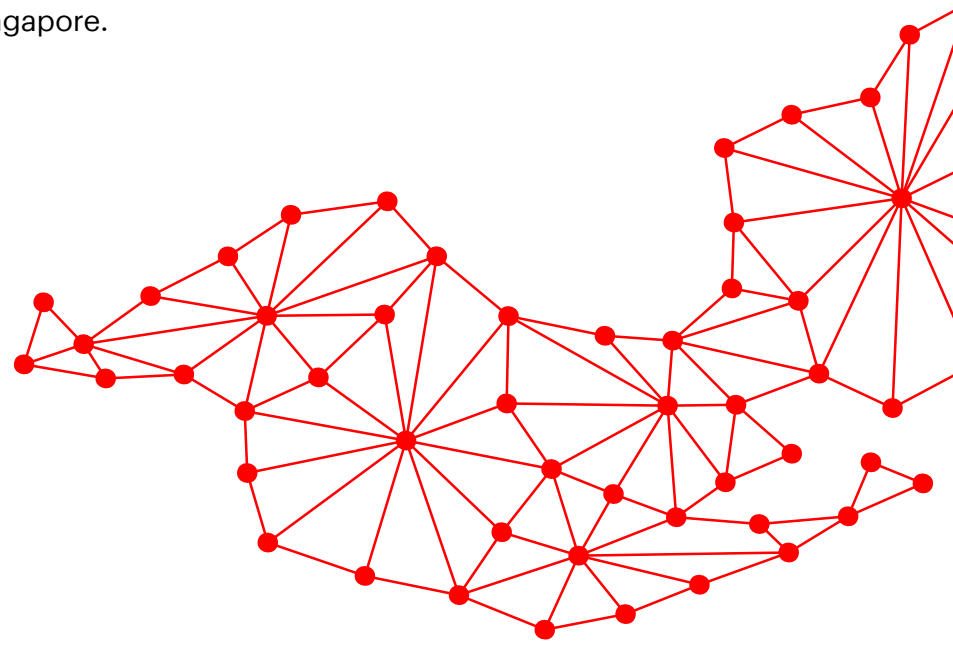
The process of using the escrow account is described below:

- Bank A locks the funds in an escrow account with Intermediary A in Singapore as the counterparty.
- Intermediary A in Singapore claims the money only after receiving the secret from Intermediary A in Canada.
- While Intermediary A in Singapore is claiming money from the escrow account, a set of validations (a predefined set of rules) is performed to ensure that the correct counterparty is claiming the money. Once all the validations have successfully been completed, the amount will move from the escrow account to Intermediary A in Singapore.

Below are a few predefined rules used in the proof of concept:

1. The requester of the funds and the counterparty should be same.
2. The HTLC identifier must match between the fund requester and the escrow account.
3. The HTLC completion or timeout criteria must be met.

Similarly, for the timeout scenario, funds are transferred from the escrow account to Bank A automatically once the validation succeeds.



4.3 CANADA NETWORK DESIGN

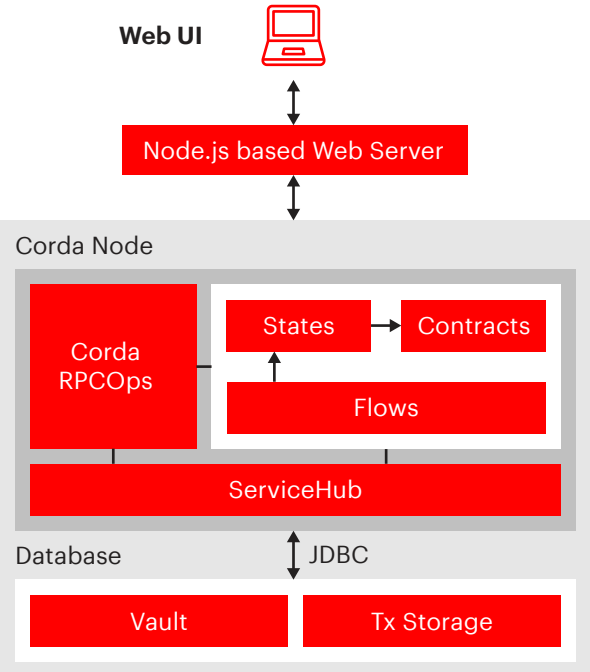
The Canada network was built using open source Corda version 3.2. Corda is a DLT platform from R3 that is designed for use with regulated financial institutions. Refer to Appendix B (Corda concepts) for a detailed technical description of the Corda platform. This section of the report provides an architecture and design overview of the Canada network solution used in the proof of concept.

4.3.1 ARCHITECTURE

For the proof of concept, Corda nodes (Bank, Bank of Canada, notary, etc.) and other components are hosted in Azure virtual machines. Figure 18 shows the architecture of the Corda-based DLT solution used to set up the Canada network and process HTLC transactions.

A sample web-based user interface (UI), similar to the one used in the Singapore network was created using React JS to run use cases for the proof of concept

Figure 18: Corda node



(mainly the issuance of CAD W-CBDC and cross-border transactions using HTLC). The UI also demonstrate the actions taken by various participants during the life cycle of an HTLC contract. The UI, also known as the client application, communicates all user actions to the Corda distributed applicationn (CorDapp) via RESTFul calls and displays the up-to-date balances and transaction details.

In the Corda-based DLT network, each participant runs a node with business-specific functionality implemented in CorDapp to provide peer-to-peer private transactions. CorDapp (consisting of states, contracts, transactions and flows) was developed using the Corda platform to support various use cases. These are the key nodes in the networks:

- Bank of Canada node issues tokenized CAD W-CBDC.
- Bank nodes participate in the cross-border payment transactions.
- Notary node provides uniqueness consensus on the transactions.
- Escrow node is a trusted entity introduced in the proof of concept to hold the funds in escrow to facilitate cross-border transactions using HTLC.

The tokenization of cash (W-CBDC) follows a digital depository receipt (DDR) model similar to the one implemented in the previous Jasper projects. A participating bank can obtain CAD W-CBDC tokens from the Bank of Canada by pledging cash from its existing account at the bank (off-DLT ledger). The Bank of Canada issues CAD W-CBDC tokens for the given amount and transfers the same amount from the requester's account to a pool account. Similarly, a participating bank can redeem CAD W-CBDC tokens it owns at the Bank of Canada in exchange for receiving the underlying cash in its account, transferred from the pool account.

For the cross-border (interledger) payment transactions, we considered various architecture options (using Composite Keys, Encumbrance states etc.) to successfully build HTLC functionality in a Corda network. Currently, the Corda platform does not fully support the features (locking, secret disclosure, timeout) required to properly implement HTLCs without introducing unacceptable failure modes and a trust model. The development effort that would have been required at a platform level to achieve this was not possible within the time available to the team.

For this reason, we introduced a trusted entity, the escrow node, which is part of the transactions, to implement the HTLC functionality. The escrow node is assumed to be trusted and will be owned/maintained by the network operator, similar to the trust framework for the notary. Around the escrow node, the flows and signatures are carefully managed to ensure that the transaction cannot be sabotaged by any party. As part of the HTLC protocol, the escrow node provides hash validation and time validation.

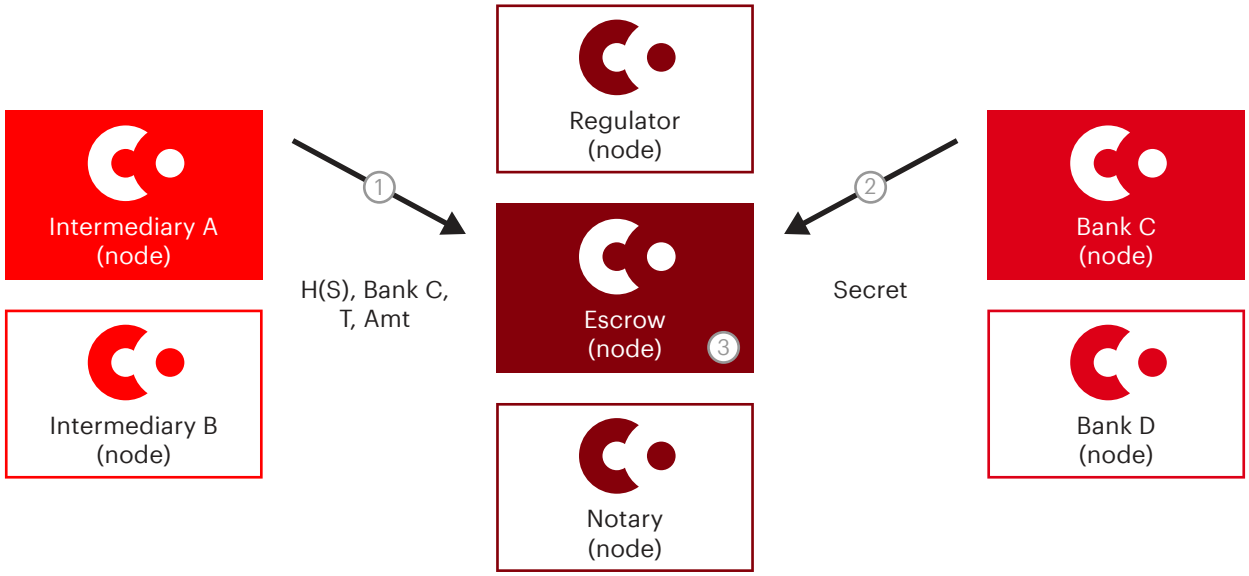
The hash validation process using the escrow node is described below:

1. Intermediary A locks the funds with the escrow node by providing the hash of the secret.
2. Bank C presents the secret to the escrow node.
3. The escrow node hashes the secret and compares it with the hash provided by Intermediary A. If it matches, the escrow node releases the funds to Bank C.

For the time validation, the escrow node ensures that Bank C claims the funds (by providing the secret) within the time window. Otherwise, it automatically returns the funds to the intermediary. The time lock element of HTLC is implemented by suspendable Corda flows with timeouts set to throw an exception and reject the transaction for the originator to reclaim the funds.

The HTLC protocol relies on the ability of the nodes in two different networks to pass secure reliable messages (secret, hash code, timeout, amount, etc.). This functionality was implemented in the network using custom RESTFul APIs to communicate with the Singapore network.

Figure 19: Corda logical architecture



DISCUSSION

05

The project successfully implemented and demonstrated the ability to perform atomic transactions between a Quorum-based network in Singapore and a Corda-based network in Canada using HTLC. This was proven by the successful transfer of SGD$105 between local Bank A in Singapore and local Bank B in Canada with the FX rate of 1 SGD to 0.95 CAD. At the end of the transaction, local Bank B received CAD$100. Various failure scenarios were also successfully tested. For example, local Bank A in Singapore was able to get SGD $105 back when the Canadian bank did not claim the corresponding $100 before the deadline set by the HTLC timeout period. We also analysed the design for various failure modes, e.g. the impact of the failure of different nodes at specific points in a transaction execution. We found HTLC to be robust in handling these scenarios.

While the specific proof of concept of cross-border payments between two different DLT networks was proven, the limited scope of the Jasper-Ubin project means that many opportunities for in-depth research remain open. The key findings, challenges and limitations are listed in this section to encourage further exploration by the DLT community.

5.1 DLT PLATFOM SUPPORT FOR HASHED TIME-LOCKED CONTRACTS (HTLC)

HTLC protocol uses hash locks and time locks to ensure the atomicity of transactions across two DLT platforms. The receiver of the payment either acknowledges receiving the payment prior to a deadline (timeout) by generating cryptographic proof of payment (hash lock) or forfeits the ability to claim the payment, which results in the payment being returned to the payer. A DLT platform must support locking, secret disclosure and timeout to successfully build the HTLC functionality. A DLT platform lacking these features may find it difficult or impossible to implement HTLC. However, there are no standards to govern how HTLC is implemented on each of the platforms; therefore, HTLC implementation may differ from one platform to another. In addition, reliable communication channels, such as redundant circuits, are needed among the networks for the transfer of the secret, contract details, etc.

5.2 HTLC ACROSS MULTIPLE NETWORKS

HTLC could theoretically be used for atomic transactions across three or more networks, but this was not tested in the current proof of concept, which tested only for atomic transactions across two DLT networks. There are possible use cases for such transactions, such as a foreign currency transaction with a bridge currency, i.e., SGD to USD to CAD, or in delivery-versus-payment-versus-payment transactions where investors convert their LCY to FCY to purchase securities denominated in FCY. Further experimentation on HTLC across three or more networks would be required to support such use cases.

5.3 ADVANTAGES AND LIMITATIONS OF HTLC

The HTLC protocol was originally designed for public blockchain networks, where there is no trusted central authority and transacting parties are possibly adversarial. It works well in facilitating atomic transactions between DLT networks while minimizing risk to transacting parties.

However, the HTLC protocol fails if the intermediary in the receiving network is not able to transfer the secret to the intermediary in the sending network within the specified time period.

The crux of this issue lies with the time-lock or rollback mechanism when the transaction is not completed in time.

In a permissioned network where participants are known, and where there are trusted third parties, there could be alternative methods of rolling back a transaction; each with its own set of considerations.

One possibility is for the rollback to be triggered manually rather than through a time-lock mechanism. This would solve the problem of the secret not being transferred in time, but it could be disadvantageous to the sender because of the opportunity cost of having funds locked up for a longer period (an example of this is in the scenario where there are deliberate delays by the intermediary.). However, in a consortium blockchain, the governance framework could prevent a deliberate delay, possibly by monitoring and imposing fees or penalties when such activities are detected.

Another possibility is for the rollback to be a transferred to an account controlled by a third party instead of directly to the sender. This would introduce dependency on a central entity, which would not work in a public blockchain, but it could possibly be a solution in consortium networks. In such a case, failed transactions would not cause financial loss, but they would incur other costs in the form of operational needs (manual intervention to release the payment) and opportunity costs (funds locked up). However, if most transactions are completed successfully, then such a solution may prove to be effective while still reducing risks.

5.4 HTLC ALTERNATIVES

Although HTLC has become quite popular and gained momentum in the financial services industry, there are other alternatives. Vitalik Buterin defines three categories of strategies for chain interoperability solutions in his paper "Chain Interoperability,"[6] defines three categories of strategies for chain interoperability solutions:

- Centralized or multisig notary schemes, where a trusted entity or a set of entities trusted as a group is used to inform chain X that a claim in chain X is true. An example of this is the Interledger payment protocol[7] by Ripple.
- Sidechains/relays, where blockchains have the capability to read and validate events from other blockchains. These are a more direct method of facilitating interoperability compared with the notary schemes. An example of this is BTC Relay[8] between Bitcoin and Ethereum.
- Hash-locking, which uses the preimage of a particular hash on both chains to perform interoperability. This is being actively explored as an interledger protocol by many public and private DLT platform providers. Along with HTLC, Clearmatics's Ion[9] is a good example of a protocol in this category.

In addition, there is active research and development in off-chain channel networks to achieve scalability and interoperability. Some examples of these are Lightning Network, Raiden and COMIT.

5.5 NETWORK SCALABILITY

The proof of concept was tested with a limited number of participants on each network, and we assumed that each participant would be able to transfer directly to any other participant in the other network. In a real-world scenario where there are hundreds to thousands of participants on each network, with tens to hundreds of inter-connected networks, such a model is untenable due to the complexity and scalability challenges. The direct node-to-node connectivity is also a single point of failure, with a negative impact on resiliency.

We hypothesize that these challenges could be overcome through alternative network models, such as:

- Using gateway nodes that act as service nodes for its network participants
- Leveraging on a centralized connector between networks, where networks register and connect directly to the broker
- Having an additional DLT to established connections between the networks

Each network model has its own considerations, benefits and limitations. More in-depth research is required.

REFERENCES

1 See Bank of Canada, Bank of England and Monetary Authority of Singapore, "Cross-Border Interbank Payments and Settlements: Emerging Opportunities for Digital Transformation," November 2018.

2 A smart contract is a software construct that computationally (not legally) binds parties to programmatically expressible commitments.

3 Corda is a DLT platform from R3 that is designed for use with regulated financial institutions. It is inspired by blockchain systems and designed for recording, managing and synchronizing commercial agreements between known and identified parties at scale without compromising privacy.

4 Quorum is a blockchain platform developed by JP Morgan. It is a fork of Ethereum, meant explicitly for enterprise use within the financial services sector.

5 We do not believe the HTLC implementation required is dependent on the choice of model from Table 1.

6 See V. Buterin, "Chain Interoperability," September 2016.

7 See S. Thomas and E. Schwartz, "A Protocol for Interledger Payments."

8 See BTC Relay.

9 See GitHub, "Ion Interoperability Framework."

ACKNOWLEDGEMENTS

BANK OF CANADA

Rakesh Arora

Dinesh Shah

Scott Hendry

André Usche

Yusu Guo

Adrian Guerin

MONETARY AUTHORITY OF SINGAPORE

Wee Kee Toh

Lizhi Chen

Damien Pang

Sopnendu Mohanty

PROJECT TEAM

Fraser Edwards (Accenture)

Hong Yi Lim (Accenture)

James Hwa Jaan Gan (Accenture)

Janis Olekss (Accenture)

Jessica Johnson (Accenture)

John Velissarios (Accenture)

Jonathan Seah (Accenture)

Naveen Mallela (JP Morgan)

Peter de Rooij (Accenture)

Raunak Rajpuria (JP Morgan)

Sai Valiveti (JP Morgan)

Samantaray Debidutta (JP Morgan)

Viswanathan Sugumar (Accenture)

William Lim (Accenture)

GLOSSARY

06

ACK	Acknowledgement, a positive response signal between data communicating processes or computers
API	Application Programming Interface
BIC	Bank Identifier Code
BOC	Bank of Canada
DApp	Decentralized Application
DDR	Digital Depository Receipts
DLT	Distributed Ledger Technology
FCY	Foreign Currency
HTLC	Hashed Time-Locked Contracts
HTTP	Hypertext Transfer Protocol
LCY	Local Currency
LVTS	Large Value Transfer System (Canada RTGS System)
MAS	Monetary Authority of Singapore
MEPS+	MAS Electronic Payment System (Singapore RTGS System)
RPC	Remote Procedure Call
RTGS	Real Time Gross Settlement
W-CBDC	Wholesales Central Bank Digital Currency

APPENDIX

07

7.1 QUORUM FRAMEWORK

Quorum is an Ethereum-based distributed ledger protocol that was developed to provide the financial services industry with a permissioned implementation of Ethereum that supports transaction and contract privacy.

Quorum includes a minimalistic fork of the Go Ethereum client (also known as Geth) and, as such, leverages the work that the Ethereum developer community has undertaken.

The primary features of Quorum, and therefore extensions over public Ethereum, are:

- Transaction and contract privacy
- Multiple voting-based consensus mechanisms
- Network/Peer permissions management
- Higher performance

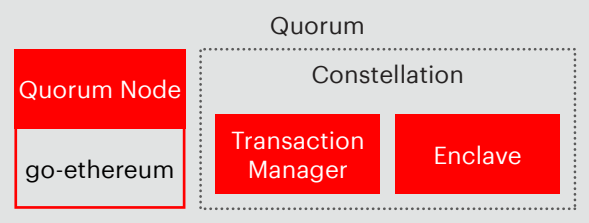
Quorum currently includes the following components:

- Quorum Node (modified Geth Client)
- Constellation/Tessera – Transaction Manager
- Constellation/Tessera – Enclave

While Quorum has been designed with financial services use cases in mind, its implementation is not specific to financial services and hence is appropriate for other industries that are interested in utilizing Ethereum but require the above primary features.

Figure 20 depicts key components of the Quorum design:

Figure 20: Quorum design



7.1.1 QUORUM NODE

The Quorum Node includes the following modifications to Geth:

- Consensus is achieved with the Raft or Istanbul BFT consensus algorithms instead of using Proof-of-Work.
- The P2P layer has been modified to only allow connections to/from permissioned nodes.
- The block generation logic has been modified to replace the 'global state root' check with a new 'global public state root'.
- The block validation logic has been modified to replace the 'global state root' in the block header with the 'global public state root'

- The State Patricia trie has been split into two: a public state trie and a private state trie.
- Block validation logic has been modified to handle 'Private Transactions'
- Transaction creation has been modified to allow for Transaction data to be replaced by encrypted hashes in order to preserve private data where required
- The pricing of Gas has been removed, although Gas itself remains

7.1.2 CONSTELLATION AND TESSERA

Constellation and Tessera are Haskell and Java implementations of a general-purpose system for submitting information in a secure way. These are comparable to a network of MTA (Message Transfer Agents) where messages are encrypted with PGP. These are not blockchain-specific, and have many other potential applications for the exchange of individually-sealed messages within a network of counterparties.
The Constellation and Tessera modules consist of two sub-modules:

- The Node (which is used for Quorum's default implementation of the Transaction Manager)
- The Enclave

Transaction Manager

Quorum's Transaction Manager is responsible for Transaction privacy. It stores and allows access to encrypted transaction data, exchanges encrypted payloads with other participant's Transaction Managers but does not have access to any sensitive private keys. It utilizes the Enclave for cryptographic functionality (although the Enclave can optionally be hosted by the Transaction Manager itself.)

The Transaction Manager is restful/ stateless and can be load balanced easily.

The Enclave

Distributed ledger protocols typically leverage cryptographic techniques for transaction authenticity, participant authentication, and historical data preservation (i.e., through a chain of cryptographically hashed data.) To achieve a separation of concerns, as well as to provide performance improvements through parallelization of certain crypto-operations, much of the cryptographic work, including symmetric key generation and data encryption/ decryption, is delegated to the Enclave.

The Enclave works hand in hand with the Transaction Manager to strengthen privacy by managing the encryption/ decryption in an isolated way. It holds private keys and is essentially a virtual hardware security module isolated from other components.

7.2 CORDA FRAMEWORK

Corda is a DLT platform from R3 that is designed for use with regulated financial institutions. It was used in the Jasper III proof of concept to build the delivery-versus-payment equity settlement system. This appendix provides a simplified explanation of Corda concepts, highlighting how Corda specifies and enforces control over use of assets on-ledger. For a detailed technical description of the Corda platform, refer to the introductory and technical white papers available on Corda's website.

A Corda distributed application (CorDapp) is a distributed application installed at the node level that leverages Corda's platform to handle business logic and processes. It consists of four components that jointly determine the capabilities and controls of that application: states, transactions, contracts, and flows.

- **States** are immutable on-ledger objects that represent shared facts. Participants that hold a state are in consensus about the contents of that state. In the proof of concept, cash and equity on-ledger are digital depository receipts (DDR), which are represented as a state that fully specifies the DDR, including its current owner.
- **Transactions** are actions, known as commands, on a set of states. Corda has an unspent transaction output (UTXO) model in which a transaction consumes a set of input states (i.e., marks them as historic or no longer valid) and produces a set of output states, as specified by one or more commands. States are immutable, but transactions offer a way to mark consensus on a change of the facts in the state: it consumes the state with the old values and produces a new state with updated values. For example, to transfer some W-CBDC to another participant, a "transfer" transaction would consume a state with the payer as its owner, have one single "transfer" command, and produce a new state with the payee as its owner.

Corda transactions are atomic: they either succeed entirely or fail entirely. For the transfer transaction, this means that either the payer owns the old W-CBDC, or the payee owns the new W-CBDC. A situation where both W-CBDC states are valid, or that neither of them is, will never happen as a result of this transaction.

- **Contracts** specify the rules associated with a state. They take a proposed transaction as input and define whether the transaction is valid based on the contract's rules for every input and output state present in the transaction. For example, a W-CBDC contract is associated with all W-CBDC states, and it specifies (among other rules) that a transfer transaction cannot change the issuer, increase the amount, or change the currency.
- **Flows** specify how participants communicate transactions and reach consensus on them.

Ultimately, flows orchestrate the inclusion of states from each necessary party, the contracts that govern these states, and the collection of signatures from these same parties to establish a new transaction, which is stored to each participant's ledger.

The Corda platform provides a network map service that manages and publishes the identities of the nodes on the network. This service can be distributed and run by an independent party.

The pluggable notary service of the Corda platform provides a uniqueness and/or validating consensus by attesting to the finality of the transactions. A notary may be a single network node, a cluster of mutually trusting nodes, or a cluster of mutually distrusting nodes. Corda also has a "pluggable" consensus, allowing notaries to choose an algorithm based on their requirements in terms of privacy, scalability, legal-system compatibility, and algorithm agility. In the Jasper III proof of concept, the notary provides only uniqueness consensus.

ABOUT ACCENTURE

Accenture is a leading global professional services company, providing a broad range of services and solutions in strategy, consulting, digital, technology and operations. Combining unmatched experience and specialized skills across more than 40 industries and all business functions—underpinned by the world's largest delivery network—Accenture works at the intersection of business and technology to help clients improve their performance and create sustainable value for their stakeholders. With 477,000 people serving clients in more than 120 countries, Accenture drives innovation to improve the way the world works and lives.
Visit us at www.accenture.com.

181931