EUROPEAN CENTRAL BANK

EUROSYSTEM

BANK OF JAPAN

STELLA - a joint research project of the European Central Bank and the Bank of Japan

Securities settlement systems: delivery-versus-payment in a distributed ledger environment

March 2018

Contents

1 Background 2

2 Main findings of the joint analysis 3

3 Delivery versus payment (DvP) 4

3.1 Definition of DvP 4

3.2 DvP approaches in DLT environment 5

4 Process flow of DvP on DLT 7

4.1 Process flow of single-ledger DvP 9

4.2 Process flow of cross-ledger DvP with HTLC 12

5 Analysis of DvP approaches on DLT 21

5.1 Achievement of DvP 22

5.2 Efficiency in the use of liquidity 23

5.3 Speed of settlement 23

5.4 Privacy 24

5.5 Interdependency between ledgers 25

5.6 Infrastructure design 25

Annex 1: DvP in Bank of Japan and ECB services 27

Annex 2: Process fail scenarios 30

Annex 3: Unspent Transaction Outputs (UTXO) 32

Annex 4: Process flow on Corda 36

Annex 5: Process flow on Elements 41

Annex 6: Process flow on Fabric 46

EUROPEAN CENTRAL BANK
EUROSYSTEM

STELLA

BANK OF JAPAN

Project Stella

Securities settlement systems: Delivery versus payment in a distributed ledger environment

1 Background

This report is the outcome of the second phase of Project Stella, the joint study on distributed ledger technology (DLT) by the Bank of Japan (BOJ) and the European Central Bank (ECB).[1] The objective of Stella phase 2 is to explore how the settlement of two linked obligations, such as the delivery of securities against the payment of cash, could be conceptually designed and operated in an environment based on DLT.[2] Legal aspects have not been the object of the study.

Settlement mechanisms based on delivery versus payment (DvP) link the transfer of two assets in such a way as to ensure that the transfer of one asset occurs if and only if the transfer of the other asset also occurs. The outcome of settlement is either both parties successfully exchanging those assets, or no transfer taking place. Such a condition is also often referred to as "atomicity" in computer science.[3]

DLT is a combination of technological components such as cryptography, peer-to-peer (P2P) networks, consensus algorithms and smart contracts, that allows multiple parties to share and process data without a single central management system.

This report examines ways in which DvP can be conceptually designed and technically achieved in a DLT environment drawing on existing DvP models as well as innovative solutions that are being discussed for distributed ledgers. In order to gain practical understanding on the DvP functioning on DLT, prototypes were developed using three DLT platforms, Corda, Elements and Hyperledger Fabric (hereafter referred to as Fabric).

---

1 The joint research was conducted by Shuji Kobayakawa (BOJ team leader), Yuji Kawada and Akiko Kobayashi from the BOJ, and by Dirk Bullmann (ECB team leader), Andrej Bachmann, Cedric Humbert, Stephan Mögelin (seconded expert from Bundesanstalt für Finanzdienstleistungsaufsicht), Giuseppe Galano (Banca d'Italia), with contributions from José Cerecedo, Thomas Leach and Andrea Pinna from the ECB.

2 The authors are grateful to R3, IBM and DG Lab for technical advice.

3 The word, "atomic", comes from Greek meaning indivisibility or irreducibility. Atomic operations cannot be divided into smaller operations; either (i) all operations are fully performed or (ii) they are not performed at all.

The analysis is based on a basic, stylised scenario of two counterparties exchanging securities against cash. As was the case in Stella phase 1,[4] this work does not attempt to replicate existing payment and securities settlement systems and is not geared towards replacing existing central bank services with DLT-based solutions.

The report is structured as follows: Section 2 presents the most salient results of the second phase of the project. Section 3 provides background information on DvP and the categorisation of DvP solutions in a DLT environment based on whether assets are on the same ledger or on separate ones. Section 4 describes generic process flows for each DvP approach and Section 5 assesses their main characteristics from a range of perspectives.

2 Main findings of the joint analysis

The main findings of the joint analysis, as detailed in this report, can be summarised as follows:

DvP can run in a DLT environment subject to the specificities of the different DLT platforms: DvP could be conceptually and technically designed in a DLT environment with cash and securities on the same ledger (single-ledger DvP) or on separate ones (cross-ledger DvP). The concrete design of DvP, however, depends on the characteristics of the DLT platforms (e.g. range of information shared among participants, data structure and locking of delivered assets). In addition, depending on the use case, the design of DvP can be influenced by a number of factors including the interaction of the DvP arrangement with other post-trade infrastructures.

DLT offers a new approach for achieving DvP between ledgers, which does not require any connection between ledgers: Conceptual analysis and conducted experiments have proven that cross-ledger DvP could function even without any connection between individual ledgers, a novelty which does not exist in today's set-up. Functionalities such as "cross-chain atomic swaps" have the potential to help ensure interoperability between ledgers (of either the same or different DLT platforms) without necessarily requiring connection and institutional arrangements between them.[5]

Depending on their concrete design, cross-ledger DvP arrangements on DLT may entail certain complexity and could give rise to additional challenges which would need to be addressed: The conduct of DvP transactions between ledgers that have no connection requires several process steps and interactions between the seller and the buyer. Depending on the concrete design, this could impact transaction speed and require the temporary blockage of liquidity. It should also be borne in mind that independently acting ledgers may inadvertently affect

[4] European Central Bank and Bank of Japan (September 2017), "Payment systems: liquidity saving mechanisms in a distributed ledger environment".

[5] From a technical point of view, functionalities that enable "cross-chain atomic swaps" could be implemented for non-DLT platforms.

each other from an operational perspective. From a risk perspective, also the absence of fully synchronised process steps could expose participants to principal risk if one of the two counterparties does not complete the necessary process steps. Those additional risk aspects would need to be properly addressed.

3 Delivery versus payment (DvP)

3.1 Definition of DvP

DvP is defined in the existing international standards for financial market infrastructures, namely the Principles for financial market infrastructures (PFMIs).[6]

A DvP transaction involves the settlement of two linked obligations, namely the delivery of securities and the payment of cash. Principle 12 of the PFMIs requires that "*[…] the final settlement of one obligation occurs if and only if the final settlement of the linked obligation also occurs, regardless of whether the FMI settles on a gross or net basis and when finality occurs*". DvP avoids counterparties being exposed to principal risk, i.e. the risk that the seller of securities could deliver but would not receive payment or that the buyer of securities could make payment but would not receive delivery.[7]

Following this requirement, a DvP securities settlement mechanism has to ensure that the delivery of securities and the payment of cash are linked in a way where one leg (obligation) of the securities trade is conditioned to the final settlement of the other leg (obligation) of the trade. Thereby final settlement is defined as "*the irrevocable and unconditional transfer of an asset or financial instrument, or the discharge of an obligation by the FMI or its participants in accordance with the terms of the underlying contract"*.

DvP can be achieved through different timing arrangements and does not require, in principle, a simultaneous settlement of securities and cash. As a consequence, DvP can be achieved by settlement mechanisms where settlement of one obligation follows the settlement of the linked obligation within certain time lag. DvP can also be achieved either on a gross or on a net basis.[8]

---

6 Committee on Payment and Settlement Systems (CPSS) and Technical Committee of the International Organization of Securities Commissions (IOSCO), “Principles for financial market infrastructures” (April 2012).

7 See Section 5 for discussion of types of risks faced by counterparties to securities trades.

8 For discussion of stylised models of DvP settlement, see CPSS (September 1992), "Delivery versus payment in securities settlement systems".

3.2 DvP approaches in DLT environment

There are conceptually two distinct approaches on how DvP could be performed in a DLT environment. It could either be assumed that cash and securities are on the same ledger (single-ledger DvP) or on separate ones (cross-ledger DvP).

Single-ledger DvP: In the single-ledger approach, cash and securities are recorded on the same ledger. In this case, the exchange of both assets is typically processed as a single transaction, after two counterparties have each confirmed the transfer instruction. This concept is similar to the "integrated model" in existing securities settlement mechanisms (such as TARGET2-Securities and BOJ-NET JGB Services; see Annex 1) in which securities and cash are processed on a single integrated platform.

Cross-ledger DvP: In the cross-ledger model, cash and securities are recorded on two separate ledgers, with mechanisms in place to link the transfer of the two assets.[9] This approach is inherently complex because one network relies on another's update to proceed; the rules that govern one network thus need to account for the other. The cross-ledger model can be broken down further into two types.

1. **Cross-ledger DvP with connection[10] between ledgers:** This model is analogous to the "interfaced model" in existing securities settlement mechanisms in which securities and cash are settled in two different systems and the two systems coordinate to facilitate DvP by blocking assets (such as the DvP link between BOJ-NET Funds Transfer Service and Japan Securities Depository Center, see Annex 1). In the existing model, typically, a securities settlement system first blocks the securities to be delivered by either earmarking balances or transferring the securities to an escrow account. Based on the confirmation from the securities settlement system, the payment system processes the transfer of cash and informs the securities settlement system to release the securities. After receiving this confirmation from the payment system, the securities settlement system releases the block and processes the transfer of securities.

   In a distributed ledger environment, such an approach may require an intermediary to facilitate and control the coordination between the two ledgers. The intermediary could be further categorised into a trusted or an untrusted intermediary,[11] although at this stage of development, mechanisms to achieve two-way payments such as DvP have not yet been well described other than those using a trusted intermediary. In this study, this approach is not focused and is left for future developments.

---

9 Cross-ledger DvP is often discussed in the context of "interoperability" among different ledgers including non-DLT ledgers.

10 Connection between ledgers could be either direct or indirect. Indirect connection means the two ledgers are connected via a third ledger.

11 The approach using an untrusted intermediary, which acts just like a relay point (or a payment processor), is being actively developed by DLT communities both technically and theoretically, including designs for incentive mechanisms for those intermediaries.

2. **Cross-ledger DvP without connection between ledgers:** Cross-ledger DvP without connection between ledgers does not exist in today's set-up. Following the emergence of DLT, functionalities such as "cross-chain atomic swaps" were developed which could facilitate DvP without any connection between the two ledgers or a single entity coordinating the interactions between them.

   Cross-chain atomic swaps mechanisms were originally developed for the purpose of exchanging two crypto-assets on two separate blockchains without relying on a third party.[12] The key elements to achieving cross-chain atomic swaps are the use of digital signatures and "Hashed Timelock Contracts" (HTLC) to support the atomicity in transferring two assets across two separate ledgers.[13]

   HTLC consists of (i) a cryptographic hash function and (ii) a timelock (or timeout) function. For example, where two counterparties A and B transfer cash against securities in the cross-ledger approach using HTLC. A and B first block the assets to be delivered using the one-way hash of the secret value that A has generated.[14] After both assets are blocked, counterparty A retrieves the asset it was planning to receive (e.g. cash) by using the secret value. At the same time the secret is disclosed to B. Counterparty B then retrieves the asset it was planning to receive (e.g. securities) using the same secret.[15] A timelock function ensures the refund of securities and cash to the original holders in the event that the necessary process flows are not completed within the predefined time period (for further explanation, see Section 4.2).[16]

   Although there could be other solutions for cross-ledger DvP without connection between ledgers, the report focuses on cross-ledger DvP on the HTLC-based approach (hereafter, cross-ledger DvP with HTLC).

---

12 The original idea was first described by Tier Nolan in 2013 (https://bitcointalk.org/index.php?topic=193281). However, it had security risks due to several technical limitations at that time. To be concrete, OP_CSV (BIP-112), OP_CLTV (BIP-65) and "Segregated Witness" (so-called "SegWit") were not introduced and refunding transactions could be tampered with due to the transaction malleability problem. Later, the introduction of those functionalities eliminated the technical limitations. In this study, the modified version of the original Tier-Nolan approach, which does not require signing refunding transactions beforehand, was used. For further information, see Annex 5.

13 HTLC is one of the building blocks of Lightning Networks and similar ideas are also being used in Ripple Interledger Protocol, although they assume connections between ledgers and they could be categorised into "Cross-ledger DvP with connection between ledgers". For further information about HTLC, refer to Joseph Poon and Thaddeus Dryja (January 2016), "The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments".

14 By using the one-way hash function, it is impossible with reasonable assumptions for counterparty B to find the original value (secret) generated by A from its hash.

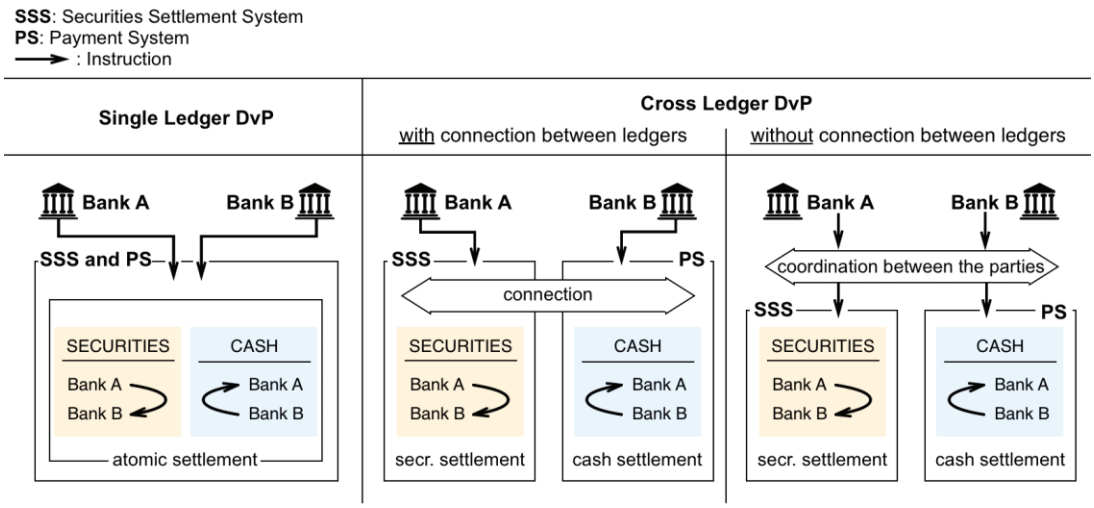
15 For further information of sharing of secrets, see Section 5.4.

16 One of the important characteristics of this approach is the asymmetry between the two counterparties. For example, a secret value is generated by either of the two counterparties. If both of them generate secret values respectively and both deliveries of assets are blocked by hashes of the respective secret values, the retrieval of assets is not possible until either of them shares the secret with the other, which is the same situation as the one without using those secret values. The same is true of the duration of two types of locking time specified by both counterparties. For further information, see Section 4.2 "Requirements - asymmetry of locking time".

It should be noted that various arrangements for coordination between ledgers have been discussed and tested in the DLT communities, some of which may not fit in the above categories.[17]

Chart 1

Stylised approaches for DvP on DLT



4 Process flow of DvP on DLT

In Stella phase 2, the ECB and the BOJ analysed DvP arrangements based on conceptual analysis and the conduct of practical experiments. The analysis is based on a basic and stylised scenario of two counterparties (Bank A and Bank B) exchanging the agreed amounts of securities against cash without a single central management system.

Prototypes were implemented for "single-ledger DvP" and for "cross-ledger DvP with HTLC" using three DLT platforms, namely Corda, Elements, and Fabric. This section describes the generic process flows of these prototypes, as well as potential scenarios in which settlement fails might occur.[18] Potential settlement fail scenarios which are closely linked to the relevant process flows are provided in Annex 2 and further details on how the process flow is designed for Corda, Elements and Fabric can be retrieved from Annexes 4, 5 and 6. In this section we will abstract from the

[17] For example, "trust model", instead of connection between ledgers, is used for categorisation in Vitalik Buterin (September 2016), "Chain Interoperability"; https://static1.squarespace.com/static/55f73743e4b051cfcc0b02cf/t/5886800ecd0f68de303349b1/1485209617040/Chain+Interoperability.pdf.

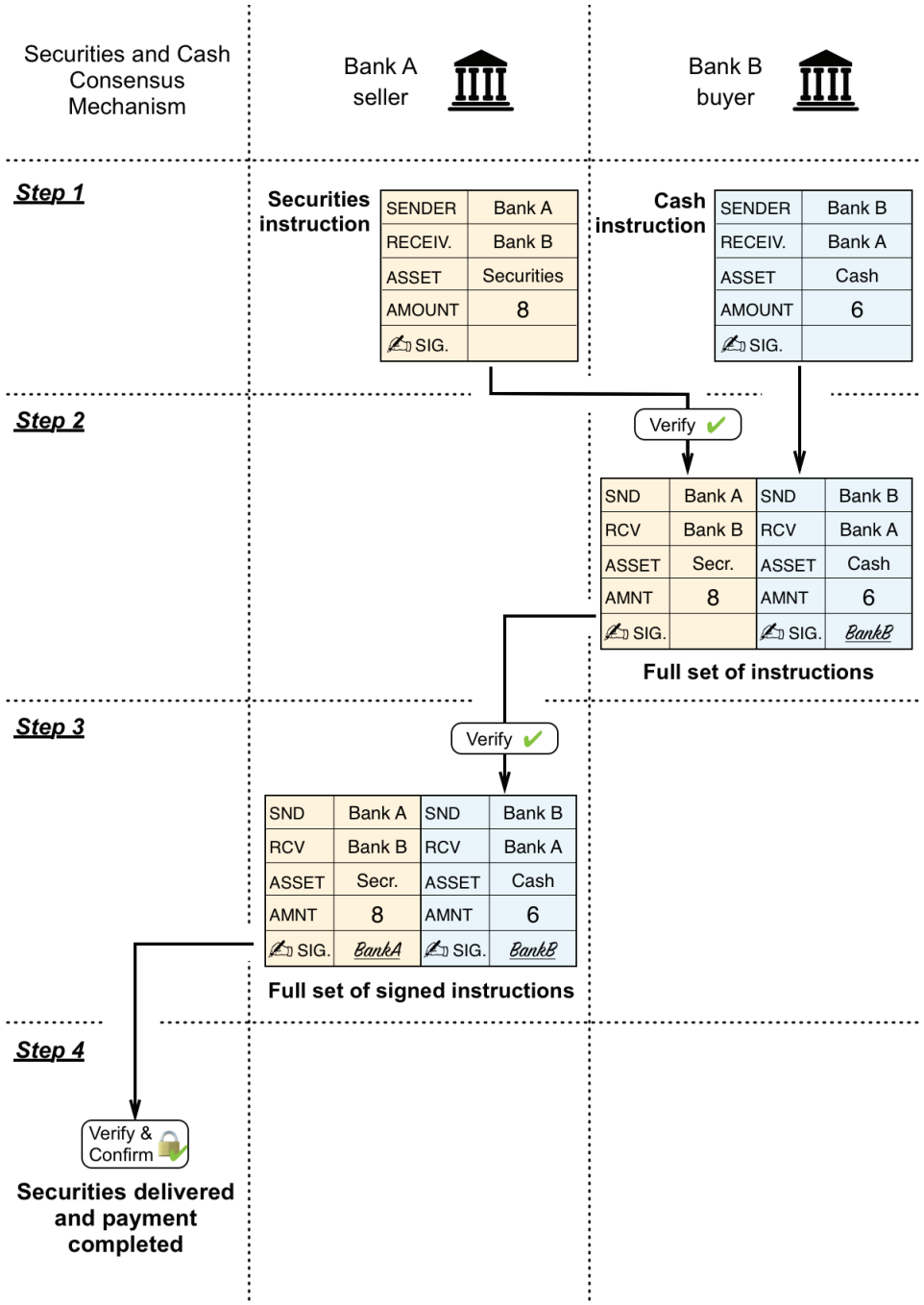
[18] In order to highlight the essence of the process flow, subsequent descriptions are based on the UTXO (Unspent Transaction Output) model. A description of "fee" and "change" is omitted in the case of Elements.

platform-specific implementation and use the term "consensus mechanism" to describe the component in charge of ensuring the consistency of the ledger.[19]

[19] In the process flows, "consensus mechanism" encapsulates the platform-specific mechanisms for transaction relaying, verification and confirmation, time-ordering, smart-contract execution and commitment.

4.1 Process flow of single-ledger DvP

Chart 2
Process flow of single-ledger DvP



The idea behind the single-ledger DvP is for the two counterparties to agree on the contents of the transfer instructions, which are then processed as a single transaction. The process of agreeing on the content of the instruction and combining the two legs into a single transaction can be conducted directly between both counterparties using cryptographic signatures without a need for a specific matching function on the DLT network.

In the diagram above, the seller of securities (Bank A) and the buyer of securities (Bank B) have agreed to the "amount" and the "asset type" to be exchanged. The agreement comprises of two transfers: (i) 8 units of securities from Bank A to Bank B, and (ii) 6 units of cash from Bank B to Bank A. Both Bank A and Bank B have access to the same DLT network where the transfer of both securities and cash is settled.

Settlement Success Scenario: Under this setup, settlement is successful when participants follow the following steps (see Chart 2):

1. Bank A (original holder of the securities) creates the securities instruction (spending of the agreed amount of securities) and Bank B (original holder of cash) creates the cash instruction (spending of the agreed amount of cash). At this stage, neither of the instructions has been signed yet.

2. Bank A sends his part of the instruction (securities instruction) to Bank B without his signature.[20] Bank B verifies the contents of the securities instruction and combines that securities instruction with the cash instruction of his own, thereby creating a full set of instructions. Bank B signs his part of instruction (cash instruction) and sends it back to Bank A.

3. Bank A verifies the full instruction and signs his part of instruction (securities instruction) and submits the full-signed instruction to the consensus mechanism.

4. Following the implemented consensus mechanism of the platform, the submitted full-signed instruction is verified and confirmed,[21] and the results are written on the ledger.[22]

---

[20] Either Bank A or Bank B can be the initiator of the transaction; for this study Bank A is the initiator.

[21] The verifications made by the two counterparties ensure validity with respect to their agreements, while those made by the platform ensure confirmation with respect to its consensus mechanism.

[22] Actual flow of transaction processing/commitment depends on the architecture of DLT platforms.

For example, in Corda, after a validating notary service checks the uniqueness of the input states, it executes verifications and signs the transaction marking the previous states as spent. The requester of the finalisation then broadcasts the signed transaction to all parties involved, and nodes commit it on its ledger accordingly.

In Elements, other nodes that receive the signed instruction verify it and broadcast it to the specified number of nearby nodes. After a miner node receives and verifies the signed instruction, it creates a block containing the instruction and broadcasts it. Each node then verifies the content of the block and commits it on its ledger.

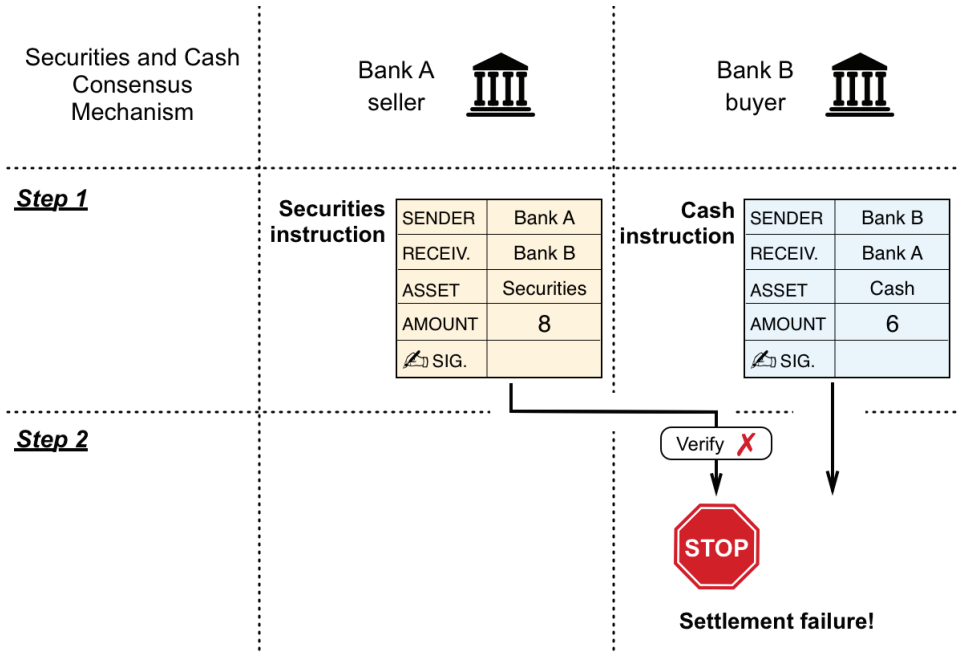
In Fabric, after endorsers execute the instruction, they send back a response value and read/write sets as well as their signatures to Bank A. Bank A verifies the result from endorsers, then sends the signed read/write sets to an orderer. An orderer creates a block which includes the signed read/write sets and broadcasts it to all nodes in the same channel. Each node then commits it on its ledger.

At this point, the agreed amounts of cash and securities are transferred to Bank A and Bank B respectively.

Potential settlement fail scenarios: In the single ledger approach, settlement fails could occur if one of the steps described is not completed (e.g. Bank B finds that the Bank A's instruction has errors). In any of these possible scenarios the process is suspended, the instruction is not confirmed by the consensus mechanism and no update is committed on the ledger. As a result, cash and securities will be kept by the original holders and they can be immediately used in other transactions.[23] The following chart illustrates an example of the potential settlement fail scenarios.

Chart 3

Settlement fail scenario of single-ledger DvP, process is suspended at step 2

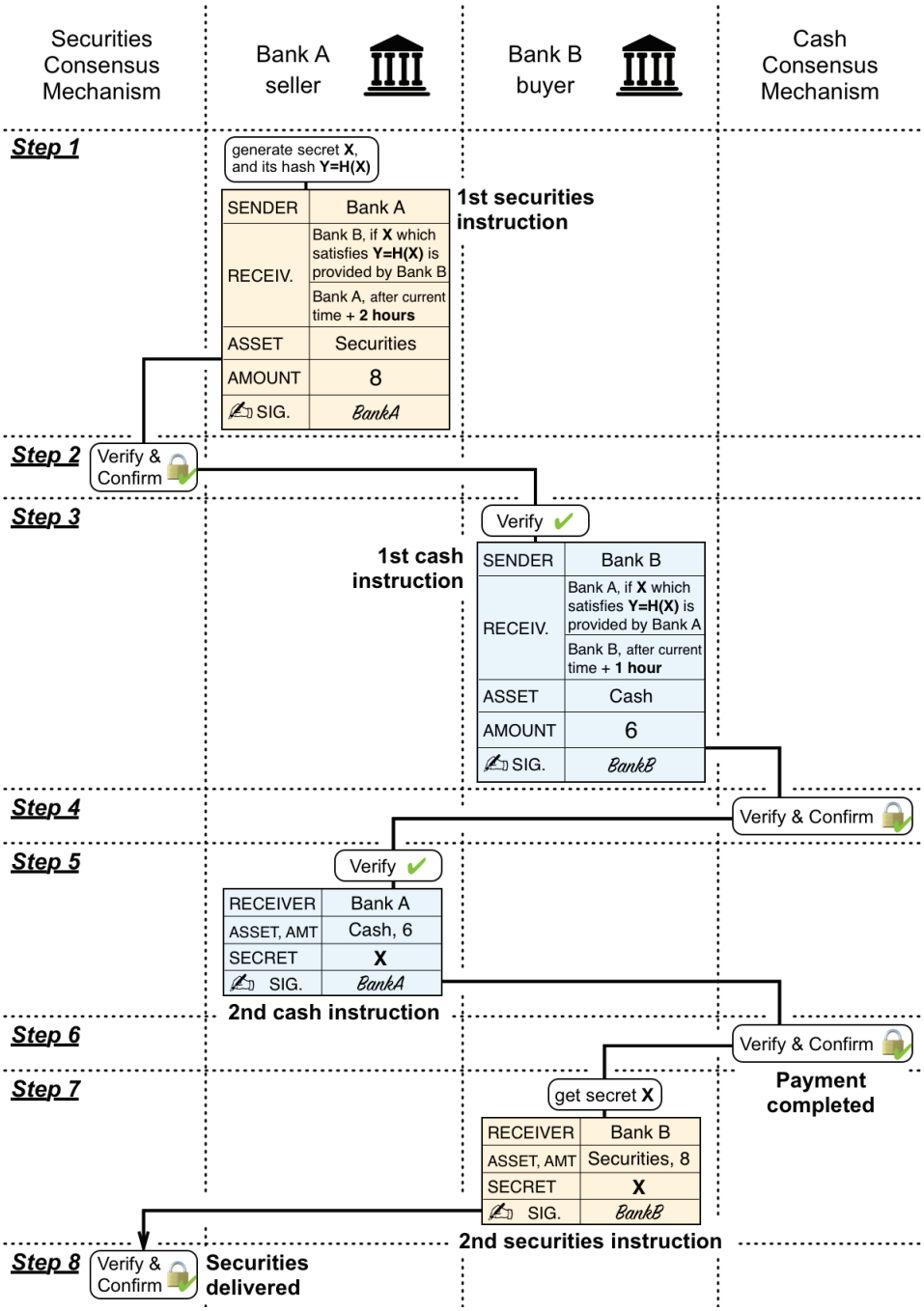


[23] Some locking mechanism can be implemented on the participants' sides to avoid creating conflicting transactions that comprise the same input UTXOs. For further information, see Annexes 4, 5 and 6.

4.2 Process flow of cross-ledger DvP with HTLC

Chart 4

Process flow of cross-ledger DvP with HTLC[24]



[24] See footnotes 18 and 19.

The idea behind the cross-ledger DvP is (i) for the two counterparties to agree on the contents of the transfer instructions based on the committed records on ledgers and (ii) to use HTLC for conditional delivery of securities and payment of cash. To be concrete, a cryptographic hash function enables the two counterparties to block the assets to be delivered and a timelock enables them to recover the assets when the process fails. As with the single-ledger process flow, the cross-ledger DvP process flow enables counterparties to directly match settlement instructions between counterparties by using cryptographic signatures. In addition, the cryptographic hash of the secret links all instructions throughout the process flow. As with the single-ledger DvP process flow, there is no need for a specific matching function on the DLT network.

In the diagram above, the seller of securities (Bank A) and the buyer of securities (Bank B) have agreed to the "amount", the "asset type", the "locking time" and the "cryptographic hash function" (H) to be exchanged. The agreement comprises of two sets of transfers: (i) 8 units of securities from Bank A to Bank B within two hours, and (ii) 6 units of cash from Bank B to Bank A within one hour.[25] Both Bank A and Bank B have access to the DLT networks where securities and cash are settled respectively and the flow of time of these networks is predictable by both participants (see Section 5.5).

Settlement Success Scenario: Settlement is successful when participants follow the following steps (see Chart 4):

1. Bank A (original holder of the securities) generates a secret ($\boldsymbol{X}$) and its hash ($\boldsymbol{Y} = \mathrm{H}(\boldsymbol{X})$).[26] Bank A shares $\boldsymbol{Y}$ with Bank B. As long as a one-way hash function is used, it is impossible within reasonable assumptions for Bank B to find $\boldsymbol{X}$ from $\boldsymbol{Y}$. Bank A creates the 1st securities instruction (spending of the agreed amount of securities). In this instruction, Bank A specifies the following two states: (i) the receiver of the securities will be Bank B if Bank B provides $\boldsymbol{X}$ which satisfies $\boldsymbol{Y} = \mathrm{H}(\boldsymbol{X})$, or (ii) the receiver of securities will be Bank A if two hours pass. Bank A then signs it and submits the signed instruction to the securities consensus mechanism.

2. Following the implemented consensus mechanism of the platform, the submitted 1st securities instruction is verified and confirmed, and results are written on the ledger in the securities DLT network.

3. Bank B (original holder of the cash) verifies the content of the committed 1st securities instruction of Bank A. Bank B then creates the 1st cash instruction (spending of the agreed amount of cash). In this instruction, Bank B specifies the following two states: (i) the receiver of cash will be Bank A if Bank A provides $\boldsymbol{X}$ which satisfies $\boldsymbol{Y} = \mathrm{H}(\boldsymbol{X})$, or (ii) the receiver of cash will be Bank B if

[25] In most of the DLT platforms used in this study, the locking time can be defined either as an absolute time (e.g. 12:00 AM, 31th, March, 2018) or a relative time (e.g. within one hour after the instruction is created). The locking time used in the process flow description is for illustrative purposes only and actual implementation would differ based on the configuration of the environment.

[26] Either Bank A or Bank B can be the generator of the secret; for this study Bank A is its generator.

1 hour passes. Bank B signs it and submits the signed instruction to the cash consensus mechanism.

4. Following the implemented consensus mechanism of the platform, the submitted 1st cash instruction is verified and confirmed, and the results are written on the ledger in the cash DLT network.

5. Bank A verifies the content of the committed 1st cash instruction of Bank B. Bank A then creates the 2nd cash instruction (obtaining of the agreed amount of cash) providing X, signs it and submits the signed instruction to the cash consensus mechanism.

6. Following the implemented consensus mechanism of the platform, the submitted 2nd cash instruction is verified and confirmed, and results are written on the ledger in the cash DLT network.

   At this point, the agreed amount of cash is transferred from Bank B to Bank A.

7. Bank B obtains X specified in the committed 2nd cash instruction of Bank A. Bank B then creates the 2nd securities instruction (obtaining of the agreed amount of securities) providing X, signs it and submits the signed instruction to the securities consensus mechanism.

8. Following the implemented consensus mechanism of the platform, the submitted 2nd securities instruction is verified and confirmed, and results are written on the ledger in the securities DLT network.

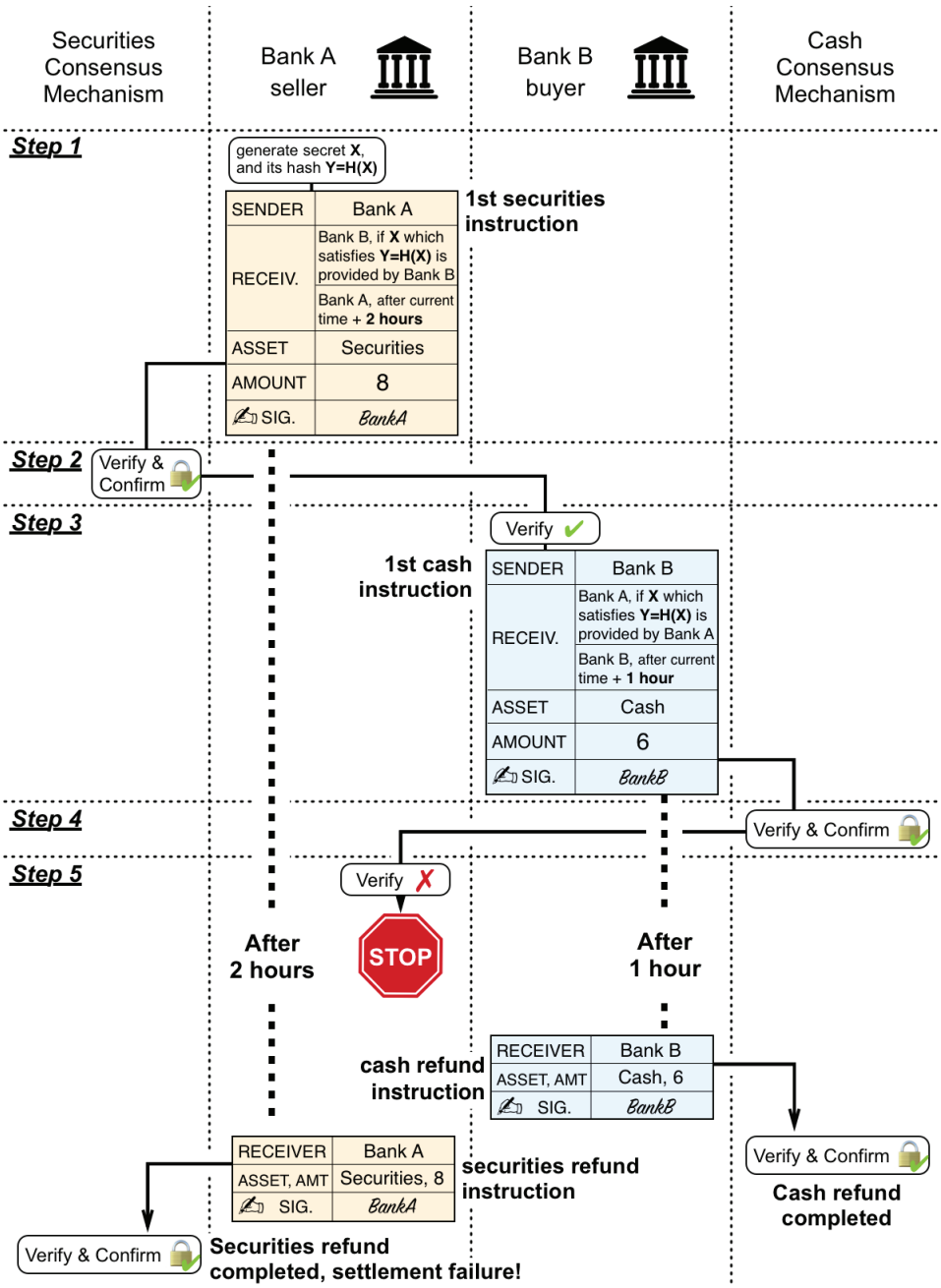
   At this point, the agreed amount of securities is transferred from Bank A to Bank B.

Potential settlement fail scenarios: Settlement could fail if one of the steps described above is not completed. For cross-ledger DvP with HTLC, potential settlement fail scenarios could result in two different risk scenarios. In the first scenario, settlement is not successful and cash and securities are returned to the original holders. In the second scenario, settlement is not successful and one of the counterparties could be exposed to principal risk.

In the first scenario (see Chart 5 when the process is suspended at step 5), settlement fails could occur, for example, where the 1st securities instruction and the 1st cash instruction are completed but Bank A (receiver of cash and generator of the secret) does not submit the 2nd cash instruction within the predefined locking time (one hour). In this case, although the transfer of both cash and securities is not successful, neither of the counterparties is exposed to principal risk as the assets are returned to the original holders after the locking time expires. The counterparties would, however, be exposed to replacement cost risk and liquidity risk.

Chart 5

Settlement fail scenario of cross-ledger DvP with HTLC, process is suspended at step 5



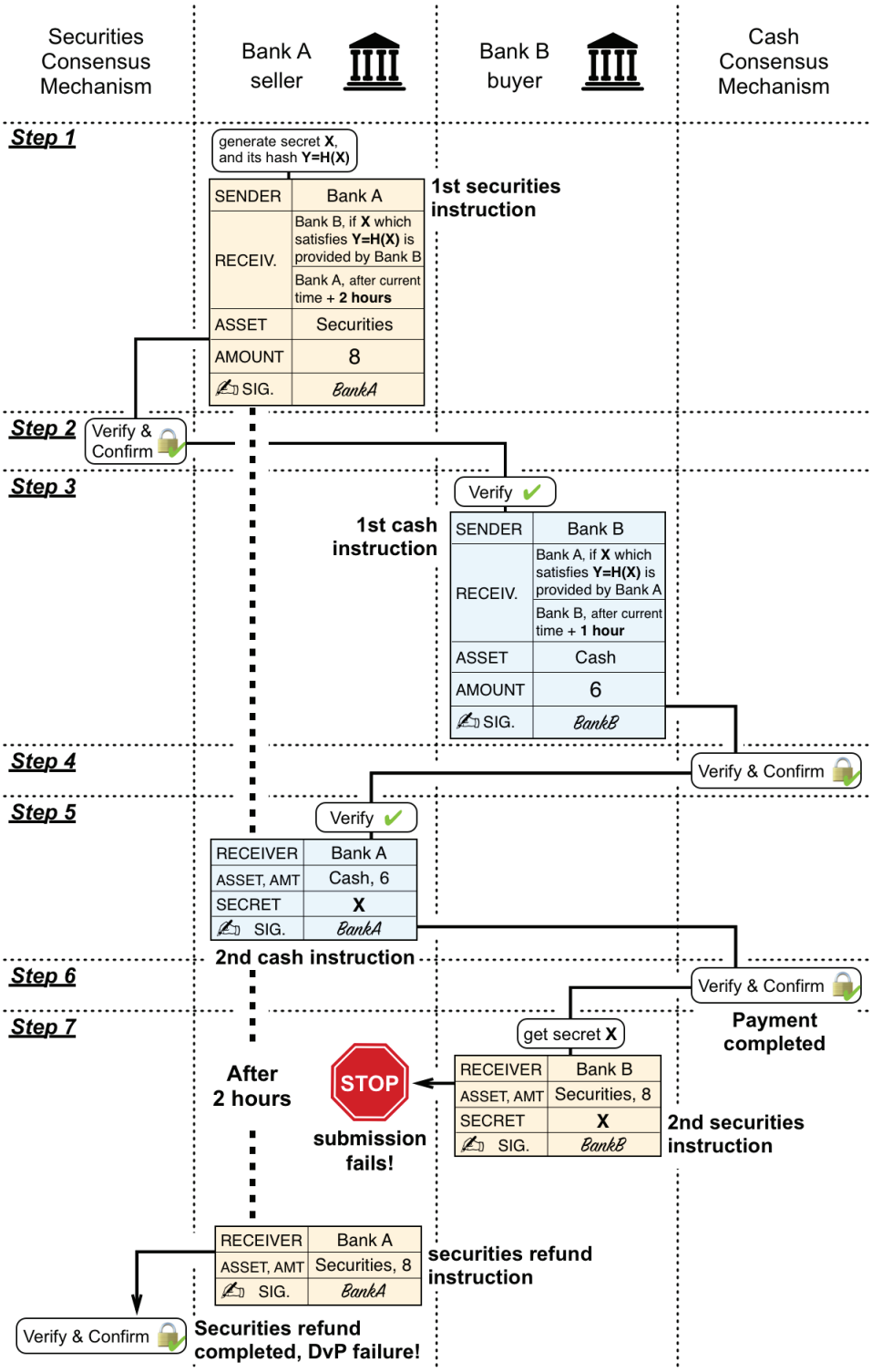
In the second scenario (see Chart 6 when the process is suspended at step 7), settlement fails could occur during the process flow where one counterparty (here Bank A) already retrieved the agreed amount of cash and the other counterparty (here Bank B) did not complete the 2nd securities instruction within the predefined locking time (two hours). In this case the locking time for the latter instruction will expire and the original holder (Bank A) can refund the locked assets (securities).

Ultimately, this counterparty (Bank A) will hold both his refunded assets (securities) and the retrieved assets (cash) while the other counterparty (Bank B) will be exposed to principal risk. In this specific fail scenario only one leg of the transaction is settled and DvP will not be achieved.[27] This scenario illustrates the weakness of HTLC and stresses the need for further developments.

[27] Several arrangements may be considered to mitigate such risks. For example, the locking time could be set at a large interval (e.g. 24 hours, 48 hours). Larger difference between the two locking times increases the likelihood of successful settlements, while it also reduces the efficiency in the use of liquidity when a settlement fails. Another approach could be for Bank B to incentivise a third party to send the 2nd securities transaction on its behalf, with the assumption that an instruction with a cryptographic signature can only be changed by Bank B.

Chart 6

Settlement fail scenario of cross-ledger DvP with HLTC, process is suspended at step 7

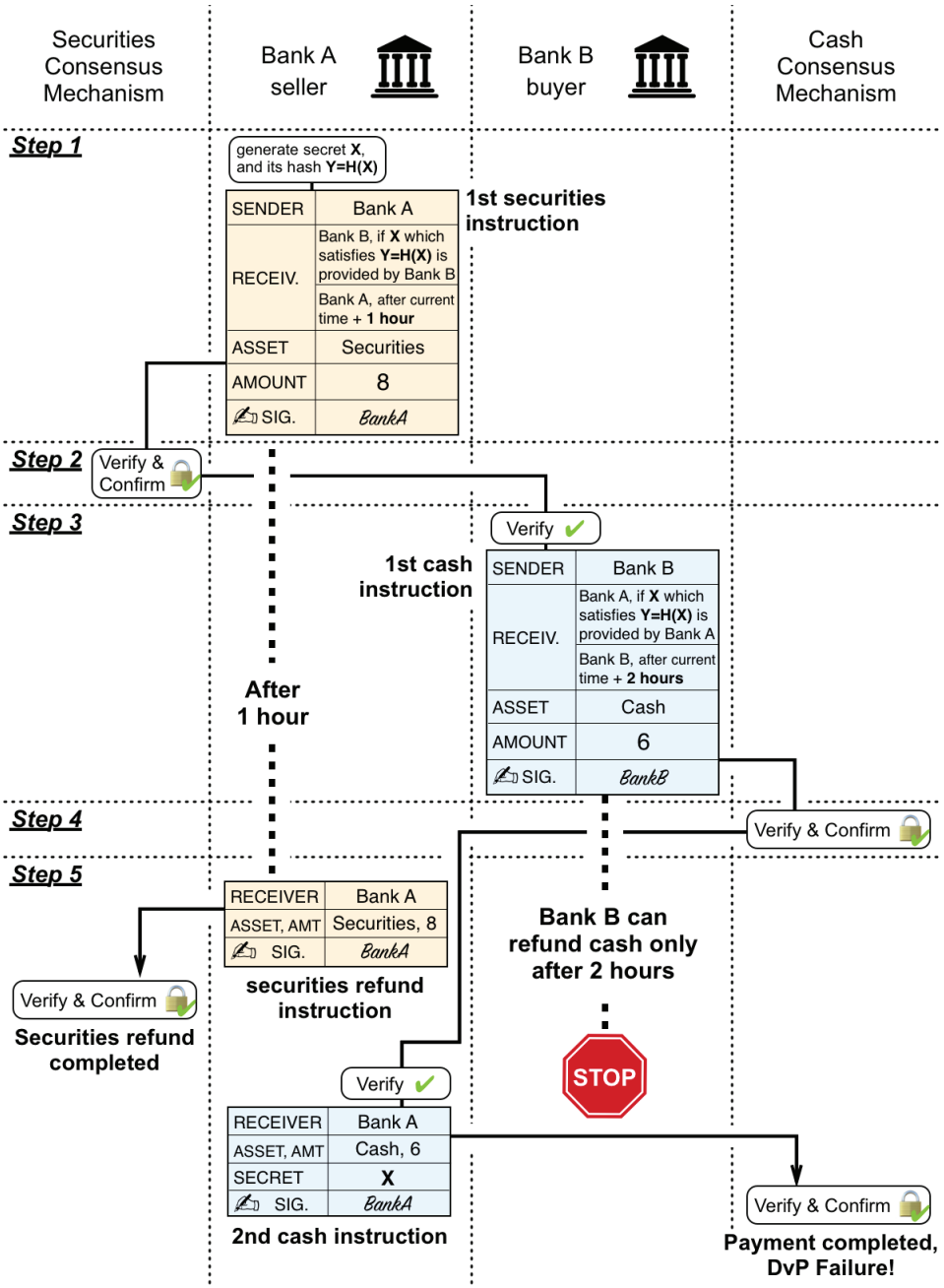


Requirements – asymmetry of locking time: It should be noted that the locking time of one asset (e.g. two hours) should be larger than that of the other asset (e.g. one hour) to prevent a counterparty refunding the assets while obtaining the other asset.

For example (see Chart 7), if the locking time of the 1st securities instruction is one hour and that of the 1st cash instruction is two hours, then assuming the flow of time of the two DLT networks is the same, Bank A would be able to refund securities and obtain cash by not sending the 2nd cash instruction within one hour.

Chart 7

Settlement fail scenario of cross-ledger DvP, two types of locking time are not properly set



BOX

Overview of the three DLT platforms used

In Stella phase 2, the practical experience was gained with three open-source DLT platforms, Corda, Elements, and Fabric.[28] This box briefly introduces the main characteristics of the three platforms.

Corda: Corda is a platform designed to record, manage and automate legal agreements between known and identified parties. The network is permissioned with access control by public key infrastructure (PKI) based authentications. This platform has been used for the projects led by other central banks including the Bank of Canada's "Project Jasper"[29] and phase 2 of the Monetary Authority of Singapore's "Project Ubin".[30]

Elements: Elements is a platform providing functionalities such as (i) the ability to issue multiple assets where asset identifiers and their amounts are encrypted but still auditable and (ii) pegged sidechains that enable assets to be transferred from/to other blockchains.[31]

Fabric: Fabric is a platform for distributed ledger solutions, underpinned by a modular architecture delivering an enterprise-ready blockchain. As with Corda, the Fabric's network is permissioned and this platform has also been used in the first phase of Stella.[32] It has been used in other projects such as phase 2 of Project Ubin and the joint project of the Deutsche Bundesbank and Deutsche Börse.[33]

These platforms have their own uniqueness in many respects including the trust model of network participants, the consensus algorithm, and the transaction flow. However, in terms of designing DvP in the DLT environment, the following key characteristics need to be considered.

Finality of settlements: For DvP arrangements it is important that once a transfer of assets is completed the transfer is irrevocable. With respect to DLT platforms, changes in the balances of participants as a result of a DvP transaction should be irrevocable after the transaction was committed on the ledgers. The finality of settlements depends on the consensus algorithm. Both in Corda and Fabric, a deterministic result is guaranteed. By contrast, in Elements, deterministic finality is not guaranteed when proof-of-work is used; however, in order to achieve the deterministic finality, mining can be restricted to a single node while the ledger is distributed over the peers.[34]

Range of information shared among the participants: Cross-ledger DvP with HTLC depends on the privacy model of each platform. In Corda, communication between nodes is point-to-point, which

28 In this study, Corda release-V2, Elements v2.14.1.1, and Hyperledger Fabric v1.1.0-alpha were used.

29 https://www.bankofcanada.ca/wp-content/uploads/2017/05/fsr-june-2017-chapman.pdf.

30 http://www.mas.gov.sg/~/media/ ProjectUbin/ Project%20Ubin%20Phase%202%20Reimagining%20 RTGS.pdf.

31 "Segregated Witness" (so-called "SegWit") and "Relative Lock Time" that were first implemented in Elements were then adopted by other blockchains.

32 https://www.ecb.europa.eu/pub/pdf/other/ecb.stella_project_report_september_2017.pdf.

33 https://www.bundesbank.de/Redaktion/EN/Downloads/Press/Pressenotizen/2016/2016_11_28_block chain_prototype.pdf?__blob=publicationFile.

34 Furthermore, to obtain faster finality of settlements, an interval of block creation was also set to be short (e.g. two seconds) and each node was specified as a whitelisted peer. For more information, see Annex 5.

allows information to be shared on a need-to-know basis and participants have a part of the global ledger which they share on a "per-record" basis. This prevents any party from having a complete view of the information on the network unless explicitly designed. On the other hand, while Fabric provides channels which allow for data isolation and confidentiality, communication between the nodes relies on broadcasts within the same channel and the participants share a ledger on a "per-channel" basis. In Elements, it fully relies on global broadcasts and participants share one single ledger, and more complex cryptographic techniques may be needed to ensure an adequate level of privacy and confidentiality.

Data structure: Process flow of DvP arrangements is affected by the data model of each platform. Both Corda and Elements use a UTXO (Unspent Transaction Output) model to store data in their ledgers.[35] The process flow moves to the next step based on a transaction created by participants using unspent states. On the other hand, Fabric's ledger is comprised of a blockchain to store sequenced records in blocks, as well as a state database to maintain the current status as a collection of key-value pairs. In Fabric, although UTXO model can be applied, an account model is typically used and a process is represented as a state transition in the state database. The process flow proceeds to the next step based on a participant's instruction to update the status.

5 Analysis of DvP approaches on DLT

Based on the process flow described above, this section analyses the main characteristics of the single-ledger DvP approach and the cross-ledger DvP approach with HTLC. Specifically, it considers how each approach could potentially entail the following risks which are typically attached to a securities transaction.

Principal risk: If assets are exchanged between two counterparties, a risk could arise from one counterparty irrevocably delivering the asset while not receiving the asset it was due to receive.[36]

Replacement cost risk: Until a securities transaction is settled a counterparty faces the risk of incurring the cost associated with replacing the original transactions.

Liquidity risk: Liquidity risk (i.e. the risk that a counterparty will have insufficient funds to meet its financial obligations when due, but may be able to do so at some time in the future)[37] would arise in the case of settlement fails. Liquidity risk could also emerge depending on the way in which liquidity – either the cash leg or the securities leg in the DvP transactions – is available during the settlement process.

This section also touches upon other aspects of the DvP approaches, such as the speed of settlement, privacy, interdependency between the ledgers and

[35] The UTXO model is often said to be compatible with parallel processing because each transaction can use different input states. However, it should be noted that the benefit of it depends very much on how account balances are spread into multiple UTXOs.

[36] Principal risk and replacement cost risk are also referred to as credit risk, i.e. the risk that a counterparty will be unable to meet fully its financial obligations when due or at any time in the future; see Section 2.5 of PFMI.

[37] See Section 2.6 of PFMI.

infrastructure design. Legal aspects are, however, not part of the analysis and require further examination as to how the DvP transactions and the application of smart contracts in the DLT environment could be treated legally and how finality of settlement in the DLT environment could be achieved.[38]

5.1 Achievement of DvP

The exchange of two assets between the two counterparties inherently entails principal risk. In order to mitigate this risk, DvP arrangements ensure that the final settlement of one obligation occurs if, and only if, the final settlement of the linked obligation occurs. If one of the counterparties fails to fulfil any steps in the DvP process flow, resulting in settlement fail, the funds and securities should be returned to their original holders.[39]

For the single-ledger DvP arrangement, even if the steps in the process flow (see Section 4.1) are not completed, the counterparties would not be exposed to principal risk because the securities leg and the cash leg are executed only in the last step of the process and are simultaneously transferred as a single transaction.

For cross-ledger DvP with HTLC, the counterparties could be exposed to principal risk if some of the steps in the process flow are not completed. Specifically, if the participant does not retrieve securities before the locking time expires (Bank B in step 7 and 8 of Chart 4 in Section 4.2), securities could be returned to the other participant (Bank A in the given example) which has already received cash.[40] Although there are ways in which such a risk can be addressed,[41] its mitigation in essence depends on the participants' management of the timing of relevant processes.

---

38 There is currently no common legally defined moment for the finality of settlement in the DLT environment. Although DvP ensures that the delivery of securities occurs if and only if the corresponding payment occurs, there is uncertainty in terms of the finality of the DvP settlement in the DLT environment. Put differently, if the transfer in one of the ledgers (e.g. cash ledger) is not irrevocable and could be reversed after the transfer in the other ledger (e.g. securities ledger) is completed, one of the counterparties (e.g. seller of the securities) would be exposed to principal risk. Moreover, from an operational perspective, for some DLT platforms that rely on a proof-of-work consensus mechanism, the finality of settlement arrangements is probabilistic. Some platforms may also be subject to a hard fork, which is a rollback of a previously agreed transaction history after a certain point of time. On top of those, the legal effectiveness of the issuance and the transfer of dematerialised securities in a DLT environment could have an impact on possible DvP implementation.

39 Settlement fail scenario might materialise due to other reasons such as miscommunication between the counterparties or unintended failures during the process flow. The occurrence of settlement fail does not necessarily mean that the DvP arrangements in the DLT environment do not work.

40 In this particular scenario, DvP could fail because the completion of the 2nd cash transaction does not automatically trigger the 2nd securities transaction. In other words, in the cross-ledger approach using HTLC, the arrangement itself does not guarantee the linkage of the two transfers and it is the responsibility of each counterparty to execute the relevant processes within the predefined locking times.

41 See footnote 27.

5.2 Efficiency in the use of liquidity

Efficient use of liquidity in a settlement arrangement is influenced by many factors, including the settlement mode (i.e. gross or net settlement), the availability of intraday credit, and the timing at which participants send their instructions.[42] In the context of DvP, liquidity efficiency also depends on whether and for how long cash or securities will be locked during the process.

For the single-ledger DvP arrangement, as long as the securities leg and the cash leg are simultaneously transferred as a single transaction, neither of these legs is formally locked in advance. Liquidity efficiency would therefore not be hampered. On the other hand, locking securities and cash to prevent settlement fails could be considered (see step 4 of Chart 2 in Section 4.1). There consequently is a trade-off between liquidity efficiency and certainty of settlement.

By contrast, for the cross-ledger DvP arrangement with HTLC, locking of the assets in the early step of the whole processes (i.e. step 1 and step 3 of Chart 4 in Section 4.2 where the 1st securities instruction and the 1st cash instruction are created) is necessary for achieving DvP.[43] If securities and cash are not locked in step 1 and step 3, there is a possibility a securities transfer will fail in step 8 due to one participant having an insufficient amount of securities (Bank A in the given example), even if the cash transfer is completed in step 6. Therefore, liquidity efficiency under the arrangement is somewhat lower compared to the single-ledger DvP arrangement. Liquidity efficiency is especially low when the assets were locked, settlement is not successful and assets are returned to the original holder after expiration of the locking time.

5.3 Speed of settlement

In this study, the speed of settlement refers to the time interval between (i) when one of the counterparties creates the instructions for the transfer of securities or cash and (ii) when the transfers of both securities and cash are recorded on the ledgers.

For the cross-ledger DvP with HTLC, the process flow entails a higher level of complexity compared to single-ledger DvP as it entails more steps (verification and confirmation through the consensus mechanism). Moreover, time is required for each party to recognise the completion of the previous step and the preparation for the next one. Against this backdrop, the cross-ledger DvP with HTLC is likely to require more time.

To compare the speed of settlement between the single-ledger and the cross-ledger DvP arrangements, simulation exercises were conducted using different DLT platforms. Under all test arrangements, the completion of the entire process flow of

[42] Efficient use of liquidity could also depend on the order in which the relating transactions will be processed and settlement occurs in the consensus mechanism.

[43] See Annexes 4, 5 and 6 for more information regarding the different platforms experimental implementations.

the single DvP settlement was achieved in a time range of seconds (single digit). Cross-ledger DvP with HTLC took up to three times longer than the tests in the single ledger set-up.[44]

The speed of settlement is affected by (a) the time it takes for the two parties to communicate with each other to generate and sign a DvP transaction and (b) the time required for the single DvP transactions to be processed. It should be noted that in both the single-ledger DvP and the cross-ledger DvP with HTLC, most of the latency (in some cases up to around 97%) stems from the time it took for transactions to be verified and committed on the ledger, rather than the time it took for the two parties to communicate with each other and generate transfer instructions.[45]

5.4 Privacy

There are a variety of ways in which the visibility of transactions can be designed in the DLT platforms. For example, information is shared by all the relevant parties in Elements and Fabric, while it is shared only among the involved parties in Corda.[46]

For the single-ledger DvP approach, the two parties need to verify and sign a single transaction. Irrespective of the underlying DLT platform, such communication between the two parties is not committed on ledgers and thus no visibility is required in the verification and signing processes.

In contrast, under the cross-ledger DvP with HTLC, secret information may need to be propagated throughout the network. When the seller of the securities (Bank A in the given example) obtains cash and reveals the secret, this allows the buyer of the securities (Bank B) to receive the secret and obtain securities (step 7 of Chart 4 in Section 4.2). In this secret-sharing, the fact that Bank A has completed its asset-retrieving transaction needs to be shared with Bank B, although Bank B is not directly involved in that transaction. In other words, in order for the cross-ledger DvP with HTLC to work without intermediary, the secret used by Bank A may need to be

---

[44] The maximum number of DvP settlements that could be processed in a given time period for the single-ledger DvP was more than twice as large as that for the cross-ledger DvP. Note that the cross-ledger DvP requires two transactions for each of the two DLT networks.

[45] Performance highly depends on the environment and configurations of parameters. In our experiments, each node was run on a separate Ubuntu server (16.04.1 LTS 64bit), each with 7.5 GB of RAM and 8 GB of storage. Default parameters were basically used for all the three DLT platforms. Prototypes were designed to handle more than one type of securities. However, when measuring performance, only one type was used and the two DLT networks were created using the same platform. The time it took for the two parties to communicate with each other was minimal as two processes representing them were located on the same server. Regarding UTXOs, in Corda, the default transaction selection algorithm with "soft-locking" was used, while in Elements, inputs were directly specified not to use the same inputs among different transactions. In Fabric, a balance of an account is stored as a series of deltas in the state database in order not to use a single key to represent a balance. For further information, see Annexes 4, 5 and 6.

[46] In Corda, there is no single central store of data. Instead, each node maintains a separate database of known facts. As a result, each node only sees a subset of facts on the ledger, and no one is aware of the ledger in its entirety. In Elements, each node holds the same copy of data. In Fabric, each node holds the same copy of data on a per channel basis and an orderer could work as a single store of data across multiple channels.

shared throughout the network,[47] which may raise challenges depending on the privacy model of the DLT platforms.[48]

5.5 Interdependency between ledgers

Under the cross-ledger DvP with HTLC, there is no connection and interaction between the two ledgers and the two ledgers act independently. The operational reliability and performance of one ledger could nevertheless affect that of the other ledger.

Moreover, although the machine time of the two ledgers does not necessarily need to be perfectly aligned, it is important that (i) the flow of time in two ledgers is predictable in order to meet the need for asymmetry between the two types of locking time and (ii) the locking times are set fairly long enough so that they can accommodate possible differences between the ledgers. In this sense, the setting of the locking time is vital to ensuring both parties have sufficient time to interact during the predefined time period while bearing in mind that a longer locking time may contribute to a delay in returning the cash and securities to the original holders of those assets in the case of settlement fails.

5.6 Infrastructure design

While this study focuses on the DvP mechanism between cash and a single type of securities, market infrastructure as a whole comprises a wide variety of securities (e.g. government bonds, corporate bonds, equities) with multiple layers of post trade processes (e.g. confirmation, netting, clearing and settlement). Against this background, the choice of the DvP mechanism may need to be considered along with the overall design of the market infrastructure service.

For example, it might be efficient to put a range of different securities on a single ledger and perform DvP across different asset classes on the same platform. Following this approach, additional settlement functionalities (e.g. liquidity saving mechanisms, netting of outstanding and payable accounts, and efficient use of securities as collateral) could benefit from the fact that their implementation would only be required for a single DLT network. Furthermore, there could be benefits for the visibility and management of various asset balances at participant level since multiple asset classes would be held in one DLT network.

Such potential benefits may, however, have to be weighed against the stability of market infrastructure, such as the scalability, flexibility and resiliency of the platform.

---

47 The secret of Bank A needs to be shared with Bank B. If Bank A does not share it with Bank B, a trusted entity who correctly shares it with Bank B on behalf of Bank A is required. This means a trusted entity is required to know the identities of the involved parties. Another approach that does not use a trusted entity could be a broadcast to the network and the secret can be visible to all participants. Note that even if all participants know the secret, only Bank B who is specified in HTLC can retrieve the asset.

48 See Annex 4 for more details.

A single ledger platform with multiple assets may require high scalability because the overall number of transactions to be settled is higher than one with only two types of assets. Transactions for certain assets with high volumes may occupy most of the resources of the network, which in turn could cause a congestion/delay of transactions for other assets. Cyber security measures for each node in the network will be particularly important to ensuring high availability, as the transaction volume usually increases with the number of assets and participants.[49]

For the cross-ledger DvP with HTLC, one obvious benefit of this approach in terms of infrastructure design is that DvP can be achieved between any ledgers under the responsibility of the participants without creating any connection between the ledgers. Theoretically, as long as HTLC is supported by the two ledgers[50] and the two parties participate in both ledgers,[51] DvP can be achieved without developing institutional arrangements or operational procedures between the two ledgers. This study confirmed that the cross-ledger DvP with HTLC could work between two separate DLT networks which can use either (i) the same DLT platform or (ii) different DLT platforms (e.g. an unrestricted DLT network using Elements and a restricted DLT network using Fabric).

[49] In addition to the challenges described above, if existing systems are not interoperable with or integrated into the single DLT network, it could lead to the distribution of cash balances on multiple systems, thereby contributing to the inefficient use of liquidity. Furthermore, a fragmentation of securities holdings on several systems might have negative effects on other securities services such as securities lending and collateral management.

[50] Non-DLT ledgers are also included since HTLC is a type of smart contract and the use of it is not necessarily linked to other components of DLT, such as peer-to-peer networks and consensus algorithms.

[51] It is technically possible to introduce more than two parties to a single DvP transaction.

Annex 1: DvP in Bank of Japan and ECB services

The ECB and the BOJ both have responsibilities in the operation of the securities settlement platforms enabling the safe and efficient settlement of securities against central bank money in the respective markets. The performance of these systems is closely interlinked with the fields of monetary policy and financial stability.

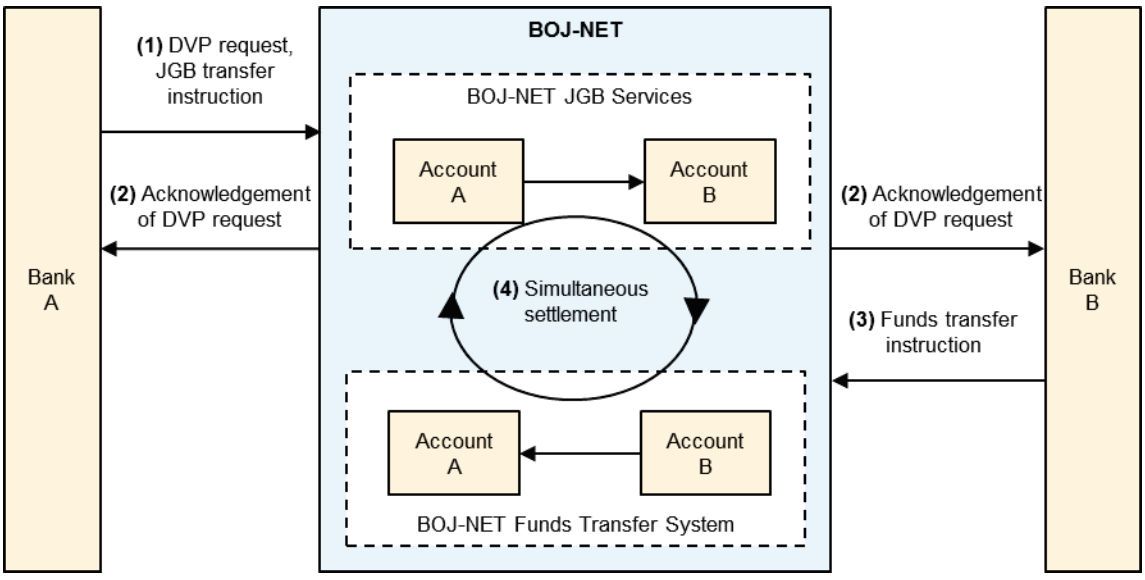
BOJ-NET JGB Services

BOJ operates the BOJ-NET JGB Services for book-entry transfer of Japanese government bonds (JGBs) and the BOJ-NET Funds Transfer System for fund transfers for current account deposits at BOJ. DvP is achieved on a single BOJ-NET platform by executing the transfers of JGBs and funds simultaneously.

Specifically, this is conducted in the following steps (see Chart A): (i) the deliverer of JGBs (Bank A) sends to BOJ-NET the DvP request (containing both JGB and funds transfer information) and the transfer instruction for JGBs; (ii) BOJ-NET sends acknowledgement of DvP request to both the deliverer of JGBs (Bank A) and the receiver of JGBs (Bank B); (iii) the receiver of JGBs (payer of funds) confirms the content of the DvP request and sends the transfer instruction for funds; (iv) BOJ-NET checks the balances of both participants and executes the transfers of JGBs and funds simultaneously if sufficient amounts of securities and cash are available in respective participants.

Chart A

Process flow of BOJ-NET JGB Services

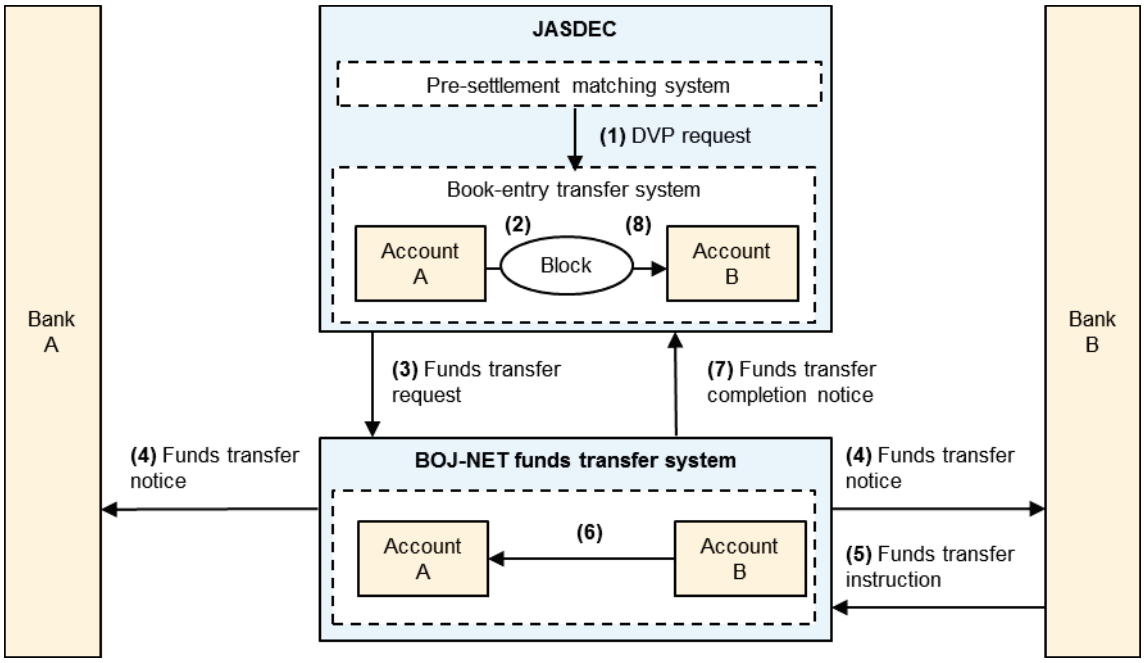


DvP link between BOJ-NET Funds Transfer Service and JASDEC Book-Entry Transfer System

Japan Securities Depository Center (JASDEC) operates the book-entry transfer system for corporate bonds and commercial paper. DvP is achieved by linking the JASDEC Book-Entry Transfer System with the BOJ-NET Funds Transfer System, with JASDEC blocking the securities to be delivered until funds transfers in the BOJ-NET is completed. Specifically, this is conducted in the following steps (see Chart B): (i) JASDEC Pre-Settlement Matching System generates and sends the DvP request to the JASDEC Book-Entry Transfer System; (ii) JASDEC blocks the securities to be delivered (balances for Bank A); (iii) JASDEC sends a request for funds transfers to BOJ-NET; (iv) BOJ-NET notifies the receipt of JASDEC message to the deliverer of securities (Bank A) as well as the receiver of securities (Bank B); (v) the receiver of securities sends a funds transfer instruction to BOJ-NET; (vi) BOJ-NET executes the funds transfer; (vii) BOJ-NET notifies JASDEC of the completion of the funds transfer; (viii) JASDEC releases the previously blocked securities and transfers them to the account of the receiver of the securities.

Chart B

Process flow of DvP link between BOJ-NET Funds Transfer System and JASDEC Book-Entry Transfer System

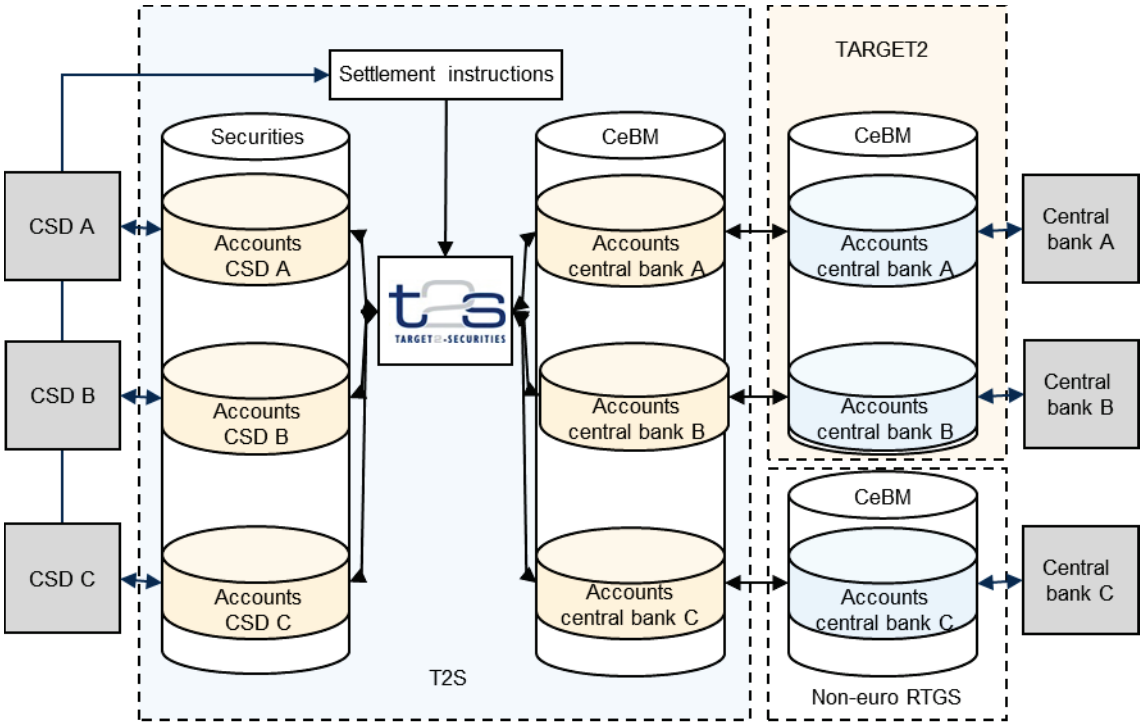


TARGET2-Securities (T2S)

TARGET2-Securities (T2S) is a single, pan-European platform for securities settlement in central bank money and is owned and operated by the Eurosystem. The T2S platform offers central securities depositories (CSDs) and their participants securities settlement in central bank money, cross-border and domestic settlement of securities in an efficient and safe manner. T2S accommodates market participants' securities accounts, held at one or more central securities depositories (CSDs), and their dedicated central bank cash accounts. CSDs keep their clients' positions in T2S, each securities account is attributable to a single CSD and each cash account is assigned to a single central bank. Dedicated cash accounts are linked to market participants' main cash accounts in TARGET2 or another non-euro real-time gross settlement account. The use of an "integrated model" allows T2S to connect any securities account at any participating CSD with any cash account at any participating central bank. T2S settlement provides both on a continuous basis during the daytime and in sequences grouped into a predefined number of cycles during the night-time.

Chart C

Process flow of TARGET2-Securities (T2S) Settlement Instruction Lifecycle / Infrastructure overview



Annex 2: Process fail scenarios

This section provides potential settlement fail scenarios in each step of the process flow of both the single-ledger DvP and the cross-ledger DvP with HTLC. Subsequent descriptions are based on Chart 2 of Section 4.1 and Chart 4 of Section 4.2 for the single-ledger DvP and the cross-ledger DvP with HTLC respectively.

Failures of the process are focused in this annex. However, it should be noted that the operational reliability of DLT networks could cause other types of failures.

Single-ledger DvP

Settlement could fail in any of the following possible scenarios. In all scenarios, the cash and securities will be kept by the original holders;

a) If the process is suspended in step 1 (e.g. Bank A fails to create an instruction), the instruction is not submitted to the consensus mechanism and no update is committed on the ledger.

b) If the process is suspended in step 2 (e.g. Bank B finds that the instruction of Bank A has errors), the instruction is not submitted to the consensus mechanism and no update is committed on the ledger.

c) If the process is suspended in step 3 (e.g. Bank A finds that the instruction of Bank B has errors), the instruction is not submitted to the consensus mechanism and no update is committed on the ledger.

d) If the process is suspended in step 4 (e.g. Bank A tampers with the instruction of Bank B, either Bank A or Bank B does not have a sufficient amount of securities or cash if assets are not locked when an instruction is created), the instruction is rejected and no update is committed on the ledger.

Cross-ledger DvP with HTLC

Settlement could fail in any of the following possible scenarios;

a) If the process is suspended in step 1 (e.g. Bank A fails to create the 1st securities instruction), the instruction is not submitted to the securities consensus mechanism and no update is committed on the ledgers in both the securities and cash DLT networks.

b) If the process is suspended in step 2 (e.g. Bank A sends the 1st securities instruction but does not have sufficient amount of securities to be locked), the instruction is rejected and no update is committed on the ledgers in both the securities and cash DLT networks.

c) If the process is suspended in step 3 (e.g. Bank B finds that the instruction of Bank A has errors and therefore does not create the 1st cash instruction), since

the Bank A's agreed amount of securities is locked by the 1st securities instruction, Bank A can submit the 2nd securities instruction (for refunding of the agreed amount of securities) to the securities consensus mechanism after the locking time expires (two hours). Bank B cannot obtain the securities since he does not know X. No update is committed on the ledger in the cash DLT network.

d) If the process is suspended in step 4 (e.g. Bank B sends the 1st cash instruction but does not have sufficient amount of cash to be locked), the instruction is rejected and no update is committed on the ledger in cash DLT networks. In this case, Bank A cannot confirm the 1st cash instruction even after the locking time has expired (one hour). As with scenario c), Bank A can submit the 2nd securities instruction (for the refunding of the agreed amount of securities) to the securities consensus mechanism after the locking time expires (two hours).

e) If the process is suspended in step 5 (e.g. Bank A finds that the instruction of Bank B has errors), Bank A and Bank B can submit the 2nd securities instruction and the 2nd cash instruction (for refunding of the agreed amount of securities and cash) after the locking time expires (two hours and one hour, respectively).

f) If the process is suspended in step 6 (e.g. Bank A provides the wrong value as X although its retries for one hour), same as scenario e), Bank A and Bank B can submit the 2nd securities instruction and the 2nd cash instruction (for refunding of the agreed amount of securities and cash) after the locking time expires (two hours and one hour, respectively).

g) If the process is stopped in step 7 (e.g. Bank B does not submit the 2nd securities instruction within two hours), Bank A can submit the 2nd securities instruction (for refunding of the agreed amount of securities) to the securities consensus mechanism after the locking time expires (two hours).

   In this scenario, Bank A can receive cash and at the same time refund the securities that it was planned to deliver to Bank B.

h) If the process is stopped in step 8 (e.g. Bank B provides the wrong value as X although its retries for two hours), same as in scenario G), Bank A can submit the 2nd securities instruction (for refunding of the agreed amount of securities) to the securities consensus mechanism after the locking time expires (two hours).

   In this scenario, Bank A can receive cash and at the same time refund the securities which were planned to be delivered to Bank B.

Annex 3: Unspent Transaction Outputs (UTXO)

As known to most readers, DLT platforms are based on append-only data structure that is used to store, in an immutable order, all updates to the ledger. These updates, which are called transactions, are considered as confirmed by DLT network participants as soon as they are provided by a consensus mechanism.

The term transaction has a very specific meaning in the area of DLTs: transactions are composed of inputs and outputs, which constitute new unspent transaction outputs (UTXOs) as explained in the following paragraphs. UTXOs are used in the blockchain data structure of Elements but are also a key element of the Corda platform, which provides for their sequential ordering. Given the relevance of UTXOs for these and other major DLT platforms, some details on their functioning are provided in this box.

UTXOs are pieces of digital information representing any number of assets (e.g. securities) resulting from a valid transaction in the distributed ledger. Although UTXOs are created through a process that is specific to every DLT platform and the assets they are meant to represent, what follows is meant to provide a general description of the process.

Participants in a DLT network agree to allow any participant owning an UTXO to either transfer it in favour of another peer or to amend its attributes. As soon as one or more UTXOs are sent to another participant, they become inputs of such a transaction and, once the transaction is valid, are “consumed” in the sense that they cannot be spent any longer. The consumption of these inputs is necessary to allow the creation of a set of new UTXOs representing, in general, an equal amount of the assets represented by the consumed input. This process guarantees that only the unspent holdings, as reflected in the latest update of the ledger, can be used to make new transactions.

Each transaction has a unique transaction ID**.** Unlike traditional databases, the transaction ID in UTXO-based distributed ledgers is not assigned by the validator but is calculated and signed by each participant – for instance, by hashing the metadata of the transaction.

Chart D

Anatomy of a transaction

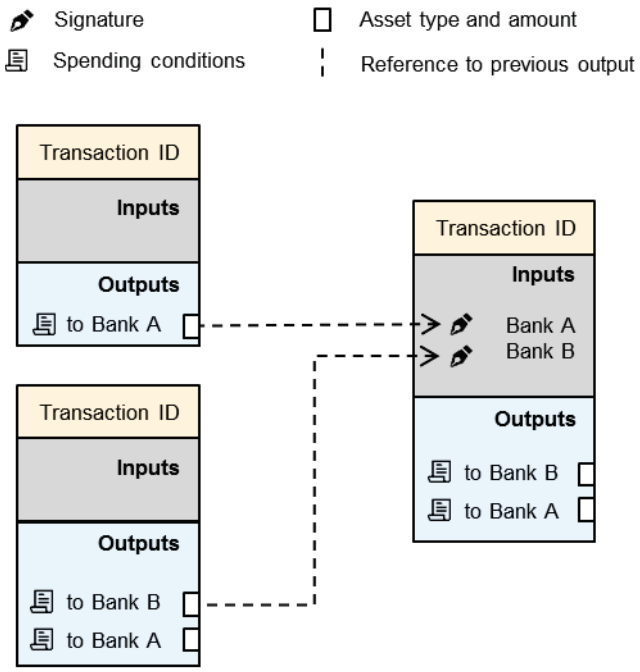
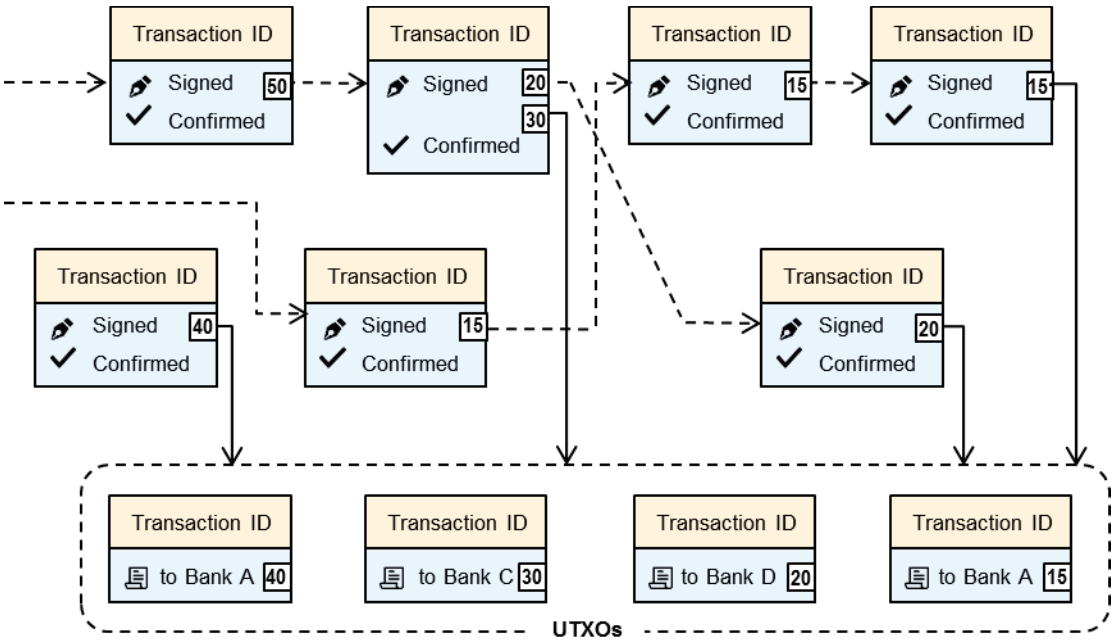


Chart E

From transactions to the unspent transaction outputs set



At any given time, the ledger is in a state that is given by the set of the output of transactions that are unspent – also called the Unspent Transaction Output Set. From this set, the service in charge of validating transactions can make sure that the same unit of asset is stored only once on its books.

As highlighted in Chart E, most of the transactions that are stored in the ledger have spent outputs, i.e. they are referenced by other confirmed transactions. There are some transactions with unspent outputs. These outputs represent the state of the ledger at a given point in time. They are used to calculate balances of wallets (e.g. the balance of Bank A wallet is 55 units of securities) and used as a starting point to create new transactions.

The information comprised on the ledger can be broken down into a chain of transactions that can be validated independently from each other by smart contracts.

UTXO in the case of a blockchain-based DLT system such as Element

To spend/destroy tokens, each input of a transaction is composed of the following:

Reference to a previous output: This is usually a reference to a confirmed unspent transaction output. The reference to a previous output is made up of the ID of the previous transaction and a non-negative number that identifies the output in the list of outputs of the referenced transaction (e.g. 0 for the first, 1 for the second).

Witness: This is a piece of information necessary for the transaction creator to prove that he has the right to redeem the referenced output. This piece of information usually includes signatures, but sometimes also includes other information such as secrets.

To create new tokens, transactions have outputs, each composed of the following:

Asset type and amount: Transactions may transfer ownership of cash, stocks, bonds or other assets and the type is specified in the output. The value of the newly generated token is specified in the output as a number. New tokens must have the same asset type as the spent ones and the total value of the newly generated tokens must be less than that of the spent ones.

Spending conditions: This is a commitment that describes the condition that a future transaction must fulfil in order to redeem this output. Simple spending conditions can be "the spending transaction must provide a signature and the public keys associated to the hash of public key" and more complex smart contracts are the "Hashed Timelock Contracts" described in this paper.

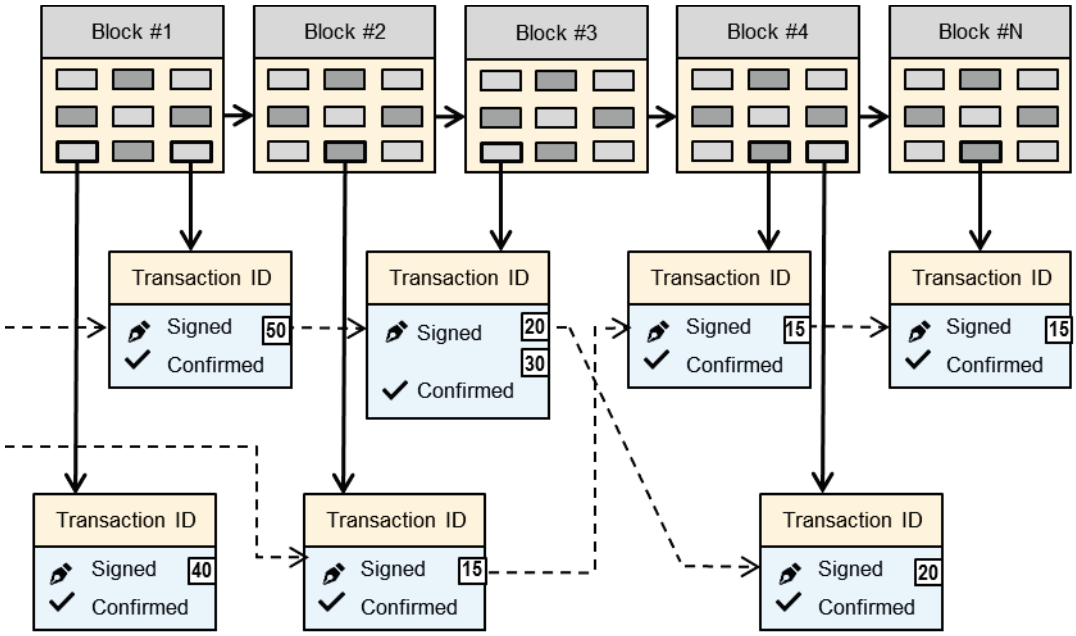
Finally, transactions have also other fields that are not illustrated in the picture for sake of simplicity. These include the following:

Fee: The total amount of new tokens is usually less than the total amount of the destroyed tokens. The difference is a fee that is rewarded to a miner that first creates a block containing the current transaction.

Time locking fields: This prevents the transaction from being confirmed before a timeout has expired. Timelocks can be absolute or relative. An absolute timelock prevents a transaction from being confirmed before a specific date (e.g. a transaction cannot be confirmed before 1 September 2018). A relative timelock prevents a transaction from being spent before some time has passed from the confirmation time of the input.

Chart F

From blocks to transactions



A transaction is considered to be valid if, for each input, the provided witness matches with the spending conditions of the referenced output. Furthermore, if all the outputs referenced by a new transaction are still unspent, i.e. they are not referenced by any other transaction already confirmed, then the transaction can be included in a block and the state of the ledger can be updated accordingly by all nodes.

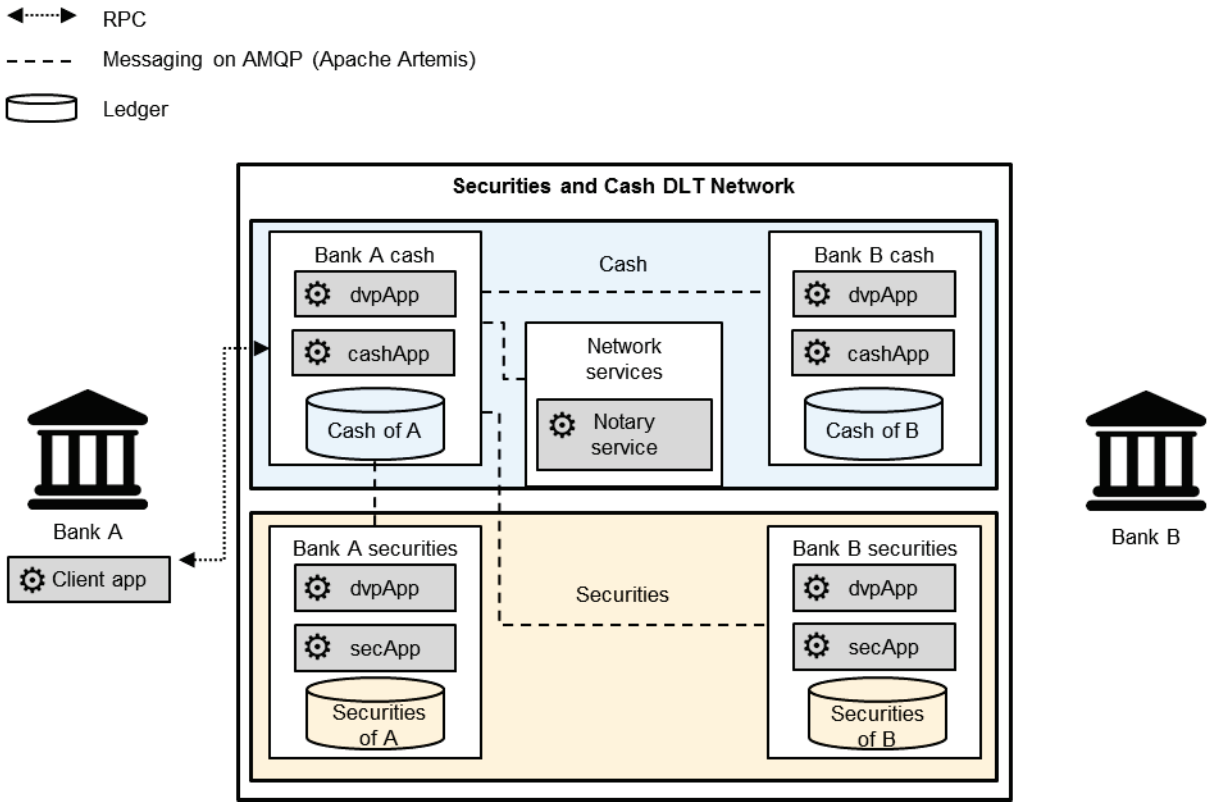
Annex 4: Process flow on Corda

Network setup

In a Corda network, nodes can run multiple applications (or CordApps) managing different classes of assets. The chosen network in Stella included four nodes: two managing cash and two managing securities with one validating notary service. The cash and securities nodes for each bank were dissociated to better model the separation of identities in a post trade settlement. Additionally these nodes operate a DvP flow that enables swapping of cash against securities coordinated between the two cash and the two securities managers. Assets are represented by states and DvP execution happens with the execution of a single flow and an atomic ledger commit transaction. In the Stella experiment, the "soft-locking" mechanism which prevents a node constructing transactions to use the same input states simultaneously was enabled by default; however, the lock could be released if the transaction is pending for too long.

Chart G

Network for single-ledger DvP



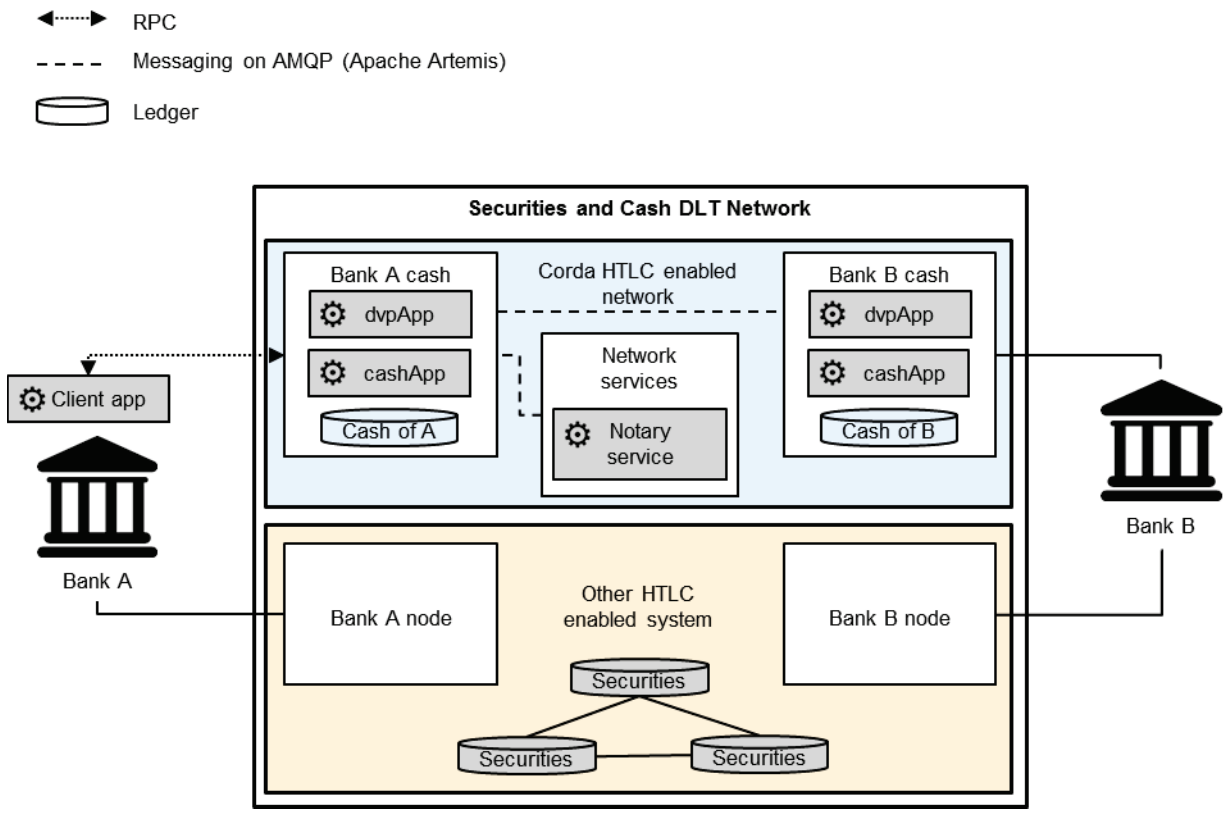
During our test with this single-ledger system, four Corda nodes run within the same network, one special node runs a validating notary service along with doorman and network map services, and each node runs in an individual Java Virtual Machine. Additionally we install applications on these nodes to enable them to transact on the different assets (cashApp and secApp) and enable DvP exchanges (dvpApp). The client was developed in Kotlin and uses the standard RPC library for Corda.

Before initiating the DvP exchange, assets were self-issued on the cash node of the buyer (Bank A) and the securities node of the seller (Bank B), either in states of large or in small amounts. This induces a different splitting requirement to select inputting and create the outputting UTXOs used in the exchange transactions.

Since a Corda network can interface with another Corda network[52], the following diagram illustrates the possibility for a network of cash using Corda nodes to process DvP transactions through an HTLC across two separate networks.

Chart H

Network for cross-ledger DvP with HTLC[53]



[52] In Corda, protocols exist to swap notary services and thus enable the exchange of assets between two different networks. This type of scheme was not studied during the course of our investigation.

[53] This scenario was, however, not developed during this study.

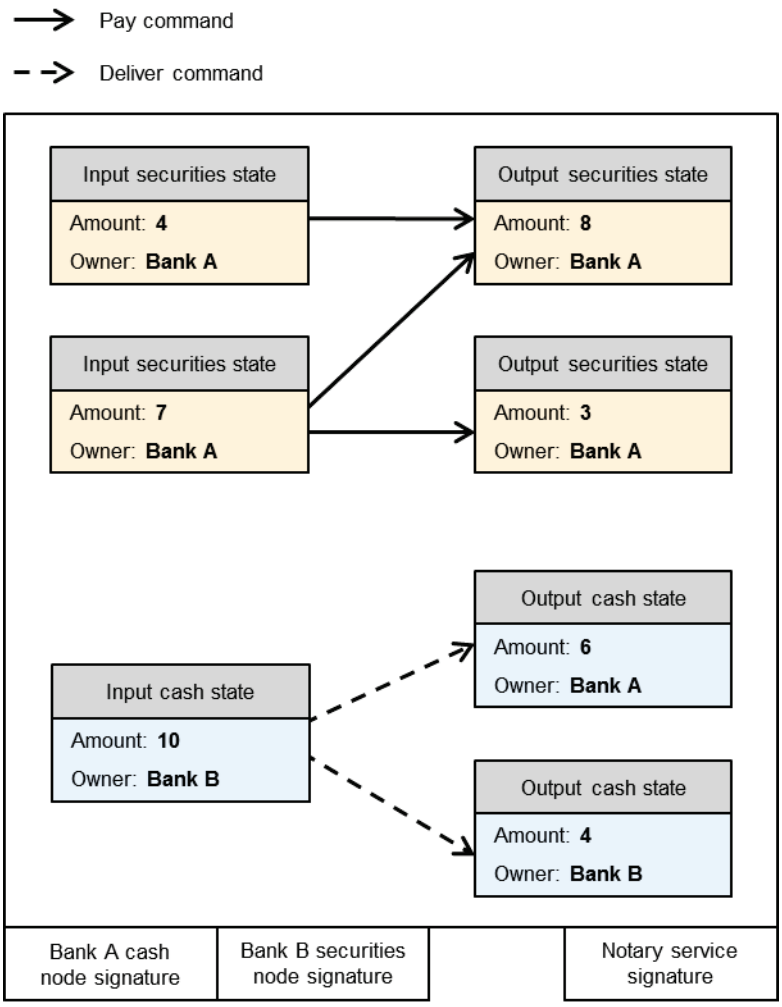
Single-ledger DvP

The Cash implementation provided in Corda was reused (enabling the exchange of two fungible assets between four nodes). In the set-up, two sets of nodes exchange on the one side cash states while the other exchanges securities states. This full process is encoded in one flow initiated by one of the nodes through a call from a client application. DvP execution happens with the execution of a single flow and the atomic commits to the ledger of the transaction.

In the experiment participants were registered within the same network map service; the initiator of the flow could thereby acquire the identity of its counterparts through standard Corda discovery instructions. This subsequently generates a spending instruction in the direction of its counterpart in the same asset class. The other participant is then instructed to carry out a similar preparation of the transaction, and returns to the initiator the inputs and outputs states corresponding to this exchange. The initiator of the flow then builds a complete transaction using all the input and output states. This can then be verified and signed by both. The complete transaction is submitted to the notary service in charge of validating and appending its UTXO ledger, and signing the transaction. The full preparation process is executed inside the nodes through smart contracts; this is a key difference from other DLT platforms.

In Corda, the node submitting the transaction to the notary service is also in charge of distributing the outcome of the transaction to the other participants once the transaction has been validated. From a privacy point-of-view, this process ensures that the notary service is not aware of the addresses of all participants in a trade. It also renders the full process atomic to the extent in which the notary service commits or rejects the full transaction atomically; it is ensured that no participant can spend the new UTXOs while others are unable to.

Chart I
Transaction for single-ledger DvP in Corda



The single-ledger transaction is signed by both participants inputting UTXOs in the transaction and is also signed by the notary following the finalisation of the transaction.

Cross-ledger DvP with HTLC

In order to enable HTLC in a Corda network, the experiment distinguished between two flows: one instigates the HTLC; it generates the secret and the transaction, placing a lock on the states through notarisation. The responder in this flow is in charge of redeeming the states and finalising the transaction once it has obtained the secret from another system. Similar to this, a subsequent flow follows an HTLC process started on another system; it is responsible for generating a transfer transaction corresponding to the agreed amount, including the secret and posting the

locked states to the notary service. In a second phase the responding flow is finalised by unlocking the states, which is carried out by the participant providing the secret from another leger.[54]

In this scenario, the notary service is responsible for locking the states and may also be involved in the secret sharing scheme; it must account for time validity of the instructions. States are indeed locked (or encumbered) on its books and must be released by the receiving participant when the secret is provided. In this case such a service must run validations on the transaction and release the states only if the time-lock limits are met, the secret is correct and signatures are provided. Different variations can be envisaged in Corda to mitigate the limitation of the time-lock. For instance, the release condition of the states can be attached to the signature of both participants in an instruction to the notary service to enable better use of liquidity, for example.

Given the need-to-know basis used as privacy model in Corda, the responding flow is most relevant for the secret needs to be shared to the other participant in the trade. This has to happen in such a way that the initiator of the responding flow can block the completion of the process only after the timelock has passed. Additionally the notary service must not know the identity of the receiving participant. Different approaches could be used to disseminate secret information, either through a trusted third party such as an oracle or via the notary distributing the finalised transaction to the participants, or the receiving participant being able to query the notary service.

[54] A Corda HTLC transfer would only be comprised of one of these two flows, although both participants would need to run either an initiator or a follower sequence.

Annex 5: Process flow on Elements

Network setup

Chart J

Network for single-ledger DvP

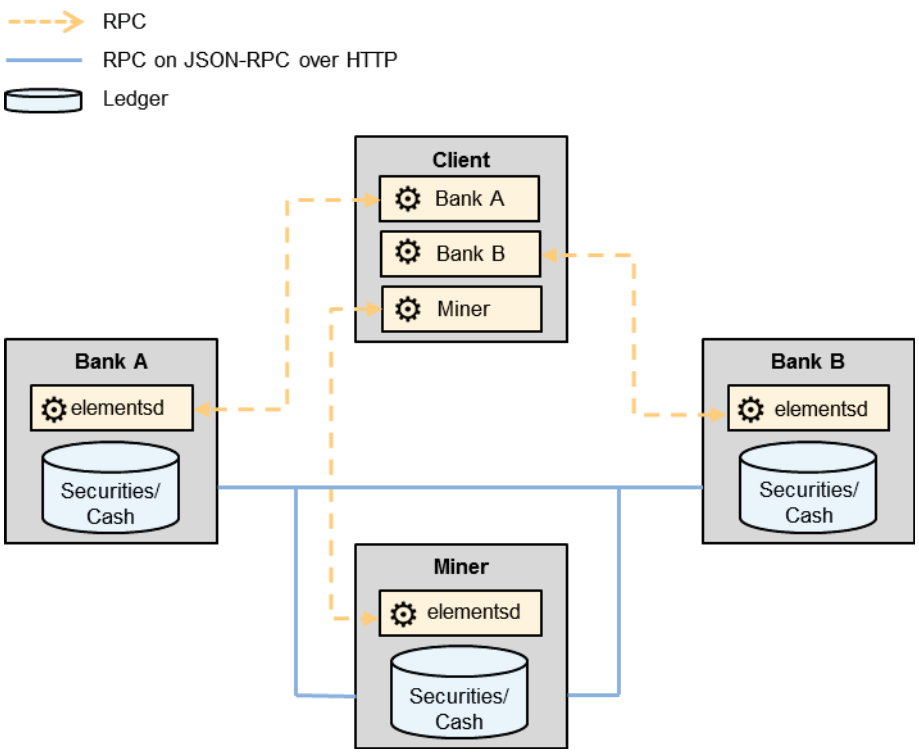
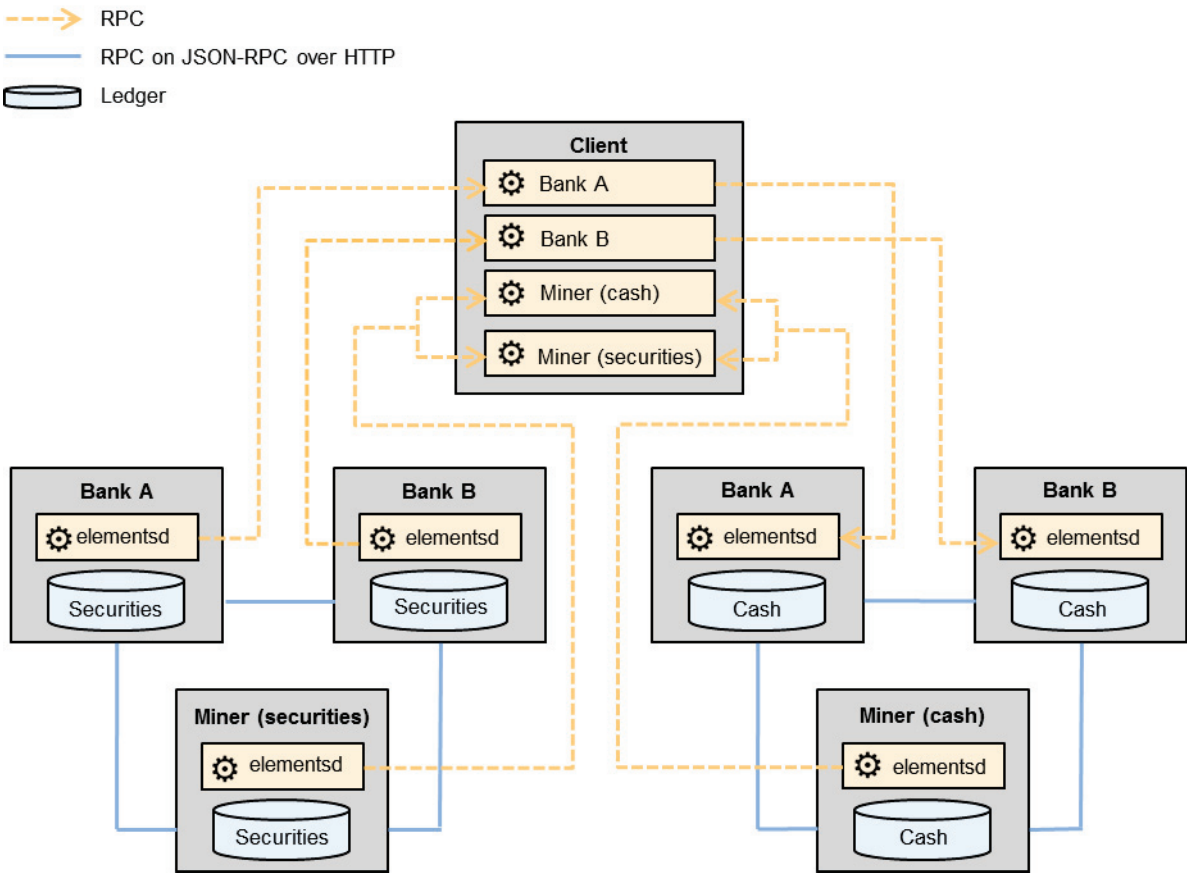


Chart K
Network for cross-ledger DvP with HTLC



Two processes on a single client server represent Bank A and Bank B respectively. A miner creates a block at a predefined interval.[55] A sufficient number of UTXO was created beforehand so that each transaction to be created in the subsequent processes does not use the same inputs.

In the subsequent description, it is assumed that (i) Bank A is the seller of securities and Bank B is the buyer of securities, and (ii) Bank A initiates the process. Please note that other implementation designs could be possible.

Single-ledger DvP

Preparation: Bank A and Bank B agree to the "amount" and the "asset type" to be exchanged. They share their own receiving addresses with each other.

[55] In order for transactions to be relayed promptly, each node was specified as a whitelisted peer. Transactions were set to be processed without a fee.

Step 1: Bank A (original holder of the securities) creates the securities transaction which specifies the unspent transactions that have the agreed amount of securities as inputs and the address of Bank B as an output.[56] Bank B (original holder of cash) creates the cash transaction which specifies the unspent transactions that have the agreed amount of cash as inputs and the address of Bank A as an output.

Step 2: Bank A sends his part of the transaction (securities transaction) to Bank B without his signature. Bank B verifies the content of the securities transaction (e.g. amount, asset type). If Bank B finds that it has errors, he stops the process. Otherwise, Bank B combines that securities transaction with the cash transaction of his own, thereby creating a full set of transactions. Bank B signs his part of the transaction (cash transaction) and sends it back to Bank A.

Step 3: Bank A verifies the full transaction. If Bank A finds that it has errors, he stops the process. Otherwise, Bank A signs his part of the transaction (securities transaction) and broadcasts it to the DLT network.[57]

Cross-ledger DvP with HTLC

Preparation: Bank A and Bank B agree to the "amount", the "asset type", two types of the "locking time" and the "cryptographic hash function" to be exchanged. They share their own public keys with each other. In the subsequent descriptions, it is assumed that the hash function is a double SHA256 (OP_HASH256) and locking times are two hours for Bank A and one hour for Bank B.

Step 1: Bank A (original holder of the securities) generates a secret (X) and a hash of it (Y). Bank A shares Y with Bank B. They both create the following scripts (script 1 and script 2) and obtain addresses of the respective scripts. Bank A adds the address of script 2 to his wallet as watch-only address, while Bank B adds both addresses to his wallet as watch-only addresses.

[56] Descriptions about "fee" and "change" are omitted. Each transaction specifies different inputs.

[57] For the rest of the flow, see footnote 22.

Chart L

Two scripts for the securities and the cash DLT network

Script 1 (for the securities DLT network)

OP_DEPTH 2 OP_EQUAL
OP_IF
OP_HASH256 <Y> OP_EQUALVERIFY <Public key of Bank B>
OP_ELSE
<Current time + 2 hours> OP_CLTV OP_DROP <Public key of Bank A>
OP_ENDIF
OP_CHECKSIG

Script 2 (for the cash DLT network)

OP_DEPTH 2 OP_EQUAL
OP_IF
OP_HASH256 <Y> OP_EQUALVERIFY <Public key of Bank A>
OP_ELSE
<Current time + 1 hours> OP_CLTV OP_DROP <Public key of Bank B>
OP_ENDIF
OP_CHECKSIG

Bank A creates the 1st securities transaction which specifies the unspent transactions that have the agreed amount of securities as inputs and the address of script 1 as an output. Bank A then signs it and broadcasts it to the securities DLT network.

Step 2: After the 1st securities transaction is confirmed,[58] Bank B (original holder of the cash) verifies the content of it. If Bank B finds that it has errors, he stops the process. In this case, Bank A creates the 2nd securities transaction (refunding of the agreed amount of securities), signs it and broadcasts it to the securities DLT network after the locking time expires (two hours). Otherwise, Bank B creates the 1st cash transaction which specifies the unspent transactions that have the agreed amount of cash as inputs and the address of script 2 as an output. Bank B then signs it and broadcasts it to the cash DLT network.

Step 3: After the 1st cash transaction is confirmed, Bank A verifies the content of it. If Bank A finds that it has errors, he stops the process. In this case, Bank A and Bank B create the 2nd securities transaction and the 2nd cash transaction (refunding of the agreed amount of securities and cash), sign them and broadcast them after each locking time expires (two hours and one hour) respectively. Otherwise, Bank A creates the 2nd cash transaction (obtaining of the agreed amount of cash) which specifies the 1st cash transaction as an input and a new address of Bank A as an output. It also provides X, the secret key associated with the public key of Bank A specified in script 2 and script 2 itself. Bank A signs it and broadcasts it to the cash DLT network.

Step 4: Bank B obtains X specified in the committed 2nd cash transaction of Bank A. Bank B then creates the 2nd securities transaction (obtaining of the agreed amount

[58] If forks do not occur, Bank B actually does not need to wait until the 1st securities transaction is confirmed.

of securities) which specifies the 1st securities transaction as an input and a new address of Bank B as an output. It also provides X, the secret key associated with the public key of Bank B specified in script 1 and script 1 itself. Bank B signs it and broadcasts it to the securities DLT network.

Annex 6: Process flow on Fabric

Network setup

Chart M

Network for single-ledger DvP

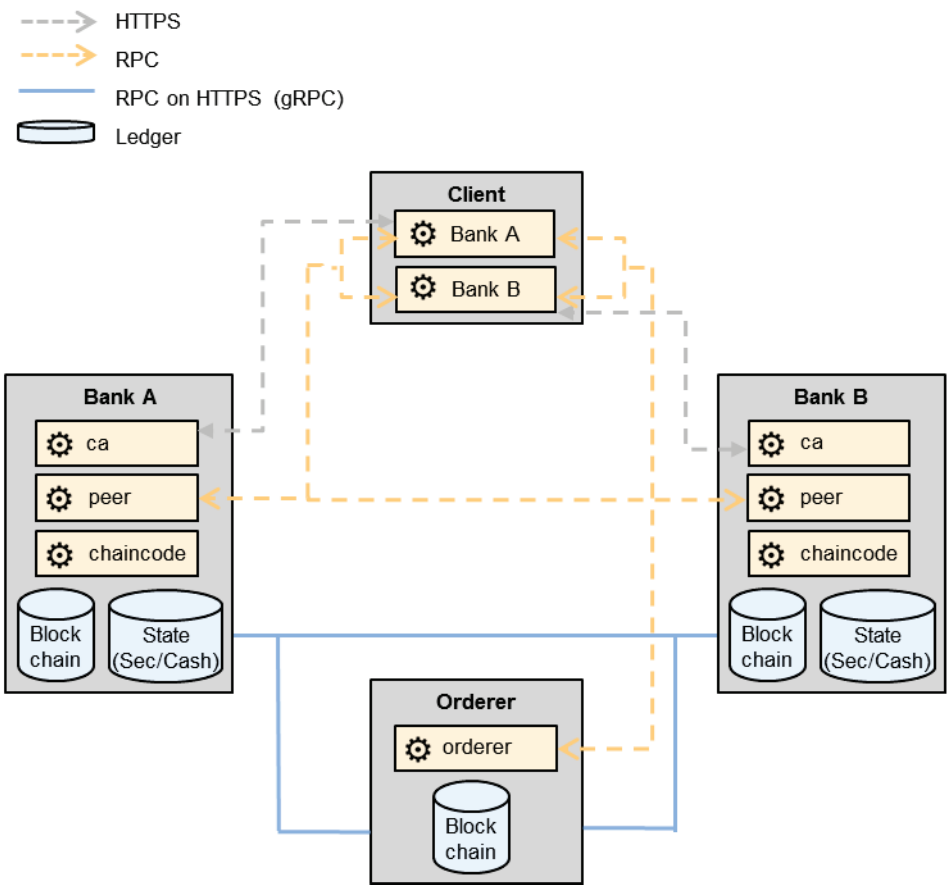
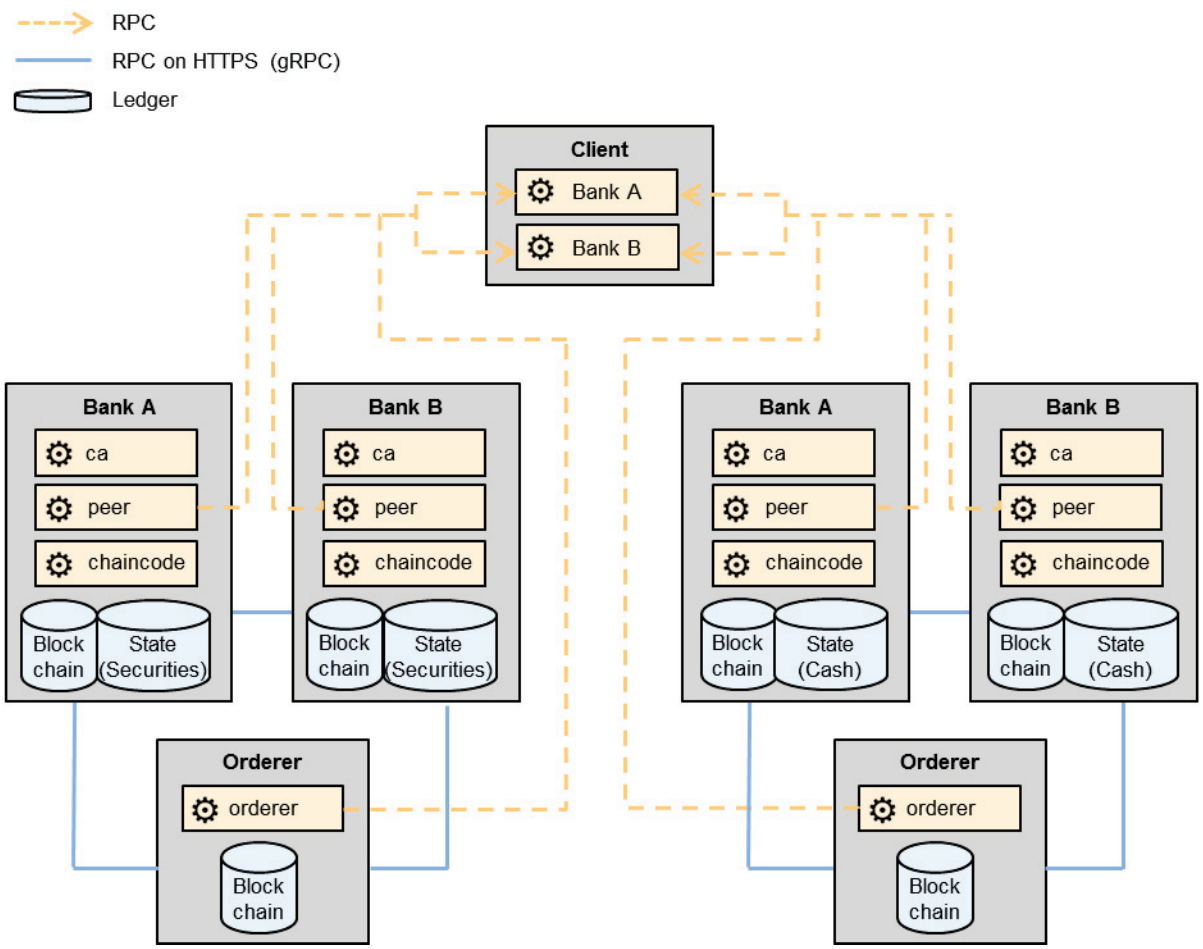


Chart N

Network for cross-ledger DvP with HTLC



Connections between clients and Certification Authorities were omitted to simplify this chart.

Two processes on a single client server represent Bank A and Bank B respectively.[59] Each network is comprised of (i) one orderer and (ii) two organisations, each of which has (a) one peer that also works an endorser and (b) one CA (Certificate Authority). The endorsement policy of chaincodes was set to AND condition i.e. only if the outputs of Bank A and Bank B are the same, the transaction is committed on the ledger. The balance of each bank was stored as a series of deltas in the state database in order not to use a single key to represent a balance.[60]

[59] Hyperledger Fabric Client SDK for Go was used in this study.

[60] In the state database, a composite key that consists of an account name, an asset type, an operation (i.e. plus or minus), an amount and a transaction ID was used as a key to store a delta. For further information, refer to https://github.com/hyperledger/fabric-samples/tree/release/high-throughput.

In the subsequent description, it is assumed that (i) Bank A is a seller of securities and Bank B is a buyer of securities, and (ii) Bank A initiates the process. Please note that other implementation designs could be possible.

Single-ledger DvP

Preparation: Bank A and Bank B agree to the "amount" and the "asset type" to be exchanged.

Step 1: Bank A (original holder of the securities) creates the 1st instruction which specifies the two transfers, i.e. Bank A delivers the agreed amount of securities to Bank B and in return Bank B pays the agreed amount of cash to Bank A. Bank A submits the 1st instruction to the consensus mechanism.[61] The chaincode stores the 1st instruction in the state database.

Step 2: Bank B queries the state database and verifies the content of the 1st instruction of Bank A. If Bank B finds that it has errors, he stops the process. Otherwise, Bank B creates the 2nd instruction which allows the two transfers to be executed and submits the 2nd instruction to the consensus mechanism.[62] The chaincode executes the two transfers and disables/removes the 1st instruction stored in the state database.

Cross-ledger DvP with HTLC

Preparation: Bank A and Bank B agree to the "amount", the "asset type", two types of the "locking time" and the "cryptographic hash function" to be exchanged. In the subsequent descriptions, it is assumed that the hash function is a double SHA256 and locking times are two hours for Bank A and one hour for Bank B.

Step 1: Bank A (original holder of the securities) generates a secret ($\boldsymbol{X}$) and a hash of it ($\boldsymbol{Y}$). Bank A shares $\boldsymbol{Y}$ with Bank B.[63] Bank A creates the 1st securities instruction which specifies two states in which (i) the receiver of the securities will be Bank B if Bank B provides $\boldsymbol{X}$ which satisfies $\boldsymbol{Y} = \mathrm{H}(\boldsymbol{X})$, or (ii) it will be Bank A if two hours pass. Bank A submits the 1st securities instruction to the securities consensus mechanism. The chaincode subtracts the specified amount of securities of Bank A and stores the 1st securities instruction in the securities state database.

Step 2: After the 1st securities instruction is committed, Bank B (original holder of the cash) queries the securities state database and verifies the content of the 1st securities instruction of Bank A. If Bank B finds that it has errors, he stops the

---

61 For the rest of the flow, see footnote 22.

62 In this study, the identity verification was not fully implemented.

63 Another approach could be Bank B obtains $\boldsymbol{Y}$ from the 1st securities instruction in step 2 either by using (i) the transaction ID of the 1st securities instruction shared by Bank A in step 1, or (ii) the content of the 1st securities instruction itself if multiple instructions with exactly the same content do not need to be differentiated. Please note that it would be better to avoid storing timestamps in the state database since the time of each endorser is not exactly aligned.

process. In this case, Bank A creates the 2nd securities instruction (refunding of the agreed amount of securities) and submits it to the securities consensus mechanism after the locking time expires (two hours).[64] Otherwise, Bank B creates the 1st cash instruction which specifies two states in which (i) the receiver of cash will be Bank A if Bank A provides $\boldsymbol{X}$ which satisfies $\boldsymbol{Y} = \mathrm{H}(\boldsymbol{X})$, or (ii) it will be Bank B if one hour passes. Bank B submits the 1st cash instruction to the cash consensus mechanism. The chaincode subtracts the specified amount of cash of Bank B and stores the 1st cash instruction in the cash state database.

Step 3: After the 1st cash instruction is committed, Bank A queries the cash state database and verifies the content of the 1st cash instruction of Bank B. If Bank A finds that it has errors, he stops the process. In this case, Bank A and Bank B create the 2nd securities instruction and the 2nd cash instruction (refunding of the agreed amount of securities and cash) and submit them to the securities and cash DLT networks after each locking time expires (two hours and one hour) respectively. Otherwise, Bank A creates the 2nd cash instruction (obtaining of the agreed amount of cash) which provides $\boldsymbol{X}$. Bank A submits the 2nd cash instruction to the cash consensus mechanism. The chaincode adds the specified amount of cash to Bank A, disables/removes the 1st cash instruction stored in the cash state database and stores $\boldsymbol{X}$ in the cash state database.

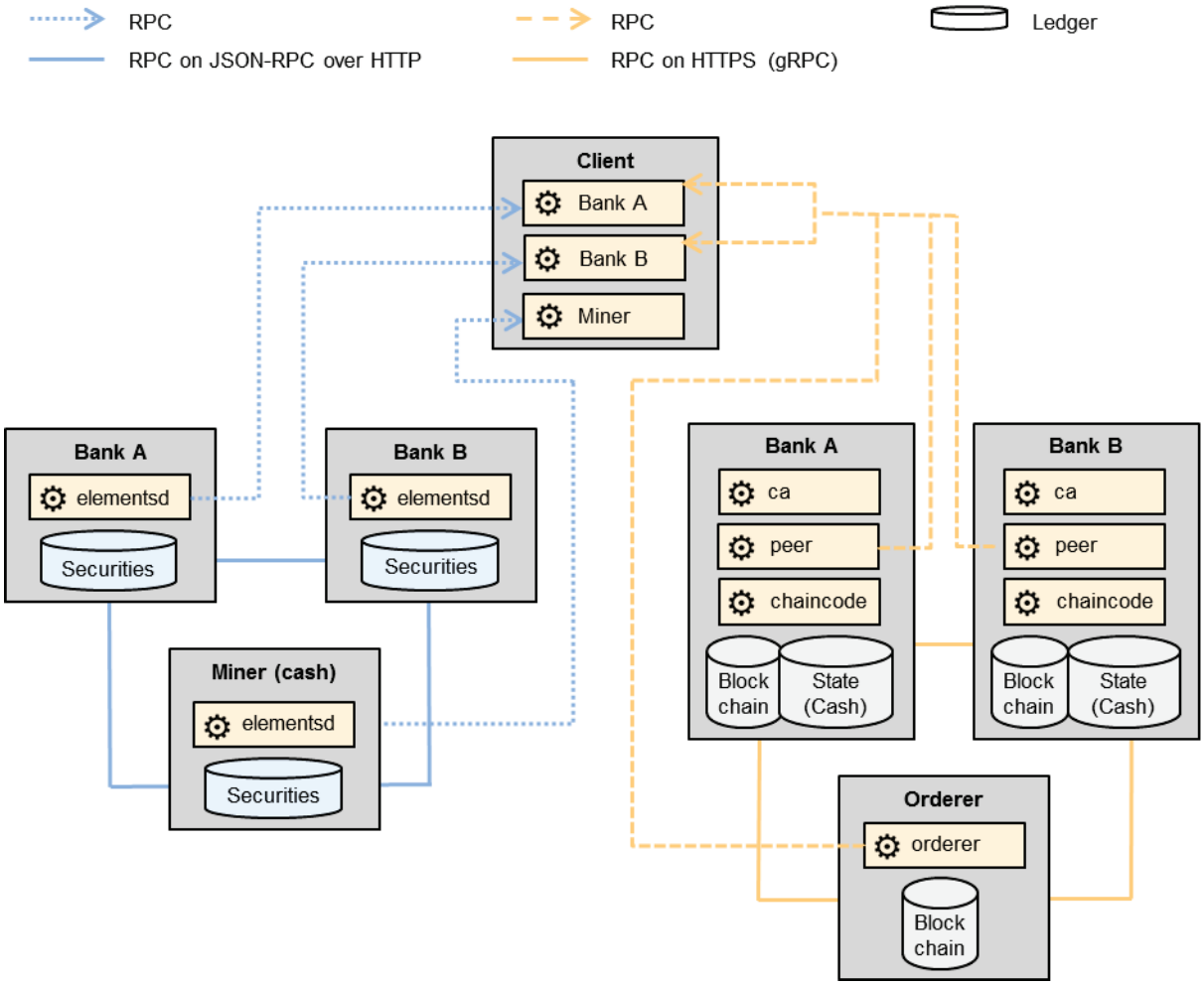
Step 4: After $\boldsymbol{X}$ is committed, Bank B queries the cash state database and obtains $\boldsymbol{X}$.[65] Bank B then creates the 2nd securities instruction (obtaining of the agreed amount of securities) which provides $\boldsymbol{X}$. Bank B submits the 2nd securities instruction to the securities consensus mechanism. The chaincode adds the specified amount of securities to Bank B and disables/removes the 1st securities instruction stored in the securities state database.

[64] Since time of each endorser is not exactly aligned, Bank A may need to retry submitting the 2nd securities instruction until it meets the endorsement policy (i.e. AND condition).

[65] The chaincode can be designed to disable/remove $\boldsymbol{X}$ stored in the state database. Another approach could be Bank B obtains $\boldsymbol{X}$ from the 2nd cash instruction itself that is stored in its blockchain.

Chart O

Network for cross-ledger DvP with HTLC between Elements and Fabric



Connections between clients and Certification Authorities were omitted to simplify this chart.

Cross-ledger DvP with HTLC was also conducted between two DLT networks which use Elements and Fabric respectively. The process followed the previous steps in Annexes 5 and 6. It is confirmed that the secret sharing could be executed in either of the two DLT networks.

Postal address 60640 Frankfurt am Main, Germany
Telephone +49 69 1344 0
Website www.ecb.europa.eu

Postal address 2-1-1 Nihonbashi-Hongokucho, Chuo-ku, Tokyo 103-0021, Japan
Telephone +81 3 3279-1111
Website www.boj.or.jp