（日本銀行仮訳）

EUROPEAN CENTRAL BANK
EUROSYSTEM

BANK OF JAPAN

Project Stella

日本銀行・欧州中央銀行による分散型台帳技術に関する共同調査

― 分散型台帳技術によるＤｖＰ決済の実現 ―

２０１８年３月

目　次

※本稿は日本銀行および欧州中央銀行による報告書「Securities settlement systems: Delivery versus payment in a distributed ledger environment」（本文）の日本銀行決済機構局による仮訳である。

Project Stella

分散型台帳技術によるＤｖＰ決済の実現

１．背景

本報告書は、日本銀行と欧州中央銀行の共同調査プロジェクト「Project Stella」の第２フェーズの結果を取りまとめたものである[1]。今次調査の目的は、資金と証券の受渡のような２つの紐付けられた決済を、分散型台帳技術（distributed ledger technology、DLT）を使って、どのようにデザインし、実現していくかについて検討を行うことである[2]。ここで、法的な論点は共同調査の対象ではない。

資金と証券の DvP（delivery versus payment）決済とは、一方の資産の受渡が行われる場合に限り、他方の資産の受渡が行われるようにする決済の仕組みである。したがって、こうした決済の結果としては、①取引の両当事者間で資金と証券の受渡が成功すること、または②一切の受渡が行われないこと、のいずれかとなる。これは、コンピュータ・サイエンスでは「アトミック性」（atomicity）と呼ばれ、一連の処理が一体不可分のものとして、成功するか失敗するかの二者択一しか生じないことを表す概念として知られる[3]。

DLT は、暗号技術、ピアツーピア（P2P）ネットワーク、コンセンサスアルゴリズム、スマートコントラクト等の要素技術を組み合わせて、単一の中央管理システムを使うことなく、複数の関係者がデータを共有・処理することを可能とする技術である。

本報告書では、DLT の環境の下で、DvP をどのようにデザインし、かつ技術的に実現するかについて検討を行う。具体的には、資金と証券の受渡を紐付ける

---

[1] 今次調査には、日本銀行から小早川周司、河田雄次、小林亜紀子、欧州中央銀行から Dirk Bullmann、Andrej Bachmann、Cedric Humbert、Stephan Mogelin、Giuseppe Galano、Jose Cerecedo、Thomas Leach、Andrea Pinna が参加した。

[2] 今次報告書の取りまとめにあたっては、R3、IBM、DG Lab の各社から技術的な助言を受けた。

[3] 「アトミック」という用語はギリシャ語からきており、不可分性（indivisibility）ないし既約性（irreducibility）を意味するものである。コンピュータ・サイエンスにおけるアトミックな処理（atomic operations）とは、細分化した処理に分割することができず、①すべての処理が一括して実行されるか、あるいは②全く実行されないか、のいずれかとなる一連の処理を指す。

DvP について、既存の方式のほか、現在、議論が進められている DLT を使った新たな方式についても検討を行った。DvP 決済が DLT を用いて実際にどのように実現できるかについて検討を深めるために、Corda、Elements、Hyperledger Fabric（以下、Fabric）という３つの DLT 基盤を使って、プロトタイプを開発した。

今次調査では、取引の両当事者間で資金と証券の受渡を行うという、基本的かつ単純化されたシナリオに基づいて分析を行った。第１フェーズ[4]と同様、第２フェーズの分析も、現行の資金・証券決済システムを再現することを試みたものではないほか、既存の中央銀行サービスについて、DLT を用いたシステムに置き換えることを意図したものではない。

本報告書の構成は、以下のとおりである。第 2 章では、今次調査における主な調査結果を概観する。第３章では、DvP に関連する情報を整理した上で、資産が同一の台帳で管理されるか、複数の台帳で管理されるかという違いに基づいて、DvP を実現する方式についての分類を行う。第４章では、それぞれの方式について、基本的な処理フローを説明した上で、第５章では、複数の視点に基づいて、それぞれの主な特徴を分析する。

２．今次調査における主な調査結果

今次調査を通じて明らかになった点を予めまとめると、以下のとおりである。

（１）DvP は、DLT 基盤によって具体的なデザインは異なるものの、DLT の環境下でも、実現することができる。

DLT を応用した環境においては、①単一のネットワーク上の台帳を使って、資金と証券の DvP 決済を行う方式（単一台帳方式、single-ledger DvP）と、②複数のネットワーク上の台帳を使って、資金と証券の DvP 決済を行う方式（複数台帳方式、cross-ledger DvP）のいずれを使っても、DvP を実現することができる。もっとも、DvP の具体的なデザインについては、DLT 基盤毎の特徴の影響を受ける。例えば、参加者の間で共有される情報の範囲、データ構造、取引相手に引き渡される資産のロックの有無などが挙げられる。さらに、ユースケース次第では、DvP のデザインは、さらに多くの要因の影響を受ける。こ

[4] 「日本銀行・欧州中央銀行による分散型台帳技術に関する共同調査―分散型台帳技術による資金決済システムの流動性節約機能の実現―」（2017 年９月）。

のなかには、DvP を実現するために他のポストトレード・インフラとの間で、どのようなやり取りをするかといった点も含まれる。

（2）DLT は、DvP を実現するにあたって、複数のネットワーク間での接続を一切必要としない新たな方式を可能にする技術である。

今次調査における概念整理と実験結果によると、複数台帳方式においては、異なるネットワーク同士を接続しなくても、DvP を実現できることが確認された。これは、既存の仕組みにはない新たな方式である。「クロスチェーン・アトミック・スワップ」（cross-chain atomic swaps、複数ネットワーク間でのアトミックな資産受渡）といった機能は、異なるネットワーク同士を接続したり、ネットワーク間で取り決めを行うことなく（同一の DLT 基盤および異なる DLT 基盤間の）相互運用性の向上に資する可能性がある[5]。

（3）具体的な DvP のデザインの仕方次第ではあるが、複数台帳方式には、一定の複雑性や、今後克服しなければならない課題が残されている。

接続されていないネットワーク間で DvP 決済を行うにあたっては、取引の両当事者が、互いにやり取りした上で、幾つかの手順を踏むことが必要となる。具体的な DvP のデザインの仕方次第では、取引処理速度に影響を及ぼしたり、一時的に流動性をブロックしたりする必要性が生じる可能性がある。また、運用面からみると、独立して稼動するネットワーク同士が、互いに意図せずして影響を及ぼし合う可能性にも留意する必要がある。さらに、リスクに照らしてみると、異なるネットワーク上で行われる手順が完全に同期されていない以上、定められた手順を終えなかった取引当事者は、元本リスクに晒される可能性がある。こうした追加的なリスクについては、今後、適切に解決する必要がある。

3．DvP

3．1　DvP の定義

DvP は、金融市場インフラにかかる国際基準、すなわち「金融市場インフラのための原則」（以下、FMI 原則）[6]において定義されている。

---

[5] クロスチェーン・アトミック・スワップを可能とする機能は、技術的には非 DLT 基盤にも実装可能である。

[6] BIS 支払・決済システム委員会（CPSS）、証券監督者国際機構専門委員会「金融市場インフラのための原則」（2012 年 4 月）。

DvP は、２つの結び付いた債務の決済、具体的には資金と証券の受渡を伴うものである。FMI 原則の原則１２では、DvP の仕組みについて「一方の債務のファイナルな決済が、それと結び付けられた債務のファイナルな決済が行われる場合にのみ実行されることを確保すること」が求められており、「その場合、金融市場インフラの決済がグロスベース（取引毎）かネットベースか、決済がファイナルとなるのがいつかは問わない」とされている。DvP を行うことにより、取引の両当事者は、元本リスク、すなわち、証券の売り手が証券を引き渡したにもかかわらず資金を受け取れない、または証券の買い手が資金を支払ったにもかかわらず証券を受け取れない、というリスクを回避することができる[7]。

DvP を実現するためには、資金と証券の受渡を紐付け、一方の債務のファイナルな決済が、他方の債務のファイナルな決済を条件として実行されるようにする必要がある。ここでファイナルな決済とは、「原契約の条件に従った、金融市場インフラやその参加者による、無条件かつ取消不能な資産・金融商品の移転または債務の履行」と定義される。

DvP の実現にあたっては、資金の受渡と証券の受渡のタイミングが異なる場合も含まれ、厳密には、資金と証券が同時に決済される必要はない。このため、DvP は、一方の債務の決済を、他方の債務の決済の後に行う方法でも実現可能である。また、DvP は、グロスベースまたはネットベースのいずれの方法でも実現することができる[8]。

３．２　DLT 環境下での DvP 実現方法

DLT 環境下で DvP を実現する方式としては、概念上、大きく分けて、①単一のネットワーク上の台帳を使って、資金と証券の DvP 決済を行う方式（単一台帳方式、single-ledger DvP）と、②複数のネットワーク上の台帳を使って、資金と証券の DvP 決済を行う方式（複数台帳方式、cross-ledger DvP）の２つが考えられる。

<u>単一台帳方式</u>
単一台帳方式では、資金と証券は単一の台帳で管理されており、典型的には、取引当事者双方が互いの振替指図の内容を確認したうえで、資金と証券の振替

[7] 取引当事者が晒されるリスクについての議論は第５章を参照。
[8] DvP の類型に関する議論は、CPSS「証券決済における DvP」（1992 年９月）を参照。

が 1 つのトランザクションとして処理される。これは、既存の証券決済の仕組みにおける、いわゆる「統合型（integrated model）」の DvP――すなわち、資金と証券が単一のシステム上で処理される方式（TARGET 2 Securities や日銀ネット国債系において採用。詳細は原文の別添１参照）――と類似していると言える。

複数台帳方式

複数台帳方式では、資金と証券はそれぞれ別々の台帳で管理され、両者の振替を紐付けるための仕組みが備わっている[9]。こうしたアプローチは、一方のネットワークが他方のネットワークにおける記録の更新に依拠するという性質上、処理が複雑となり、一方のネットワークにおける取極めは、他方のネットワークにも影響を及ぼす。複数台帳方式は、さらに、以下の２つに分類できる。

①ネットワーク間の接続[10]がある複数台帳方式（cross-ledger DvP with connection between ledgers）

この方式は、既存の証券決済の仕組みにおける、いわゆる「インターフェース型（interfaced model）」の DvP――すなわち、資金と証券の振替が別々のシステムで行われる場合に、双方のシステムが連携して DvP を実現する方式（一般債振替制度・短期社債振替制度における日銀ネットと証券保管振替機構の間の DvP で採用。詳細は原文の別添１参照）――と類似していると言える。既存の仕組みでは、一般的には、まず、証券決済システムが受渡対象証券の残高をブロックする（具体的には、残高を取り分ける、またはエスクロー口座に振り替える等）。資金決済システムは、証券決済システムからブロック完了の通知を受けると、資金の振替を実行し、証券決済システムに資金振替完了の通知を伝える。証券決済システムは、資金決済システムの通知を受けて、証券残高をリリースし、証券の振替を実行する。

DLT 環境下でこうしたアプローチを実現するには、２つのネットワーク間の連携を管理するための何らかの仲介機能（intermediary）が必要となる可能性がある。この仲介機能は、信頼のある（trusted）主体が行う場合と信頼のない（untrusted）主体が行う場合とにさらに大別できるが[11]、現時点では、DvP

[9] 複数台帳方式は、一般に、DLT・非 DLT ネットワークを含む、複数のネットワーク間の「相互運用性」を巡る文脈で議論されている。

[10] ネットワーク間の接続としては、直接的な接続、または間接的な接続が考えられる。間接的な接続とは、２つのネットワークが第３のネットワーク経由で接続されることを指す。

[11] 信頼のない主体が行う仲介機能、すなわち、それらの主体が中継ポイントとして決済中継機能のみを提供するアプローチなどは、その仲介者のインセンティブ・メカニズムの設

のような双方向の決済を実現する方法としては、信頼のある主体が行う仲介機能を用いる以外は、十分に議論が進んでいない。今次調査では、このアプローチには焦点を当てず、今後の技術発展等に委ねることとした。

②ネットワーク間の接続がない複数台帳方式（cross-ledger DvP without connection between ledgers）
既存の仕組みにおいて、ネットワーク間の接続がない複数台帳方式に類似した仕組みは存在しない。DLT の登場に伴い、ネットワーク間の接続や、ネットワーク間の連携を行う第三者を用いずに DvP を実現する仕組みとして、「クロスチェーン・アトミック・スワップ」（複数ネットワーク間でのアトミックな資産受渡、cross-chain atomic swaps）と呼ばれる手法などが開発された。

クロスチェーン・アトミック・スワップは、もともと、2 つのブロックチェーン上の異なる仮想通貨を、第三者に頼ることなく受渡する目的で開発されたものである[12]。2 つの異なるネットワーク間で 2 つの異なる資産をアトミックに受渡するために重要となるのは、電子署名と、「ハッシュ・タイムロック・コントラクト」（Hashed Timelock Contracts、以下 HTLC）と呼ばれる技術である[13]。

HTLC は、①暗号学的ハッシュ関数と、②タイムアウト（timelock）機能とで構成される。例えば、取引当事者 A と B が、HTLC を用いて複数台帳方式で資金と証券の受渡を行う場合、A と B はまず、A が生成した乱数（シークレット、

計を含め、DLT 開発コミュニティにおいて、技術・理論の両面から活発な議論が行われている。

[12] クロスチェーン・アトミック・スワップのアイディアは、2013 年に Tier Nolan によって初めて整理された（(https://bitcointalk.org/index.php? topic=193281)。しかし、当時は、技術的な制約から、クロスチェーン・アトミック・スワップにはセキュリティ上の問題が存在した。具体的にはOP_CSV(BIP-112)やOP_CLTV(BIP-65)、Segregated Witness（いわゆる「SegWit」）が当時は導入されておらず、事前に署名する返金用トランザクションはトランザクション展性問題により改変される危険性が存在した。近年になり、前述の機能が導入されたことにより、こうした問題点は解消された。今次調査では、Tier Nolan のアプローチを改良し、事前に返金用トランザクションに署名しておく必要のないアプローチを用いた。詳細は原文の別添 5 を参照。

[13] HTLC は、Lightning Networks で用いられる要素技術であり、同様の考えは Ripple Interledger Protocol（ILP）でも用いられている。もっとも、Lightning Networks も Ripple ILP も、ネットワーク間に何らかの接続があることを想定しており、その意味ではこれらは「ネットワーク間の接続がある複数台帳方式」に分類される。HTLC についての詳細は、Joseph Poon and Thaddeus Dryja「The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments」（2016 年 1 月）を参照。

secret）のハッシュ値を用いて、受渡対象となる資産をブロックする [14]。双方の資産がブロックされた後、A はシークレットを用いて受取予定の資産（例えば、資金）を受け取り、それと同時に B にシークレットが開示される。その後、B は当該シークレットを用いて受取予定の資産（例えば、証券）を受け取る [15]。タイムアウト機能は、事前に定めた時間内に必要な処理が完了しなかった場合に、元の所有者がブロックされた証券および資金を取り戻すために必要となる（詳細は第 4 章 2 節を参照 [16]）。

ネットワーク間の接続がない複数台帳方式により DvP を実現する方法には、上記以外の方法もあり得るが、本報告書では、HTLC を利用する方式（以下、複数台帳 HTLC 方式）に焦点を当てている。

なお、現在、DLT 開発コミュニティでは、ネットワーク間の連携にかかる様々な仕組みについて議論や実験が進められており、それらの中には、上記の分類に必ずしも該当しないものもある [17]。

---

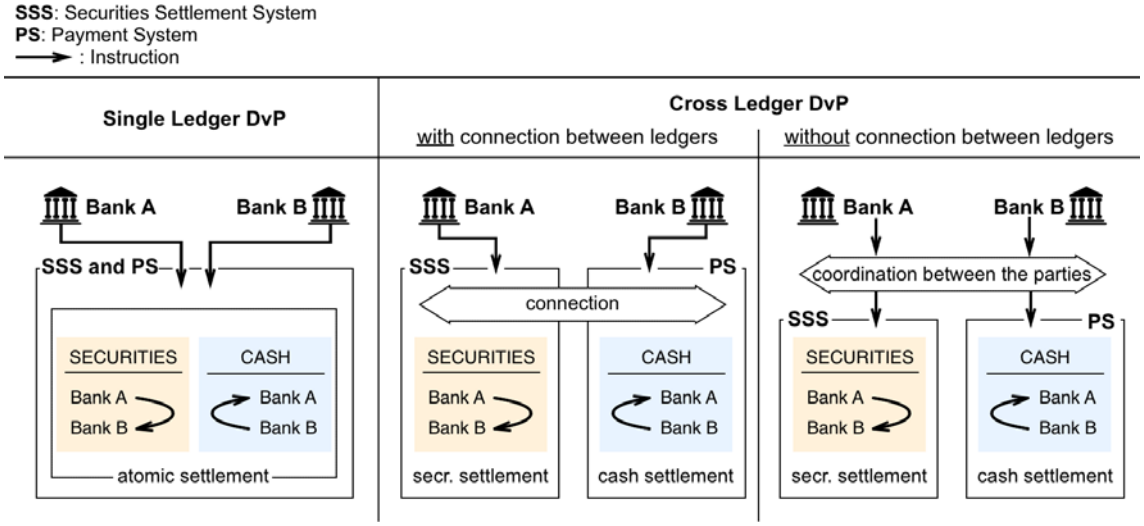
[14] 一方向性ハッシュ関数を用いた場合、合理的な想定に基づけば、A が生成したシークレットを、そのハッシュ値から B が探し出すことは不可能である。

[15] シークレットの共有については、第 5 章 4 節を参照。

[16] HTLC の特徴の一つとして、取引当事者間に非対称性が存在する点にある。例えば、シークレットは取引当事者 A と B のいずれか一方のみが生成する。DvP を実現する仕組みを一切用いない場合、A が B へ資産を渡しても、B が A へ対となる資産を渡す保証はない。そのため HTLC のような仕組みを用いるが、ここで A、B 双方が別々のシークレットを生成し、双方の資産を双方のシークレットのハッシュ値でブロックする場合を考える（注：A に渡される資産は B の生成したシークレットが掲示されない限りブロックされる、また B に渡される資産は A の生成したシークレットが掲示されない限りブロックされる）。この場合、たとえ A が B へシークレットを開示しても、B が A へ正しいシークレットを開示する保証はない。すなわち、両者が別々のシークレットを生成する場合は、資産の受渡を行うか、シークレットの受渡を行うかの違いのみで、HTLC を用いない場合と本質的な差異はないとも言える。そのため、一方が生成したシークレットのハッシュ値を双方が用いることとしている。取引当事者 A、B 双方が指定するタイムアウト時間にも同様の非対称性が存在する。詳細は第 4 章 2 節「タイムアウト時間の非対称性」を参照。

[17] 例えば、以下の論文では、台帳間の接続ではなく、参加者間の信頼度（trust model）に応じた分類が行われている。Vitalik Buterin「Chain Interoperability」（2016 年 9 月）https://static1.squarespace.com/static/55f73743e4b051cfcc0b02cf/t/5886800ecd0f68de303349b1/1485209617040/Chain+Interoperability.pdf

【図表１】DLT 環境下での DvP 実現方法



（注）図中において、SSS は証券決済用の DLT ネットワーク、PS は資金決済用の DLT ネットワークを示す。

４．ＤＬＴ環境下でのＤｖＰの処理フロー

「Project Stella」第２フェーズでは、日本銀行と欧州中央銀行は、DLT の環境下で DvP を実現する方式について、定性的な評価のほか、実際に実験を進めながら定量的な評価も行う形で検討を進めた。分析は、取引の両当事者（銀行Ａと銀行Ｂ）が、中央管理システムを介さずに、それぞれ合意した額の資金と証券の受渡を行うという、基本的かつ単純化されたシナリオに基づいている。

プロトタイプは、単一台帳方式（single-ledger DvP）と複数台帳 HTLC 方式（cross-ledger DvP with HTLC）を対象に、Corda、Elements、Fabric の３ つのＤＬＴ基盤を用いて実装された。本章では、各々の方式について、その基本的な処理フローに加え、フェイル（決済未了）が発生し得るシナリオについても記載する [18]。各方式における詳細なフェイルシナリオについては原文の別添２を、Corda、Elements、Fabric における実際の処理フローについては原文の別添４、５、６を参照されたい。本章では、各 DLT 基盤固有の説明から離れ、台帳の一貫性を維持する仕組み全般を「コンセンサス形成メカニズム」（consensus mechanism）という用語で総称する [19]。

[18] 処理の流れが分かりやすいように、ここでは Corda および Elements で用いられる「UTXO」（未使用トランザクションアウトプット、Unspent Transaction Output）モデルに基づいて処理内容を記載する。また、Elements の場合における手数料やおつりの概念などは省略する。

[19] コンセンサス形成メカニズムという用語は、トランザクションにかかるメッセージング

４．１　単一台帳方式における処理フロー

単一台帳方式において重要な点は、取引当事者双方が、互いに相手の振替指図の内容を確認した後ではじめて、資金と証券の受渡が 1 つのトランザクションとして処理されるという点である。双方の振替指図の内容確認から双方の振替指図を含むトランザクション生成までが、確認を行った証となる電子署名を用いて、取引当事者間でのみピアツーピアで行われる。そのため、DLT ネットワーク上に何らかのマッチング（照合）機能を設ける必要はないと言える。

図表２において、証券の売り手（銀行Ａ）と証券の買い手（銀行Ｂ）は、取引を行う証券の種類と受渡額について既に合意しているものとする。例えば、①銀行Ａから銀行Ｂへ８単位の証券を渡す代わりに、②銀行Ｂから銀行Ａへ６単位の資金を渡すこととする。ここで、銀行Ａと銀行Ｂの双方とも、資金と証券の受渡を行う同一 DLT ネットワークに既に参加しているものとする。

決済完了シナリオ
取引の両当事者が以下の処理ステップを踏む場合、決済は完了する。

1. 銀行Ａ（証券の売り手、資金の受け手）は事前に合意した額の証券を含む証券振替指図を生成する。銀行Ｂ（証券の買い手、資金の出し手）は事前に合意した額の資金を含む資金振替指図を生成する。この段階では、双方ともに自らの振替指図に署名は行わない。

2. 銀行Ａは、自身の署名を付さずに、証券振替指図を銀行Ｂに伝える[20]。銀行Ｂは、銀行Ａの証券振替指図の内容を確認（例えば、銀行Ａの生成した証券振替指図の内容と事前の合意に齟齬がないか等を確認）する。問題がなければ、銀行Ｂは、銀行Ａの証券振替指図と自身の資金振替指図を 1 つにまとめる（以降、当該指図を資金・証券振替指図と記載する）。銀行Ｂは、資金・証券振替指図のうち、自身が生成した資金振替指図について署名を付した上、当該資金・証券振替指図を銀行Ａへ伝える。

3. 銀行Ａは資金・証券振替指図の内容を確認（例えば、銀行Ｂの生成した資金振替指図の内容と事前の合意に齟齬がないか、自身の生成した証券振替指図

処理、検証および確認、時間順序決定、スマートコントラクト実行、台帳への書込み等を含む、DLT 基盤固有のトランザクション処理機構全般を総称するものとして用いている。

[20] 最初に自身の振替指図を伝えるのは銀行Ａと銀行Ｂのどちらでも構わないが、例としてここでは銀行Ａとする。

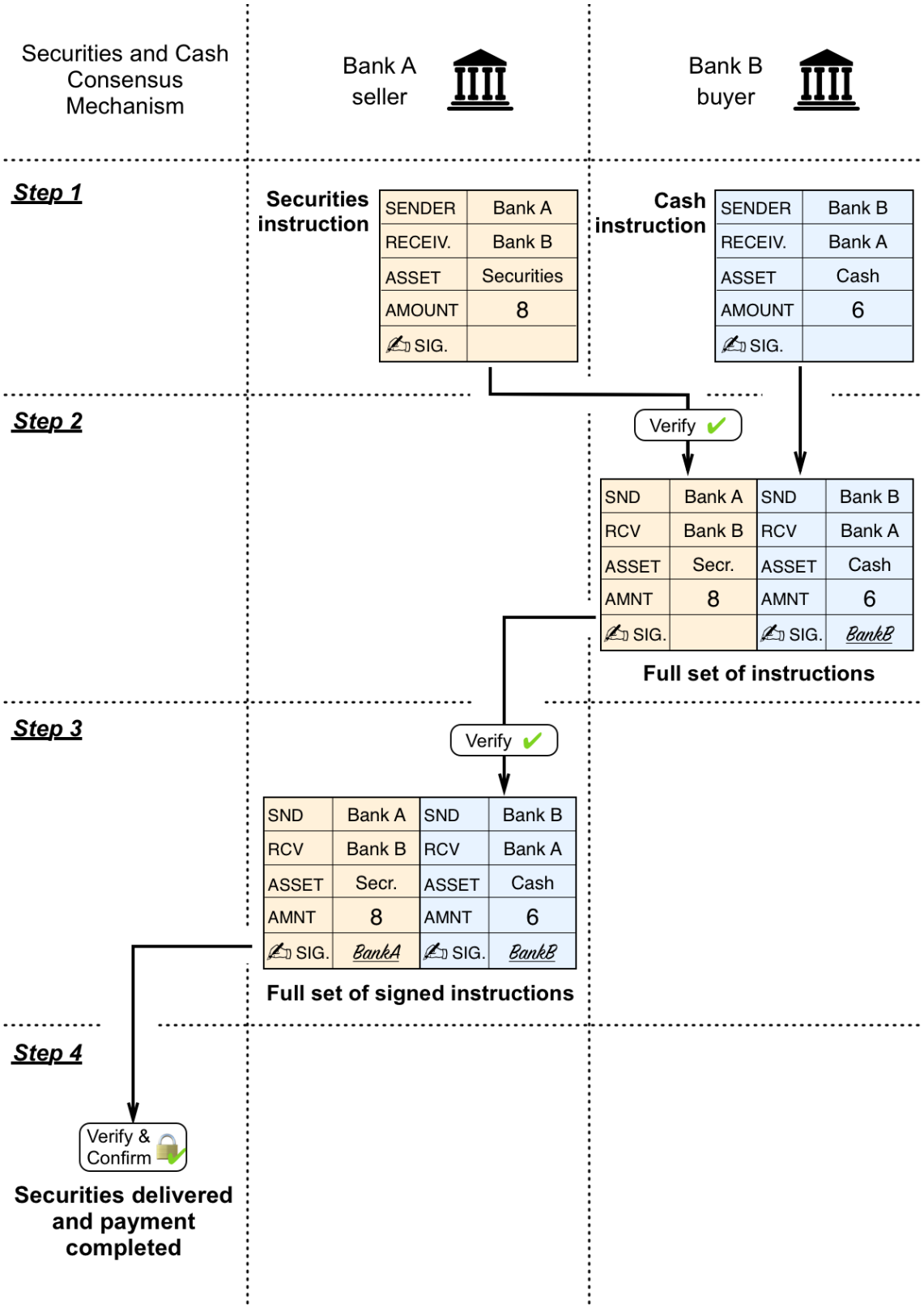
の内容について改変されていないか等を確認）する。問題がなければ、銀行Ａは、資金・証券振替指図のうち、自身が生成した証券振替指図について署名を付した上、当該資金・証券振替指図を DLT ネットワーク上のコンセンサス形成メカニズムへ送信する。

4. DLT 基盤のコンセンサス形成メカニズムにより、当該資金・証券振替指図は検証・確認され[21]、問題がなければ、その結果が DLT ネットワーク上の台帳に書き込まれる[22]。
この段階において、事前に合意した額の資金と証券は、それぞれ銀行Ａと銀行Ｂへ渡されることとなる。

[21] 取引当事者双方が行う検証とは、事前に合意した証券の種類や受渡額に対して齟齬がないかを確認するものである。一方、DLT 基盤が行う検証とは、その DLT 基盤独自のコンセンサス形成メカニズムから逸脱していないかを確認するものである（注：例えば、トランザクションのフォーマットや実行結果の同一性、インプットの一意性すなわち当該インプットが未使用か否か、など）。

[22] DLT 基盤毎に、トランザクションが処理・コミットされる実際の流れは異なる。
例えば Corda の場合、銀行 A が資金・証券振替指図を validating notary へ送信後、当該 notary がインプットステートの一意性確認や当該トランザクションの内容検証等を行い、問題がなければ、自身の署名を付して銀行 A へ結果を返す。銀行Ａは、署名付きの結果を全取引当事者（ここでは銀行Ｂのみ）へブロードキャストし、銀行Ａと銀行Ｂは署名付きの結果をそれぞれ検証の上、問題がなければ、自身の台帳へ書き込む。
Elements の場合、銀行 A が資金・証券振替指図をブロードキャストで送信後、それを受信した近傍ノードがその内容を検証し、問題がなければ、一定数の近傍ノードへさらにブロードキャストを行う。最終的にマイナーが受信後、マイナーはその内容を検証し、問題がなければ、当該トランザクションを含むブロックを生成し、ブロードキャストを行う。銀行Ａと銀行Ｂはブロック受信後に、当該ブロックの内容を検証し、問題がなければ、自身の台帳に書き込む。
Fabric の場合、銀行Ａが資金・証券振替指図を endorser へ送信後、endorser は入出力セット（使用データおよび実行結果のセット）を、自身の署名を付して銀行Ａへ返す。銀行Ａは endorser からの結果を確認の上、入出力セットをトランザクションに纏め、orderer へ送信する。当該 orderer は当該トランザクションを含むブロックを生成し、同一チャネル内の全ノードへブロードキャストする。各ノードはブロックの内容を検証の上、問題なければ、自身の台帳に書き込む。
注：上記の記述での「検証」の内容は、DLT 基盤およびノードの役割毎に異なる。

【図表 2】単一台帳方式における処理フロー

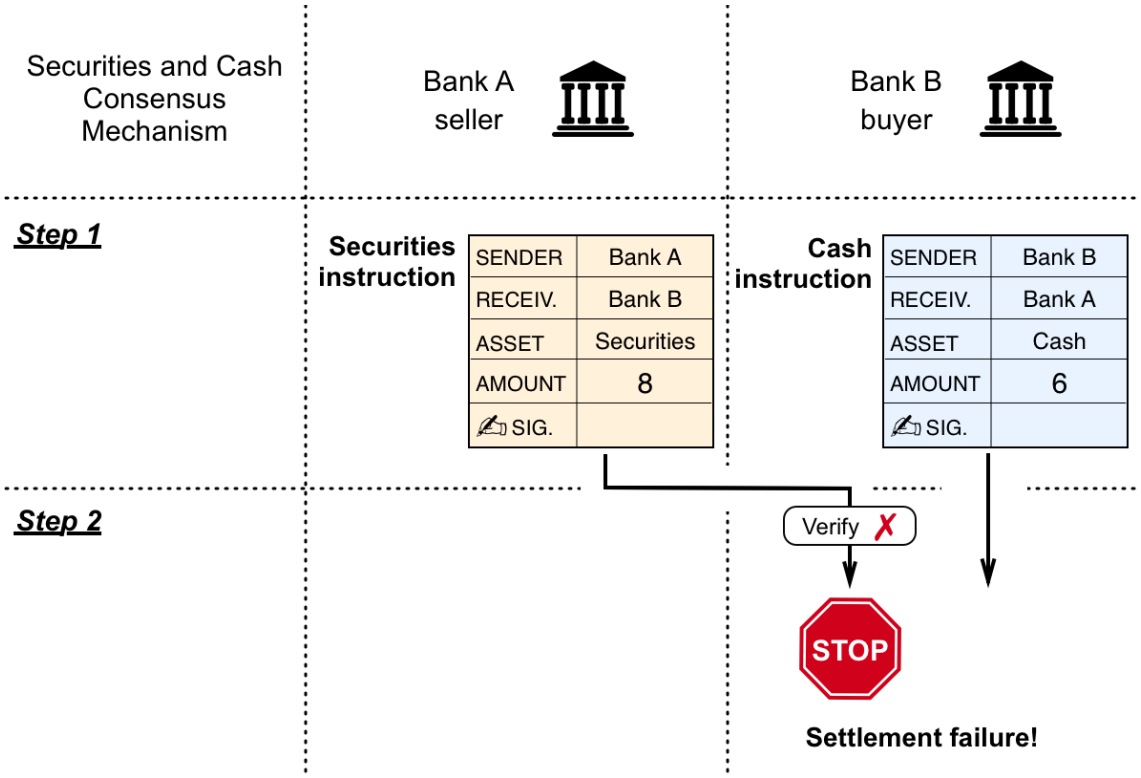


（注）銀行Ａは証券の売り手（資金の受け手）、銀行Ｂは証券の買い手（資金の出し手）を指す。
また、証券振替指図は橙、資金振替指図は青で表記。以下、同様。

フェイルシナリオ

単一台帳方式では、いずれかの処理ステップが完了しなければ、フェイルとなり得る（例えば、前述のステップ２において、銀行Ｂが、銀行Ａの生成した証券振替指図の内容と事前の合意に齟齬を確認した場合）。処理が中断され得るいずれのシナリオにおいても、最終的な資金・証券振替指図が台帳に書き込まれることはない。したがって、資金と証券は元の所有者に保有されたままであり、即時に他の振替指図作成に用いることが可能である [23]。図表３はフェイルシナリオの一例を示している。

【図表３】単一台帳方式におけるフェイルの例（ステップ２で中断された場合）



[23] 複数の異なるトランザクションが同一のインプットを使用するような競合が発生しないよう、クライアント側で何らかのロック機能を設けることが考えられる。詳細は原文の別添４、５、６を参照。

４．２　複数台帳 HTLC 方式における処理フロー

複数台帳 HTLC 方式において重要な点は、①取引当事者双方が、相手の振替指図の内容を確認するにあたり、台帳に書き込まれた結果を用いるという点（注：言い換えれば、単一台帳方式と異なり、最終的な証券資金受渡に至る中間段階の振替指図が台帳に書き込まれるという点）、②資金と証券の条件付き受渡として HTLC を用いる点の２つである。後者（②）は、具体的には、暗号学的ハッシュ関数を用いることにより、資金と証券の受渡を一時的にブロックし、タイムアウト機能を用いることにより、処理が完了しなかった場合に元の所有者がブロックされた証券および資金を取り戻すことを指す。単一台帳方式と同じく、複数台帳 HTLC 方式においても、確認を行った証となる電子署名を用いて、取引当事者間で互いの取引の確認が行われる。さらに、振替指図に含まれる共通のハッシュ値をもとに一連の振替指図を特定することが可能である。以上より、単一台帳方式の場合と同じく、DLT ネットワーク上に何らかのマッチング（照合）機能を設ける必要はないと言える。

図表 4[24]において、証券の売り手（銀行Ａ）と証券の買い手（銀行Ｂ）は、取引を行う証券の種類、受渡額、タイムアウト時間、暗号学的ハッシュ関数について既に合意しているものとする。例えば、①銀行Ａから銀行Ｂへ 2 時間以内に 8 単位の証券を渡す代わりに、②銀行Ｂから銀行Ａへ 1 時間以内に 6 単位の資金を渡すこととする[25]。ここで、銀行Ａと銀行Ｂの双方とも、証券の DLT ネットワークと資金の DLT ネットワークの両方に既に参加しており、各々の DLT ネットワーク上での時間の経過速度も把握しているものとする（時間の経過速度に関する論点については第５章５節を参照）。

決済完了シナリオ

取引の両当事者が以下の処理ステップを踏む場合、決済は完了する（図表 4 を参照）。

1. 銀行Ａ（証券の売り手、資金の受け手）は乱数（$\boldsymbol{X}$）を生成し、そのハッシュ値（$\boldsymbol{Y} = \mathrm{H}(\boldsymbol{X})$）を得る[26]。銀行Ａはハッシュ値（$\boldsymbol{Y}$）を銀行Ｂ（証券の買

[24] 脚注 18 と 19 を参照。

[25] 今次調査で用いた多くの DLT 基盤において、タイムアウト時間は絶対時刻（例えば、2018 年３月 31 日の午前 12 時）もしくは相対時刻（例えば、コンセンサス形成メカニズムによりトランザクションが生成されてから 1 時間以内）の双方を指定可能であった。本文中に記載しているタイムアウト時間（2 時間と 1 時間）はあくまで一例として用いた値に過ぎず、状況に応じて異なる値を用いることができる。

[26] 最初に乱数を生成するのは銀行Ａと銀行Ｂのどちらでも構わないが、ここでは銀行Ａが

い手、資金の出し手）に伝える。一方向性ハッシュ関数を用いる限り、合理的な想定に基づけば、銀行Ｂがハッシュ値（$\boldsymbol{Y}$）から元の値（$\boldsymbol{X}$）を得ることは不可能である。銀行Ａは事前に合意した額の証券を含む第 1 の証券振替指図を生成する。この証券振替指図（証券の出し手は銀行Ａ）において、受け手は、①銀行Ｂ（銀行Ｂが$\boldsymbol{Y} = \mathrm{H}(\boldsymbol{X})$の条件を満たす値$\boldsymbol{X}$を掲示した場合）もしくは、②銀行Ａ（2 時間が経過した場合）のいずれかである。銀行Ａは第 1 の証券振替指図に署名を付した上で、証券 DLT ネットワーク上のコンセンサス形成メカニズムへ送信する。

2. DLT 基盤のコンセンサス形成メカニズムにより、第 1 の証券振替指図は検証・確認され、問題がなければ、その結果が証券 DLT ネットワーク上の台帳に書き込まれる。

3. 銀行Ｂは証券 DLT ネットワーク上の自身の台帳を見て、第 1 の証券振替指図の内容を確認（例えば、銀行Ａの生成した第 1 の証券振替指図の内容と事前の合意に齟齬がないか等を確認）する。問題がなければ、銀行Ｂは事前に合意した額の資金を含む第 1 の資金振替指図を生成する。この資金振替指図（資金の出し手は銀行Ｂ）において、受け手は、①銀行Ａ（銀行Ａが$\boldsymbol{Y} = \mathrm{H}(\boldsymbol{X})$の条件を満たす値$\boldsymbol{X}$を掲示した場合）もしくは、②銀行Ｂ（1 時間が経過した場合）のいずれかである。銀行Ｂは第 1 の証券振替指図に署名を付した上で、資金 DLT ネットワーク上のコンセンサス形成メカニズムへ送信する。

4. DLT 基盤のコンセンサス形成メカニズムにより、第 1 の資金振替指図は検証・確認され、問題がなければ、その結果が資金 DLT ネットワーク上の台帳に書き込まれる。

5. 銀行Ａは資金 DLT ネットワーク上の自身の台帳を見て、第 1 の資金振替指図の内容を確認（例えば、銀行Ｂの生成した第 1 の資金振替指図の内容と事前の合意に齟齬がないか等を確認）する。問題がなければ、銀行Ａは、事前に合意した額の資金を受け取るために第 2 の資金振替指図を生成し、$\boldsymbol{X}$をその中に含める。銀行Ａは第 2 の資金振替指図に署名を付した上で、資金 DLT ネットワーク上のコンセンサス形成メカニズムへ送信する。

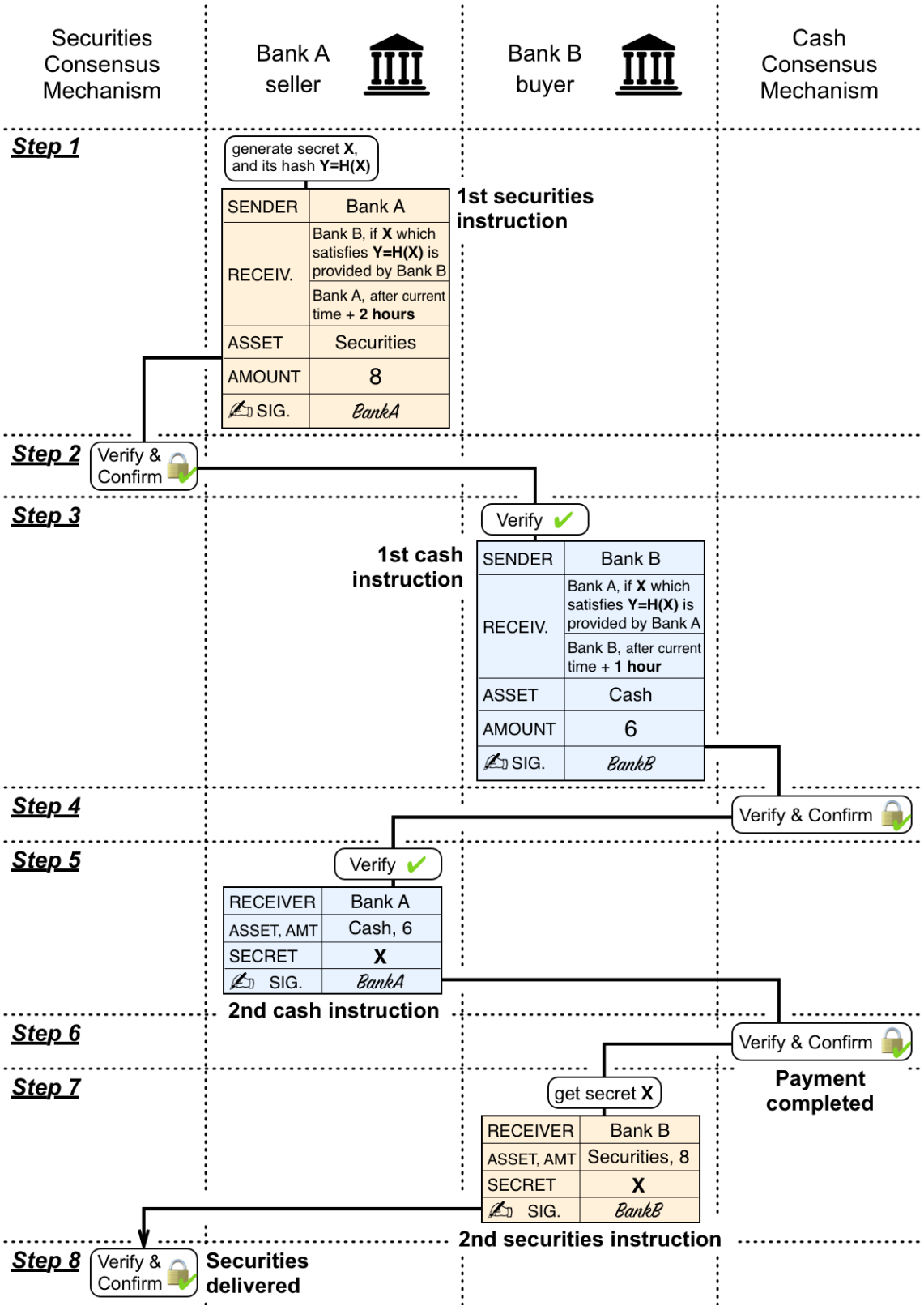
乱数を生成するものとする。

6. DLT 基盤のコンセンサス形成メカニズムにより、第２の資金振替指図は検証・確認され、問題がなければ、その結果が資金 DLT ネットワーク上の台帳に書き込まれる。
   この段階において、事前に合意した額の資金は、銀行Ｂから銀行Ａへ渡されることとなる。

7. 銀行Ｂは資金 DLT ネットワーク上の自身の台帳を見て、銀行 A の第２の資金振替指図から$\boldsymbol{X}$を得る。銀行Ｂは、事前に合意した額の証券を受け取るために第２の証券振替指図を生成し、$\boldsymbol{X}$をその中に含める。銀行Ｂは第２の証券振替指図に署名を付した上で、証券 DLT ネットワーク上のコンセンサス形成メカニズムへ送信する。

8. DLT 基盤のコンセンサス形成メカニズムにより、第２の証券振替指図は検証・確認され、問題がなければ、その結果が証券 DLT ネットワーク上の台帳に書き込まれる。
   この段階において、事前に合意した額の証券は、銀行Ａから銀行Ｂへ渡されることとなる。

【図表４】複数台帳 HTLC 方式における処理フロー



フェイルシナリオ

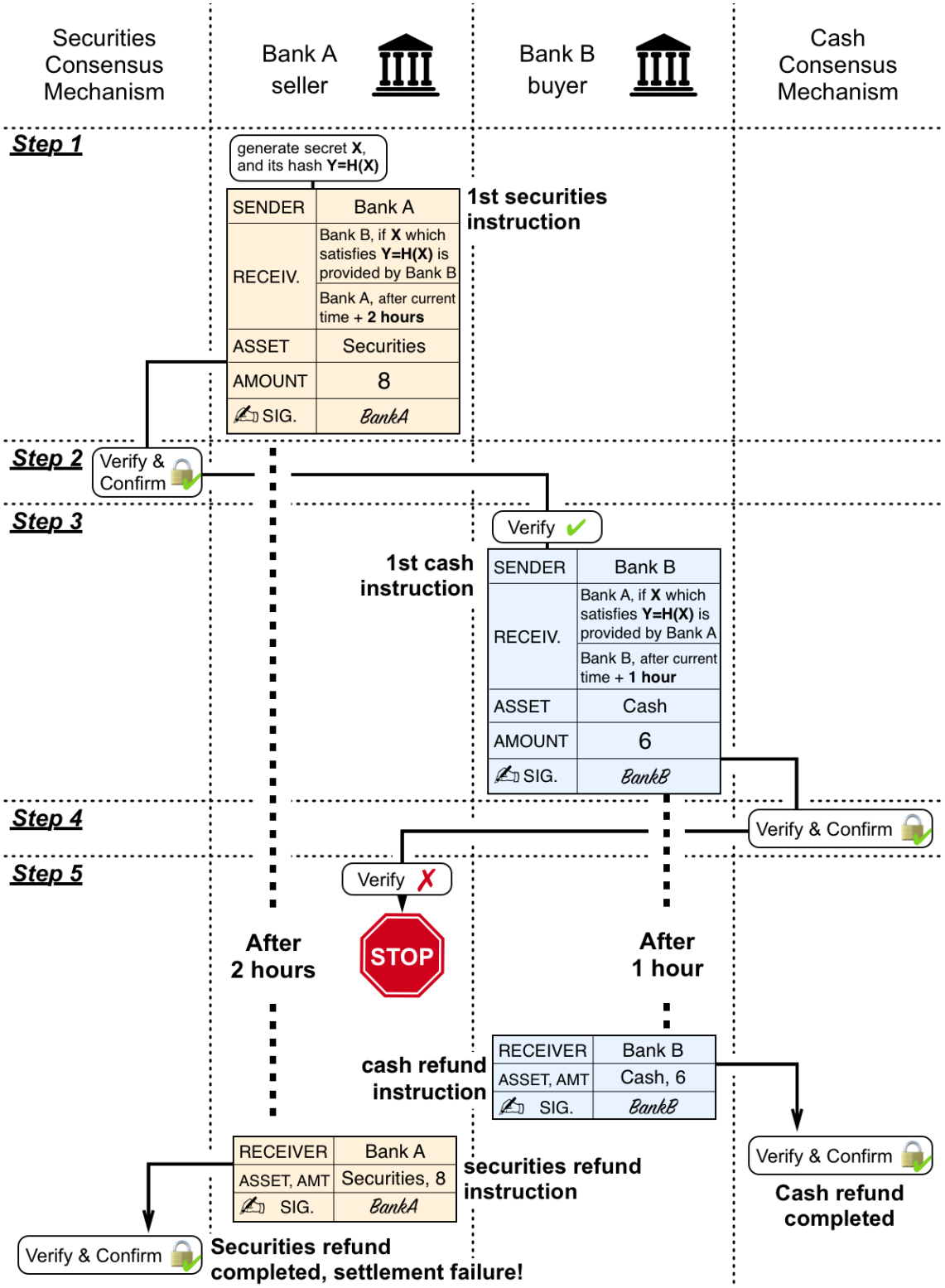
複数台帳 HTLC 方式では、いずれかの処理ステップが完了しなければ、フェイルとなり得る。すなわち、中断する処理ステップにより、2 種類のリスクシナリオに分かれることとなる。リスクシナリオ①では、資金と証券の受渡は失敗するが、それらは各々の出し手に戻る。一方で、リスクシナリオ②では、資金と証券の受渡は失敗し、さらに資金と証券の両方を一方の取引当事者のみが得ることになり、他方の取引当事者は元本リスクに晒されることとなる。

リスクシナリオ①とは、例えばステップ 5 において処理が中断された場合に起こり得る（図表 5 を参照）。ステップ 5 において、第 1 の証券振替指図と第 1 の資金振替指図は既に台帳に記録されているものの、銀行Ａ（資金の受け手であり、$\boldsymbol{X}$を生成した側）が、事前に取り決めたタイムアウト時間内（1 時間以内）に第 2 の資金振替指図を送信しない場合を考える。この場合、資金と証券の受渡は失敗するものの、タイムアウト時間経過後には元の所有者がブロックされた証券および資金を取り戻すことができるため、取引当事者いずれも元本リスクには晒されないと言える。もっとも、資金と証券の受渡失敗に伴い、取引の両当事者は再構築コスト・リスクと流動性リスクに晒されることとなる。

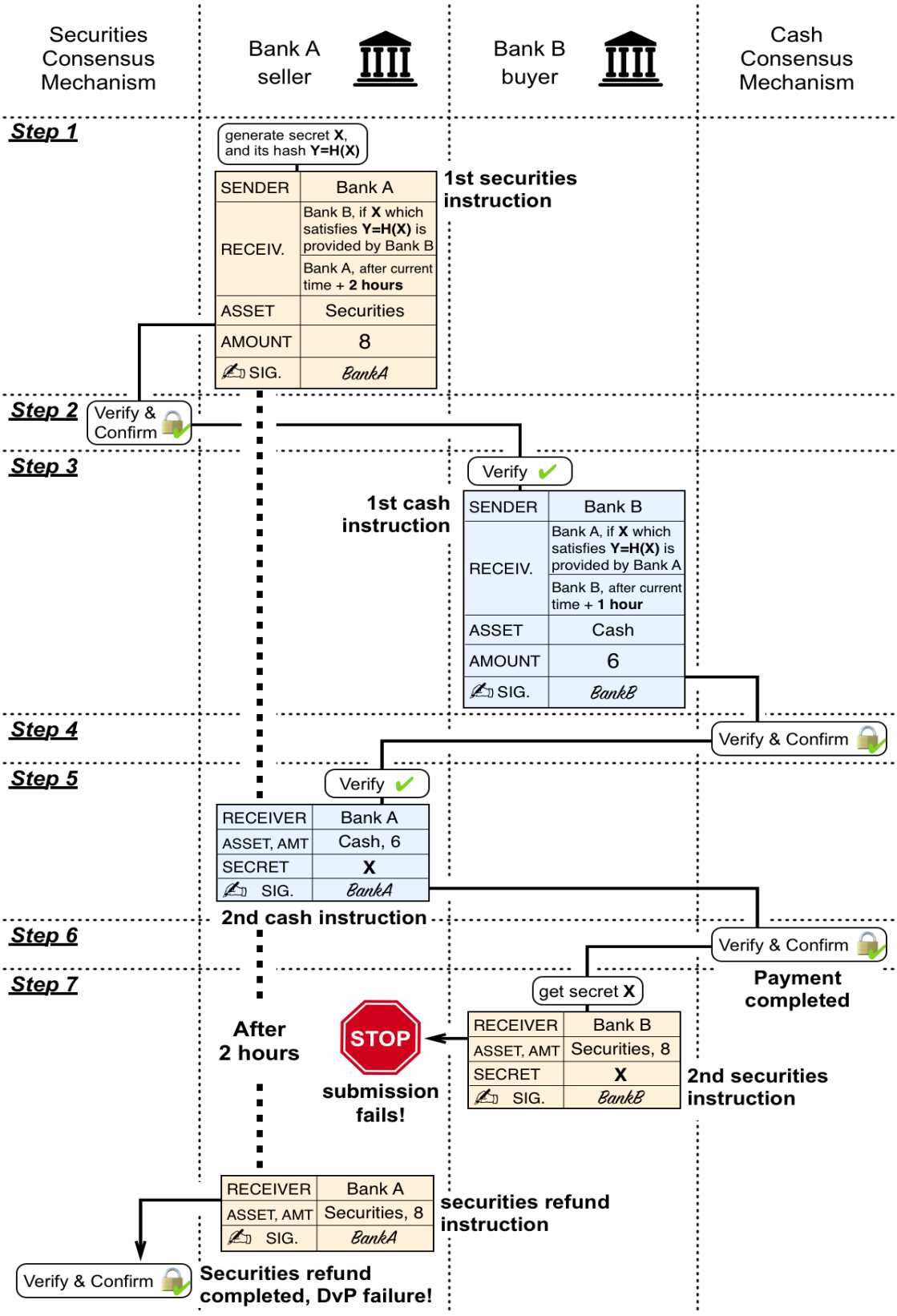
リスクシナリオ②とは、例えばステップ 7 において処理が中断された場合に起こり得る（図表 6 を参照）。ステップ 7 において、銀行Ａは既に事前に合意した額の資金を得ているものの、銀行 Ｂ（証券の受け手）が事前に取り決めたタイムアウト時間内（2 時間以内）に第 2 の証券振替指図を送信しない場合を考える。この場合、タイムアウト時間経過後（2 時間後）には、銀行 Ａ は第 1 の証券振替指図でロックされた証券を取り戻すことが可能となる。場合によっては、一方の取引当事者（銀行 Ａ）は資金と証券の両方を取得する一方、他方の取引当事者（銀行 Ｂ）は、元本リスクに晒されることが起こり得る。このようなリスクシナリオが顕現化した場合、資金と証券の片方のみしか受け渡されないことになり、決済は未了となる[27]。このシナリオは、現状の HTLC の問題点を浮き彫りにするものであり、さらなる改良が求められると言える。

[27] このようなリスクを避けるために、運用面で対処することも考えられる。例えば、タイムアウト時間を 24 時間や 48 時間などと長くすることが考えられる。資金と証券の 2 つのタイムアウト時間の差を長くするほど、資金と証券の受渡成功確率が高まることになるが、一方で、受渡失敗時を考えると（注：タイムアウト時間までは当該資産がロックされるため）流動性の効率的な利用を阻害することとなる。他の方法としては、何らかのインセンティブを与えることにより、銀行 B の代わりに、第三者に第２の証券振替指図を送信させることが考えられる。この場合、銀行 B により署名を付された第２の証券振替指図は、銀行 B 以外の第三者が変更することはできない。

【図表 5】複数台帳 HTLC 方式におけるフェイルの例（リスクシナリオ①、ステップ 5 において中断された場合）



【図表 6】複数台帳 HTLC 方式におけるフェイルの例（リスクシナリオ②、ステップ 7 において中断された場合）

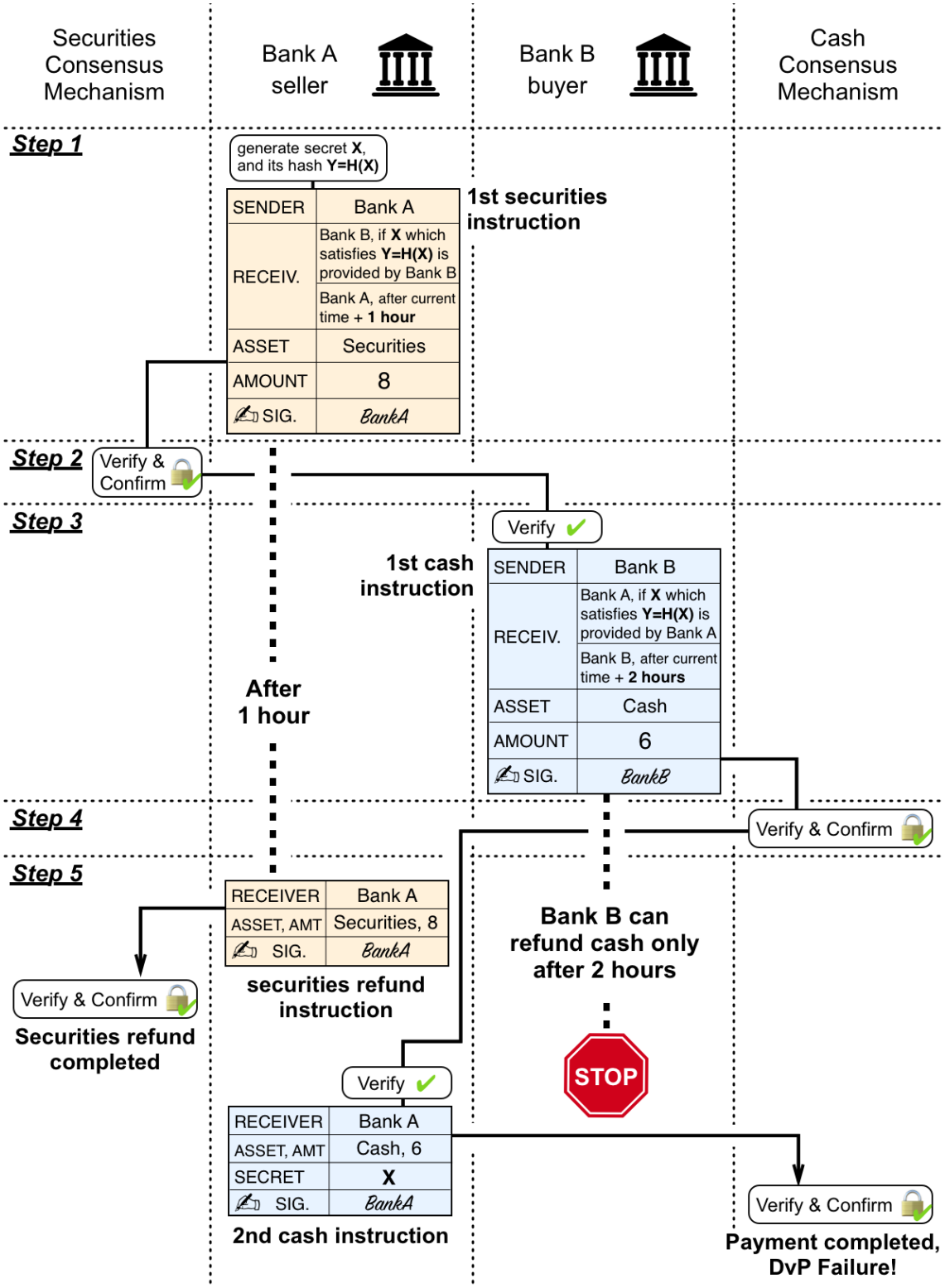


タイムアウト時間の非対称性

複数台帳 HTLC 方式においては、乱数の生成者が引き渡す資産側のタイムアウト時間（例えば、図表 4 における証券の 2 時間というタイムアウト時間）は、他方のタイムアウト時間（同じく図表 4 における資金の 1 時間というタイムアウト時間）よりも長くなければならないという点に留意が必要である。この非対称性が満たされない場合、一方の取引当事者が資金と証券の両方を取得することが可能となる。

例えば、証券のタイムアウト時間を 1 時間、資金のタイムアウト時間を 2 時間とした場合（図表 7 を参照）、銀行 A は第 2 の資金振替指図を 1 時間以内に送信しないことにより、資金と証券の両方を取得することが可能となり得る（注：図表 7 は図表 4 とはタイムアウト時間を逆にしている。この場合、銀行 A は証券を取り戻してから資金を取得するというステップを踏むことが可能となる。具体的には、銀行 A は、証券を取り戻すことのできる 1 時間経過後に、第 2 の証券振替指図で証券を取り戻し、併せて、第 2 の資金振替指図で資金も得ることができる。この間、銀行 B は 2 時間経過しないと資金を取り戻すことができないため、銀行 A に対処できない）。なお、ここでは、証券 DLT ネットワーク上の時間経過速度と資金 DLT ネットワーク上の時間経過速度は全く同じであると仮定している。

【図表 7】複数台帳 HTLC 方式におけるフェイルの例（タイムアウト時間が適切に設定されなかった場合）



ボックス 1

今次調査で用いた 3 つの DLT 基盤について

Stella 第2フェーズでは、Corda、Elements、Fabric という 3 つの DLT 基盤を用いてプロトタイプの実装を行った [28]。ここでは、各基盤の主な特徴について記載する。

Corda：Corda は、既知の参加者間でのやり取りを念頭に、法的な契約物とその処理を、記録・管理・自動化するために構築された基盤である。許可型ネットワークであり、公開鍵基盤（public key infrastructure）に基づいて認証およびアクセス制御を行う。カナダ中央銀行の Project Jasper[29]や、シンガポール通貨庁の Project Ubin[30]など、他の中央銀行のプロジェクトでも用いられている。

Elements：Elements は、①複数種類の資産を生成し、取引内容（取引額や取引に用いた資産の種類）を第三者に分からないように暗号化する機能や、②他のブロックチェーンとの間で資産のやり取りを可能とするサイドチェーン機能などをサポートする [31]。

Fabric：Fabric は、モジュラー型（組合せ型、modular）アーキテクチャに基づき、エンタープライズ向けに構築された DLT 基盤である。Corda と同じく、許可型ネットワークであり、Stella 第 1 フェーズ [32]で使用された。シンガポール通貨庁の Project Ubin や、ブンデスバンクとドイツ取引所グループの共同プロジェクト [33]でも用いられている。

各 DLT 基盤は、参加者間の信頼度、コンセンサスアルゴリズム、トランザクションの処理フローなど多くの点でそれぞれ特徴を備えている。これらの特徴のうち、DLT ネットワーク上で DvP を設計するにあたっては、下記の点などに留意する必要がある。

<u>決済のファイナリティ</u>
DvP においては、資産の受渡が完了した後は、その受渡は取消不能（irrevocable）であるという点が重要である。DLT 基盤について考えると、トランザクションを処理した結

[28] 今次調査では、Corda release-V2、Elements v2.14.1.1、Fabric v1.1.0-alpha を用いた。
[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%20RTGS.pdf
[31] Segregated Witness（いわゆる「SegWit」）や Relative Lock Time は、最初に Elements で実装された後、他のブロックチェーンでも採用されるようになった。
[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

果生じる残高変動は、台帳に記録された以降は、取消不能でなければならない。決済のファイナリティ（注：ここでは、記録の確定性を指し、法的な論点は含まない）は、コンセンサスアルゴリズムによって変わり得る。Corda と Fabric においては、確定性は保証されているものの、Elements ではコンセンサスアルゴリズムとして proof-of-work を用いる場合は保証されない。確定性を得るための一つの方法としては、マイニングを 1 台のノードに限定する（この場合も各ノードはそれぞれ台帳を保持する）ことが考えられる[34]。

共有される情報の範囲

複数台帳 HTLC 方式は各 DLT 基盤のプライバシーモデルの影響を受ける。Corda では、ノード間の通信は取引毎にその取引当事者に限定され、結果的に取引当事者の間でのみ「レコード単位」で台帳を共有することになる。明示的に設計しない限りは、各参加者は参加者全体で構成される台帳の一部のみを保持することになり、どの参加者も台帳の全体像を把握することはできない。一方で、Fabric は、台帳を分離し機密性を高めるチャネル機能をサポートしているが、同一チャネル内においてはノード間の通信はブロードキャストであり、参加者は「チャネル単位」で台帳を共有することになる。Elements では、ノード間の通信は基本的にブロードキャストのみであり、参加者はただ一つの台帳を共有する。そのため、Elements において適切なレベルのプライバシーと機密性を確保する場合は、複雑な暗号手法を用いる必要がある。

データ構造

実際の処理フローは各 DLT 基盤のデータ構造にも影響を受ける。Corda と Elements は UTXO（未使用トランザクションアウトプット、Unspent Transaction Output）モデル[35]を使用しており、参加者が作成する各トランザクションに基づいて、次のステップへと処理が進む。一方で、Fabric の台帳は、入力順にトランザクションをブロックにまとめたブロックチェーン（blockchain）と、キーに紐づくバリューとして現在の状態を保持するキーバリューストア型の状態データベース（state database）という 2 つの台帳からなる。Fabric において、UTXO モデルは適用可能であるものの、一般的にはアカウント毎にステータスを保持するアカウントモデルが用いられ、状態データベース上の状態遷移として処理は表現される。そのため、参加者からの指図に基づいて、状態データベース上のステータスが遷移することにより、次のステップへと処理が進むことになる。

[34] 今次調査では、マイナーを 1 台に限定した上、処理の迅速性を図るため、ブロック生成間隔を短い時間（例えば、2 秒間）に設定した。また、各ノードをホワイトリスト（whitelisted peer）として設定した。詳細は原文の別添 5 を参照。

[35] UTXO モデルは、各トランザクションが異なる未使用トランザクションをインプットとして用いることができるため、並列処理と一般に相性が良いとされるが、残高がどれだけ多くの UTXO に分かれているかにより、その効果は大きな影響を受ける。

5. DLT 環境下での DvP 実現方式に関する分析

本章では、第 4 章の処理フローに基づき、単一台帳方式と複数台帳 HTLC 方式の主な特徴について分析を行う。特に、証券決済に付随する以下の潜在的なリスクについて、それぞれの方式毎に分析を行う。

元本リスク：２つの資産の受渡には、取引当事者の一方が取消不能なかたちで資産を渡したにも関わらず、受取予定の資産を受け取れないリスクが伴う[36]。

再構築コスト・リスク：取引当事者は、決済が完了するまでの間、元の取引と同一の取引を新たに締結し直す（再構築する）ことで、例えば再構築までの証券価格の変動等が発生するコストを被るリスクに晒されている。

流動性リスク：フェイルが生じると、流動性リスク（取引相手が、金融債務の履行に必要な資金または証券を、決済日に手当てできない<ただし、将来のある時点では履行できる可能性がある>リスク）が顕現化する[37]。また、決済プロセスにおいて、流動性（資金または証券）のアベイラビリティ次第では流動性リスクが顕現化することもある。

本章では、処理時間、秘匿性（プライバシー）、ネットワーク間の相互依存性、インフラ全体のデザインといった他の論点についても触れる。この間、法的な論点は、分析の対象外としている。DLT 環境下での DvP 決済やスマートコントラクト活用の法的な取扱いや、法的なファイナリティの要件等については、さらなる検討が必要である[38]。

---

[36] 元本リスクおよび再構築コスト・リスクは、信用リスクである。信用リスクは、取引相手が、期日または将来のいずれかの時点で金融債務を完全には履行できなくなるリスクを指す（FMI 原則 2.5 を参照）。

[37] FMI 原則 2.6 を参照。

[38] 現時点で、DLT 環境下で決済のファイナリティがどの時点で得られるかについては、一般的な法的定義はない。DvP によって、資金の受渡が行われる場合に限り証券の受渡が行われることが保証されたとしても、DLT 環境下では、DvP 決済のファイナリティには不確実性が存在する。言い換えれば、一方の台帳（例えば、資金の台帳）における振替が取消不能（ファイナル）ではなく、他方の台帳（例えば、証券の台帳）における振替が完了した後で巻き戻される可能性があれば、当事者の一方（例えば、証券の売り手）は、元本リスクに晒されることになる。また、DLT 基盤がコンセンサスアルゴリズムとして proof-of-work を用いる場合、当該基盤における決済処理は確率的である。さらに、一部の DLT 基盤では、ハードフォークのように、過去の取引履歴が巻き戻され、ある時点以降から履歴が変更されてしまう可能性もある。これらに加え、無券面化された証券を DLT 環境上で発行・振替することの法的有効性も、DvP の実現にあたって論点となり得る。

５．１　DvP の実現

取引の両当事者が 2 つの資産を受渡する際には、必然的に元本リスクが伴う。DvP の実現にあたっては、元本リスクを除去するために、一方の債務のファイナルな決済は、それと結び付けられた債務のファイナルな決済が行われる場合にのみ実行される必要がある。また、取引当事者の一方ないし両方が、処理フローのいずれかのステップを実行しなかった場合は、資金と証券の受渡は失敗し、それぞれ元の所有者に戻る必要がある[39]。

単一台帳方式では、処理フロー（第４章１節を参照）のいずれかのステップが完了しない場合であっても、取引の両当事者が元本リスクに晒されることはない。これは、資金と証券の振替が、処理フローの最後のステップまで行われず、また、実行される際には 1 つのトランザクションとして両方の振替が同時に処理されるためである。

複数台帳 HTLC 方式では、処理フローの一部のステップが完了しない場合、取引当事者の一方は元本リスクに晒される可能性がある。具体的には、一方の取引当事者（第４章２節、図表４のステップ７、８における銀行 B）がタイムアウト時間内（2 時間以内）に証券を取得しなければ、既に資金を受け取った他方の取引当事者（銀行 A）が証券も取り戻すことが可能となる[40]。こうしたリスクに対処する方法は存在するものの[41]、元本リスクの削減にあたっては、本質的には、取引当事者が各プロセスのタイミングを管理することが前提となっている。

５．２　流動性の利用効率

決済プロセスにおける流動性の利用効率は、決済方法（グロスベースまたはネットベース）、日中流動性の利用可否、参加者が振替指図を送信するタイミングなど、様々な要素によって影響を受ける[42]。DvP の文脈では、処理フローにお

---

[39] フェイルは、このほかにも、当事者間の通信途絶や、処理フローの予期せぬ中断によっても生じ得る。ただし、フェイルの発生は、必ずしも、DLT 環境下における DvP の仕組みが機能していないことを意味するものではない。

[40] この特定のシナリオにおいて DvP が実現しないのは、HTLC では、第２の資金振替指図と第 2 の証券振替指図の完了有無が完全には連動しないことによるものである。言い換えると、複数台帳 HTLC 方式自体は、両振替指図の完全な連動を保証するものではないため、取引当事者双方が処理ステップを確実に完了させなければならない。

[41] 脚注 27 を参照。

[42] このほか、DLT 基盤のコンセンサス形成メカニズムにおいて、当該振替指図が処理・決

ける資金や証券のロックの有無およびその時間も、流動性の利用効率に影響を及ぼし得る。

単一台帳方式では、資金と証券の振替が 1 つのトランザクションとして同時に実行される場合、資金と証券のロックを事前に行う必要はない。事前にロックを行わない場合、流動性の効率的な利用が妨げられることはない。他方で、フェイル（第 4 章 1 節、図表２のステップ４における残高不足によるフェイル）を防ぐ観点から、事前にロックを行うという考え方もあり得る。このように、流動性を効率的に利用することと、決済を確実に履行することにはトレード・オフの関係が存在する。

これに対して、複数台帳 HTLC 方式では、DvP の実現のためには、処理フローの早い段階（第 4 章 2 節、図表 4 のステップ１、３における第 1 の証券振替指図と第１の資金振替指図の作成段階）で資産をロックすることが必要である [43]。ステップ１、３において資金と証券のロックが行われなければ、ステップ６で資金の振替が完了したにも関わらず、取引当事者の一方（ここでは銀行 A）が十分な証券を保有しておらず、ステップ８の証券振替が実行されないという事態が生じかねない。このため、流動性の利用という観点からは、複数台帳 HTLC 方式の方が、単一台帳方式と比べて効率性が低くなる傾向にある。特に、資産のロックが行われたものの、決済が未了となり、タイムアウト時間経過後に元の所有者に戻る場合には、こうした傾向が強くなる。

５．３　処理時間

本報告書において、DvP 決済の処理時間とは、①取引当事者の一方が証券または資金の振替指図を作成した時点から、②資金と証券双方の振替が台帳に記録された時点までの経過時間を指す。

複数台帳 HTLC 方式では、コンセンサス形成メカニズムによる検証・確認作業等を含む、多くの処理ステップを伴うため、単一台帳方式よりも処理フローはより複雑となる。さらに、取引当事者が、あるステップが完了したことを認識し、次のステップの準備を行うのにも時間を要する。こうしたことから、複数台帳 HTLC 方式の方が、単一台帳方式よりも処理時間が長くなると考えられる。

済される順番も流動性の利用効率に影響を与え得る。

[43] 各 DLT 基盤を用いたプロトタイプ実装にあたってのロックの扱いについては、原文の別添 4、5、6 を参照。

今次調査では、単一台帳方式と複数台帳 HTLC 方式の処理時間を比較する観点から、複数の DLT 基盤を用いて実験を行った。いずれの DLT 基盤においても、単一台帳方式の一連の処理プロセスは数秒程度（注：一桁の秒数）で完了したが、複数台帳 HTLC 方式では、最大 3 倍の時間を要した [44]。ここで、処理時間は、（ア）取引の両当事者が情報伝達を行い振替指図を作成・署名する時間（注：クライアント側の処理時間）と、（イ）証券・資金の振替にかかる 1 つのトランザクションが処理される時間（注：DLT ネットワーク側の処理時間）とに分けることができる。単一台帳方式と複数台帳 HTLC 方式のいずれにおいても、処理時間の大半（ある場合には処理時間の最大 97%）はコンセンサス形成メカニズムにより振替指図が検証・確認される時間（上記の（イ）の時間）であり、取引当事者間で情報伝達が行われ、双方が振替指図を作成・署名する時間（上記の（ア）の時間）は極僅かであった [45]。

５．４　秘匿性（プライバシー）

DLT 基盤において、取引内容をどこまで公開するかについては、様々なやり方がある。例えば、Elements や Fabric においては、関係する全ての参加者で情報が共有される（注：Fabric では同一チャネル内の参加者を指す）一方、Corda では、取引関係者の間でのみ情報が共有される [46]。

---

[44] 一定時間内に処理可能な DvP 決済件数は、単一台帳方式では、複数台帳 HTLC 方式の 2 倍以上であった。ただし、複数台帳 HTLC 方式では、2 つの DLT ネットワークそれぞれについて、2 つのトランザクションの処理が必要となる点に留意。

[45] 処理性能は、実験環境やパラメータ等の影響により大きく変わり得る。実験環境については、ノード 1 台につき、メモリ 7.5GB、ディスク容量 8GB の Ubuntu（16.04.1 LTS 64 bit）サーバを割り当てた。パラメータについては、基本的にはデフォルトのパラメータを３つの DLT 基盤について用いた。プロトタイプは、複数種類の証券を扱えるよう設計されていたが、処理時間の計測にあたっては 1 種類の証券のみが用いられた。併せて、複数台帳 HTLC 方式の場合は 2 つの DLT ネットワークはともに同じ DLT 基盤を用いて構築された。取引の両当事者は、同一サーバ上で動く 2 つのプロセスとして設計されたため、取引当事者間での通信時間は僅少であった。UTXO の扱いについては、Corda では、デフォルトのトランザクション選択アルゴリズムとソフト・ロック機能（soft-locking）を用いた。一方で、Elements では、異なるトランザクション間で同じインプットを使用しないよう、トランザクション毎に異なるインプットを直接指定した。Fabric では、口座残高を単一のキーとして表現しないように、ある口座の残高は状態データベース上における差分（delta）の合計として管理した。詳細は原文の別添４、５、６を参照。

[46] Corda では、情報が一か所に集約されることはない。その代わり、各ノードは自身が取引当事者となる取引内容のみを持つデータベースを個別に保持している。その結果、各ノードは、ノード全体で構成される台帳の一部のみを保持することになり、どのノードも台帳の全体像を把握することはない。Elements では、各ノードが全く同じ台帳を保持している。Fabric では、各ノードがチャネル単位で同じ台帳を保持している。また、ある 1 つの DLT ネットワークにおいて、複数のチャネルを跨いだトランザクション全体は orderer が

単一台帳方式では、取引の両当事者は、その内容を検証し、電子署名を付す必要がある。DLT 基盤如何に関わらず、これらの当事者間における情報伝達は、台帳上には記録されないため、コンセンサス形成メカニズムに送信するまでの処理ステップ（例えば、取引の両当事者による検証や署名など）は、どの DLT 基盤でも、第三者に知られることなく行うことができる。

これに対して、複数台帳 HTLC 方式では、資金と証券の受渡をブロックする際に用いるハッシュ値のシークレットは、DLT ネットワークの参加者全体で共有される必要があり得る。例えば第 4 章 2 節の図表 4 におけるステップ 7 を考えると、証券の売り手（A 銀行）が、資金を受け取るためにシークレットを掲示することではじめて、証券の買い手（B 銀行）は当該シークレットを取得し、証券を受け取ることが可能となる。こうしたシークレットの共有にあたっては、A 銀行が資金を受け取るために用いたシークレット（ないし、シークレットを含むトランザクション）は、当該トランザクションに B 銀行は出し手としても受け手としても直接関与していないにも関わらず、必ず B 銀行と共有されなければならない。そのため、複数台帳 HTLC 方式の実現にあたっては、場合によっては、A 銀行によって作成されたシークレットは DLT ネットワーク全体で共有される必要性が生じる[47]。こうした点は、各 DLT 基盤が採用するプライバシーモデルによっては、新たな課題をもたらす可能性がある[48]。

５．５　ネットワーク間での相互依存性

複数台帳 HTLC 方式の下では、２つのネットワーク間において、何らかの接続や、やり取りは存在しない。2 つのネットワークは、独立しながら処理を行う

保持している。

[47] 銀行 A の用いたシークレットは銀行 B に伝えられる必要がある。銀行 A が銀行 B に伝えない場合、例えば、信頼のある主体が銀行 A の代わりに銀行 B に正しくシークレットを伝えることが考えられる。ただし、このことは、当該主体が取引の両当事者とその取引で用いたシークレットを知り得る立場にあることを意味する（注：原文の別添 6 に記載の通り、Fabric ではクエリを用いて銀行 B のみにシークレットを渡す方法を用いた）。第三者を一切用いない場合、一般的には、ネットワーク内の全参加者へブロードキャストすることが考えられる。この場合、シークレットはネットワーク参加者全員に知られることとなる。ただし、たとえ他の参加者がシークレットを受け取ったとしても、ハッシュ値でロックする際に指定した銀行 B しか、当該シークレットを用いてブロックされた証券を受け取ることができない点に留意が必要である。

[48] 今次調査で用いた Corda（release-V2）においては、アーキテクチャ上の制限から、第三者を介さずにシークレットを共有することは実現できなかった。また、タイムアウト機能についても、その指定する長さに制限が存在した。詳細は原文の別添 4 を参照。

ことになるが、一方のネットワークの信頼性やパフォーマンス等が、他方のネットワークに影響を及ぼす可能性がある。

さらに、２つのネットワークの時刻が完全に一致している必要はないものの、①これらのネットワークにおいて、タイムアウト時間の非対称性（第４章２節、図表 4 において、証券のタイムアウト時間の方が、資金のタイムアウト時間よりも長いという非対称性）の要件を満たすために、時間の経過速度が予め予測可能でなければならないことと、②想定し得る予測誤差を吸収できる程度に、タイムアウト時間が長くなければならないことの二点が重要である。この意味において、タイムアウト時間の設定にあたっては、取引の両当事者が、必要な処理ステップを踏むために十分な時間を見込んで設定しておくことが重要となる。ただし、一方で、タイムアウト時間が長ければ、資金と証券の受渡が失敗した場合に、資金や証券が元の所有者に戻るまでの時間も長くなる点には留意する必要がある。

５．６　インフラ全体のデザイン

今次調査では、資金と 1 種類の証券との間における DvP メカニズムに焦点を当てた。しかし、市場インフラには、国債、社債、株式等を含めた様々な種類の証券があるほか、約定確認、ネッティング、清算・決済等を含め、約定後には複数の事務処理が行われる。このため、DLT による DvP 実現方式を検討するにあたっては、市場インフラサービス全体をどのようにデザインするかを考える必要があろう。

例えば、単一台帳方式のように、様々な種類の証券を単一のネットワーク上の台帳で管理し、これらの資産間における DvP 決済を全て同じ基盤の上で行った方が効率的な場合も考えられる。この場合、追加的な決済機能（例えば、流動性節約機能、口座間でのネッティング、担保としての証券の効率的な利用）は、単一の DLT ネットワークに実装すれば済むと考えられる。さらに、様々な資産が一つの DLT ネットワーク上で管理されていることから、各参加者においては、様々な資産の残高を確認したり、管理したりすることが容易になることも考えられる。

しかしながら、こうした便益は、基盤のスケーラビリティ、柔軟性、頑健性といった市場インフラの安定性と比較考量する必要がある。

様々な種類の資産を単一のネットワーク上の台帳で管理するような場合、2 つの資産だけを管理するような場合と比較して、取引件数が増加することが予想され、それに耐え得るパフォーマンスが必要となる。また、一部の資産の取引が、ネットワーク全体のリソースのほとんどを費消してしまうような事態になると、それ以外の資産の取引について、遅延等が発生する可能性もある。さらに、資産の数や参加者が増えるに連れ、取引量も増大することから、サービスの可用性（availability）をしっかりと担保するために、これら参加者レベルでのサイバーセキュリティ対策も、より一層重要となり得る[49]。

インフラデザインという観点からの複数台帳 HTLC 方式の最大のメリットは、参加者の自己責任の下、ネットワーク間を接続することなく、DvP を実現できる点にある。理論的には、HTLC が 2 つのネットワークによってサポートされており[50]、これらのネットワークに取引の両当事者が参加している限り[51]、ネットワーク間での取り決めや業務手順を設けることなく、DvP を実現することが可能である。今次調査では、2 つの分離された DLT ネットワーク――より具体的には、①同一の DLT 基盤、ないし②複数の DLT 基盤（例えば、Elements のような参加者を限定しない DLT ネットワークと、Fabric のような参加者を限定する DLT ネットワーク）において、複数台帳 HTLC 方式が正しく機能することが確認された。

[49] 上記の課題に加え、現行システムが DLT ネットワークと接続されていない場合や、こうしたネットワークに統合されていない場合には、複数のシステム上に資産を配分しなければならないことになり、流動性の効率的な利用を阻害する可能性がある。特に、複数のシステム上で証券が別々に管理されている場合には、証券貸出や担保管理の点で課題が生じる可能性もある。

[50] ここには非 DLT のネットワークも含まれる。HTLC は、スマートコントラクトの一類型に過ぎないと言え、その利用にあたっては、P2P ネットワークやコンセンサスアルゴリズムといった DLT を構成する他の要素技術を用いる必要性は必ずしもない。

[51] 技術的には、1 つの DvP 決済を 2 者以上で行うことも可能である。