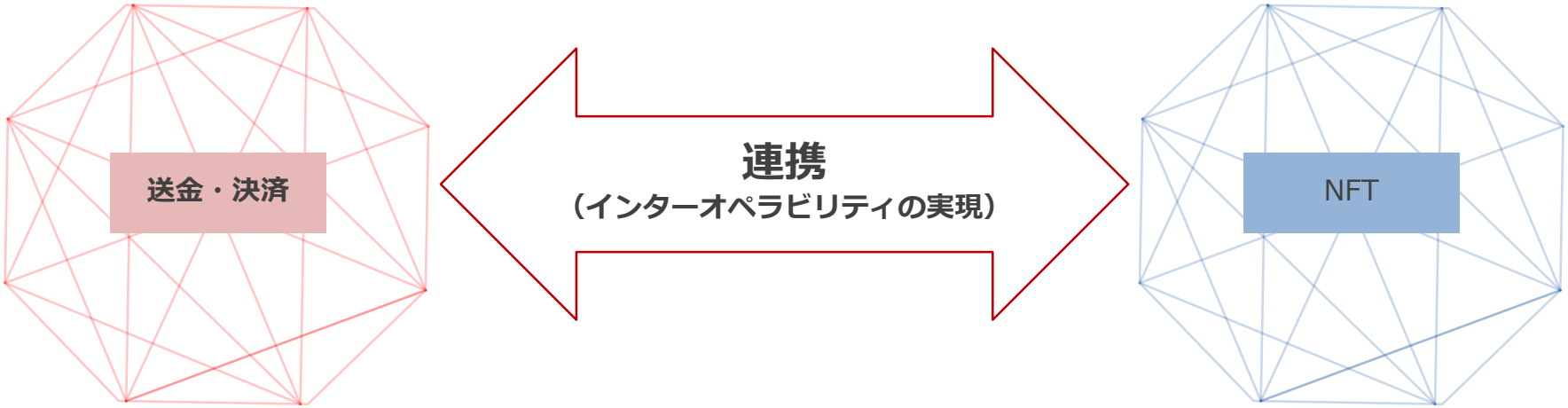
エンタープライズ領域のブロックチェーン活用における
インターオペラビリティ実現のためのプラクティス

2023.6

みずほリサーチ&テクノロジーズ株式会社
株式会社Datachain
SBI R3 Japan株式会社

異なるブロックチェーン間のインターオペラビリティ実現への取組み背景

- ブロックチェーンは特定の組織等に依存せずに取引やデータ保管が可能になる仕組みである。ブロックチェーン上で様々なサービスを提供可能であり多様なユースケースでの活用が世界中で進行している。
- たとえば、金融領域においては暗号資産に代表される送金・決済手段としての利用や証券取引サービスの提供が始まっており、非金融領域においては、食品や工業製品等のサプライチェーン情報をブロックチェーン上に記録して不正防止等に活用するサービスや電子契約、資格証明等の様々なサービスが提供され始めている。
- 現在、日本国内外に様々なブロックチェーンのネットワークが存在しており、これらのサービスは主に個別のネットワーク内でサービス展開がされている状況である。
- 上記状況を踏まえると、異なるブロックチェーン間のインターオペラビリティを実現し高度なサービスを提供することが重要になってくると考えらえる。既にブロックチェーン間連携のための様々な方法が提唱され稼働も始まっており、今後更なる進展が想定される。



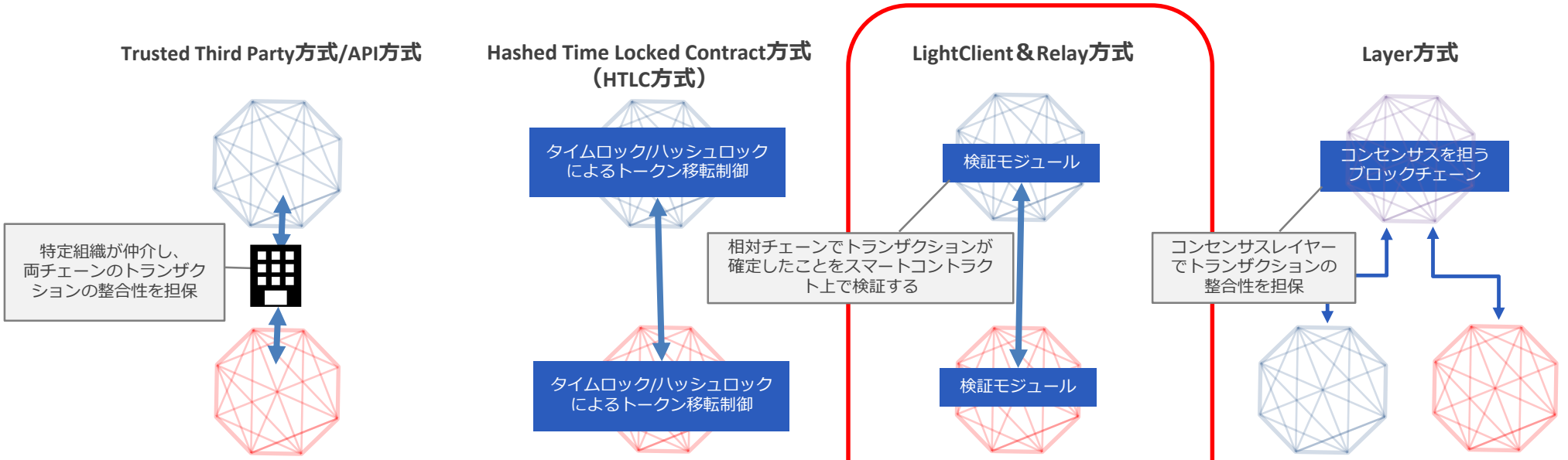
本書の目的

- 前述の背景を踏まえ、みずほリサーチ&テクノロジーズ株式会社、株式会社Datachain、SBI R3 Japan株式会社の3社では、異なるブロックチェーン間のインターオペラビリティ実現へ向けた取り組みを2022年度に共同で実施。
- 主要エンタープライズブロックチェーン基盤（以下、BC基盤）間のインターオペラビリティを念頭に、特にGoQuorum、Corda間のインターオペラビリティについて金融・非金融の想定ユースケースを設定し、機能・非機能要件の実現方法について検討を行った。
- 機能・非機能要件は各開発プロジェクトにより水準が異なるものの、本取組みによって得られた機能・非機能要件を実現するために有用であると考えられる一般的なプラクティスを本書では記載。

（参考）クロスチェーン技術に関する技術連携開始における各社リリース
https://www.mizuho-rt.co.jp/company/release/2022/crosschain1205.html
https://speee.jp/news/10700/
https://sbir3japan.co.jp/crosschain-20221205/

本書が対象とするインターオペラビリティ実現方式

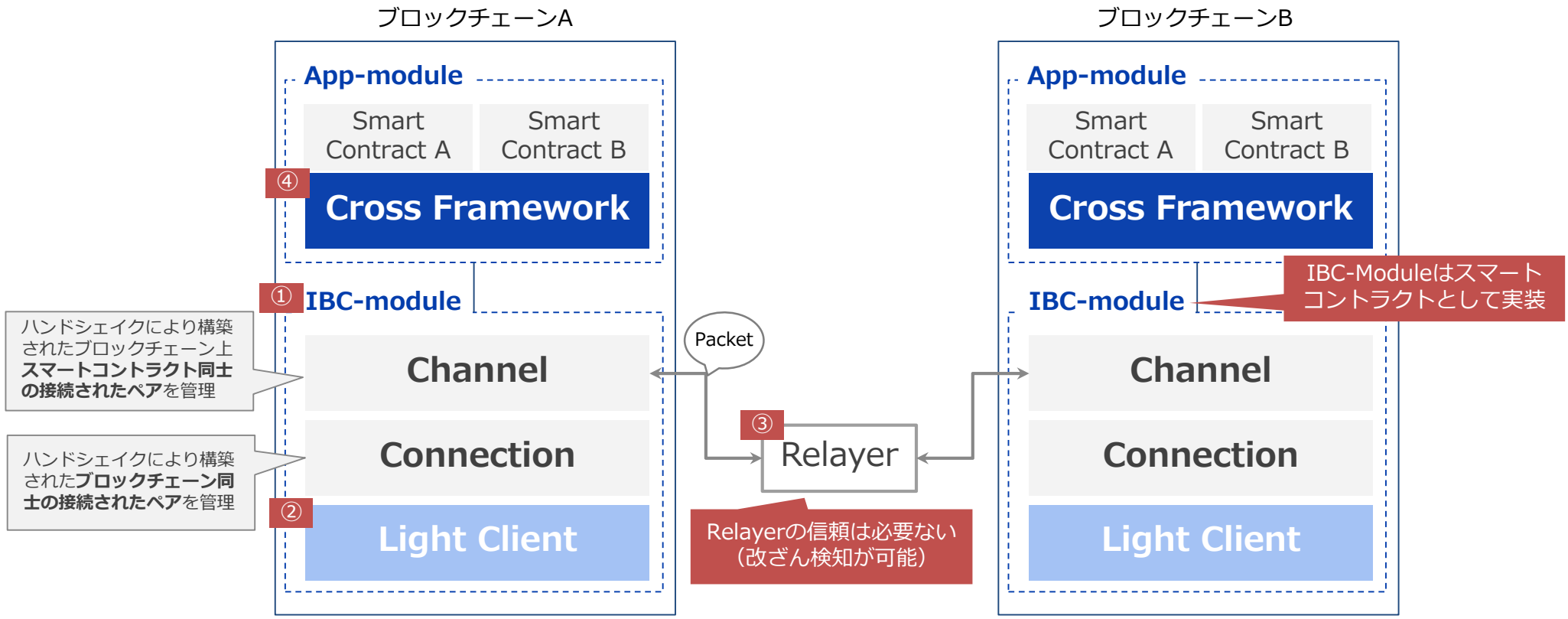
- インターオペラビリティは様々な実現方法が提唱・実装されているが、特定の組織等に依存せずに取引やデータ保管が可能になるブロックチェーンの特性を損なわないRelay方式を本取組みでは対象に選定。
- Relay方式を採用するインターオペラビリティ技術のうち、パブリックブロックチェーンのCosmosプロジェクトにて活用が進んでいるIBC Module(Inter Blockchain Communication Module)をベースとしてエンタープライズブロックチェーン基盤にも対応しているHyperledger YUIを本取組みで対象とする実現方式に選定。



	Trusted Third Party方式/API方式	Hashed Time Locked Contract方式（HTLC方式）	LightClient＆Relay方式	Layer方式
メリット	• 実現が容易	• 特定組織等への信頼が不要	• 特定組織等への信頼が不要 • 汎用性が高い	• 特定組織等への信頼が不要 • 汎用性が高い
デメリット	• 特定組織等への信頼が必要 • 信頼維持のために、システム/組織ガバナンスのコストが想定される	• ユースケースが限定的（例：トークン移転のユースケース）	• 技術的に発展途上・実装難易度が高い	• 特定組織等に依存しないようにコンセンサスチェーンを管理する工夫が必要 • コンセンサスレイヤーのチェーン停止が全チェーンに波及するリスクが想定される • コンセンサスレイヤーのチェーンに接続するためにコストが掛かるケースあり

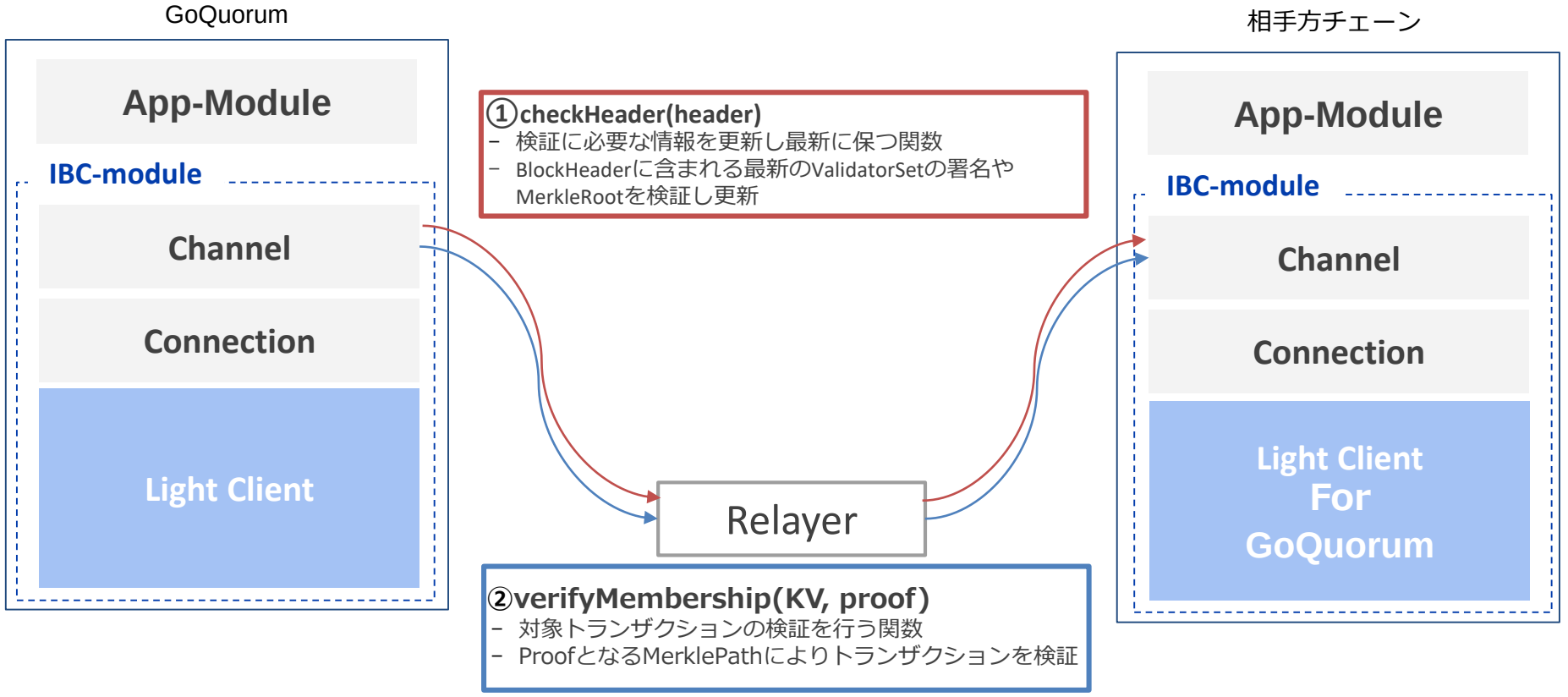
Hyperledger YUIのRelay方式によるインターオペラビリティ

- パブリックブロックチェーンのCosmosプロジェクトにて、Relay方式のインターオペラビリティのためのチェーン間コミュニケーション規格であるICS(Interchain Standards)が提唱されている。
- YUIはICSに準拠したIBC Module(下図①)をベースとしたプロジェクトである。
- 相手方のチェーンを検証するためのLight Client (LC、下図②)を両チェーンのIBC Moduleで持ち合うことで、通信仲介役となるRelayer(下図③)を通じて信頼できる第三者を設置せずにトランザクションを実行可能。
- 各チェーン上のスマートコントラクトからクロスチェーントランザクションを容易に実行するためのフレームワーク（Cross Framework、下図④）がYUI関連のOSSとして別途公開されている。



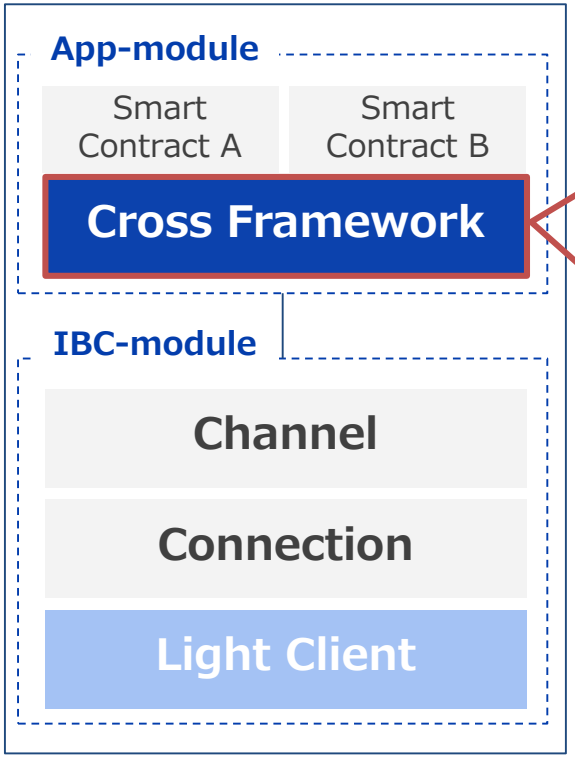
Light Client(LC)の役割ついて

- 「相手方のチェーンを検証する」とは、相手方から送付されたクロスチェーントランザクションが相手方でコンセンサスの取れたトランザクションであることを自チェーン上で検証することである。
- コンセンサス検証の仕組みはBC基盤毎に異なるが、主に以下の2ステップで実現する。
  - ① 検証に必要なBlockHeaderの検証・更新
  - ② 対象のトランザクションを検証するために必要なProofの送付
- GoQuorum (QBFT)の場合、主にBlockHeader内のValidator Setの署名やMerkleRootの検証・更新を行い(下図①)、対象のトランザクション及びMerklePathを送付する（下図②）ことで検証を実現する。



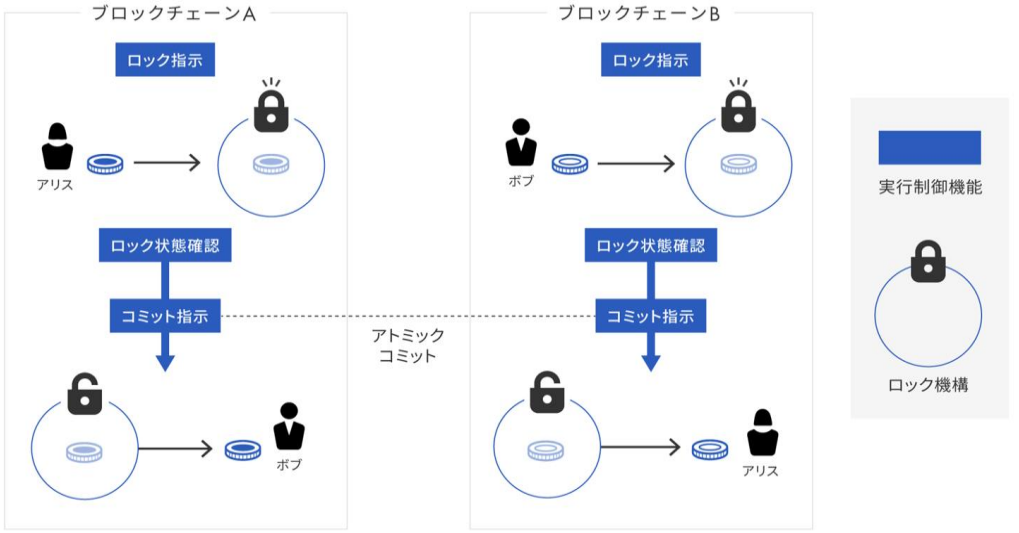
Cross Frameworkの役割について

- クロスチェーントランザクションを実現するためには、チェーン間のデータ整合性を満たしつつ、各チェーン上のスマートコントラクトで処理を実行する必要がある。
- Cross Frameworkは以下のような機能を提供し、開発者が業務ロジックの開発に集中しながらも安全にクロスチェーントランザクションを実行できるようにサポート。
  - ✓ Cross-Chain Atomic Swap：異なる両チェーンでアトミックコミットを実現する（例： DVPやPVP決済などの実行）
  - ✓ Cross-Chain Calls：相手方チェーンのContract関数を呼び出す（例：相手方チェーンの資産確認や決済前のロックなどの実行）



Cross-Chain Atomic Swapの例

①実行制御機能②ロック機構によりCross-Chain Atomic Swapが保証される。



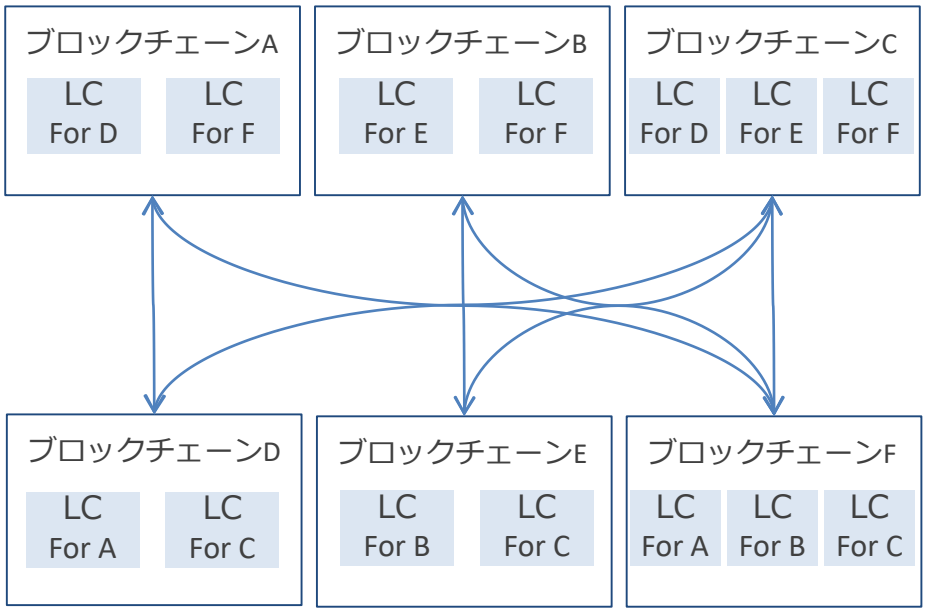
①実行制御機能
例：ブロックチェーンA及びB上のトランザクションのロック状態が確認できたら、双方のトランザクションをコミットする等

②ロック機構
例：トランザクションの実行過程においてロックを行い、コミット指示に基づいてアンロック（不備があれば元の状態に戻す）

Light Client Proxy方式とは

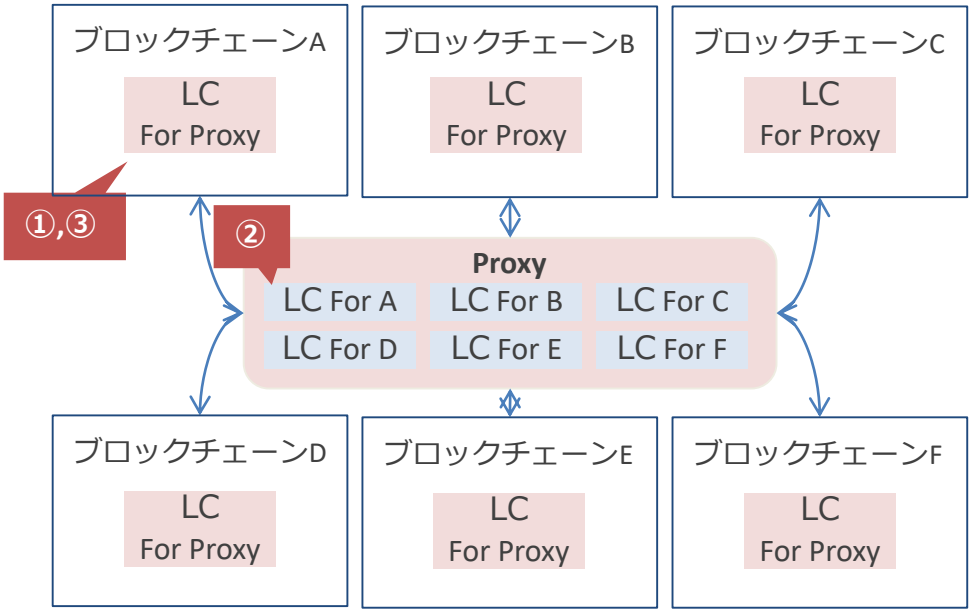
- 前述の基本的なLight Client実装方式においては、接続するブロックチェーンネットワークの数に応じてLCを実装する必要があり、拡張・保守性等に課題がある。
- 各ブロックチェーンのトランザクション検証を代替するProxyを設置するLight Client Proxy方式を導入することにより上記課題を解消できる。

基本的なLight Client実装方式



課題
① 接続するブロックチェーンネットワーク数分のLC実装が必要であり管理コストが増加する
② 各ブロックチェーンのスマートコントラクトの実装言語の違いによりLC実装に制約が発生するケースがある
(Go実装のHyperledger Fabric向けLCをHyperledger Besu上にSolidityで実装する難易度が高いなど)
③ LCによるオンチェーン検証に掛かるガスコスト
(パブリックブロックチェーンでより顕著)

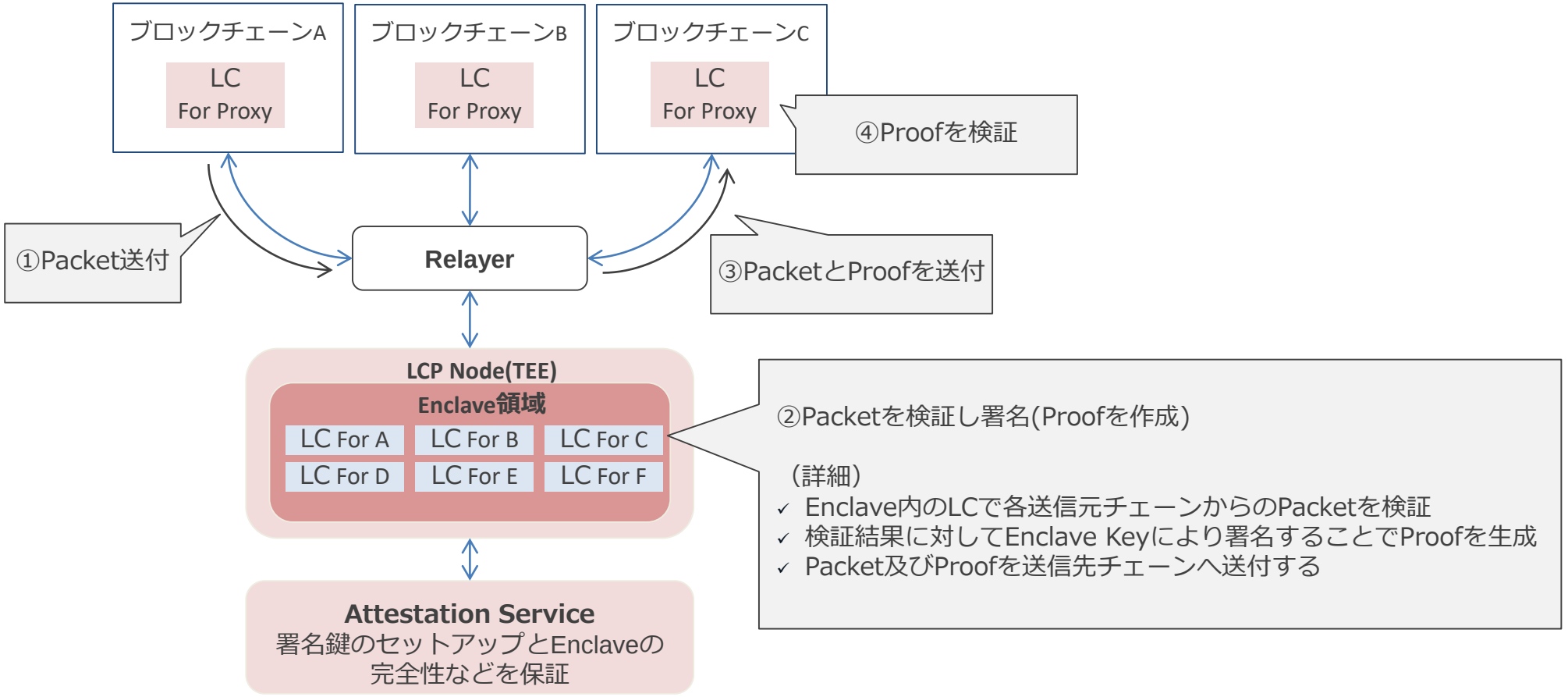
Light Client Proxy方式



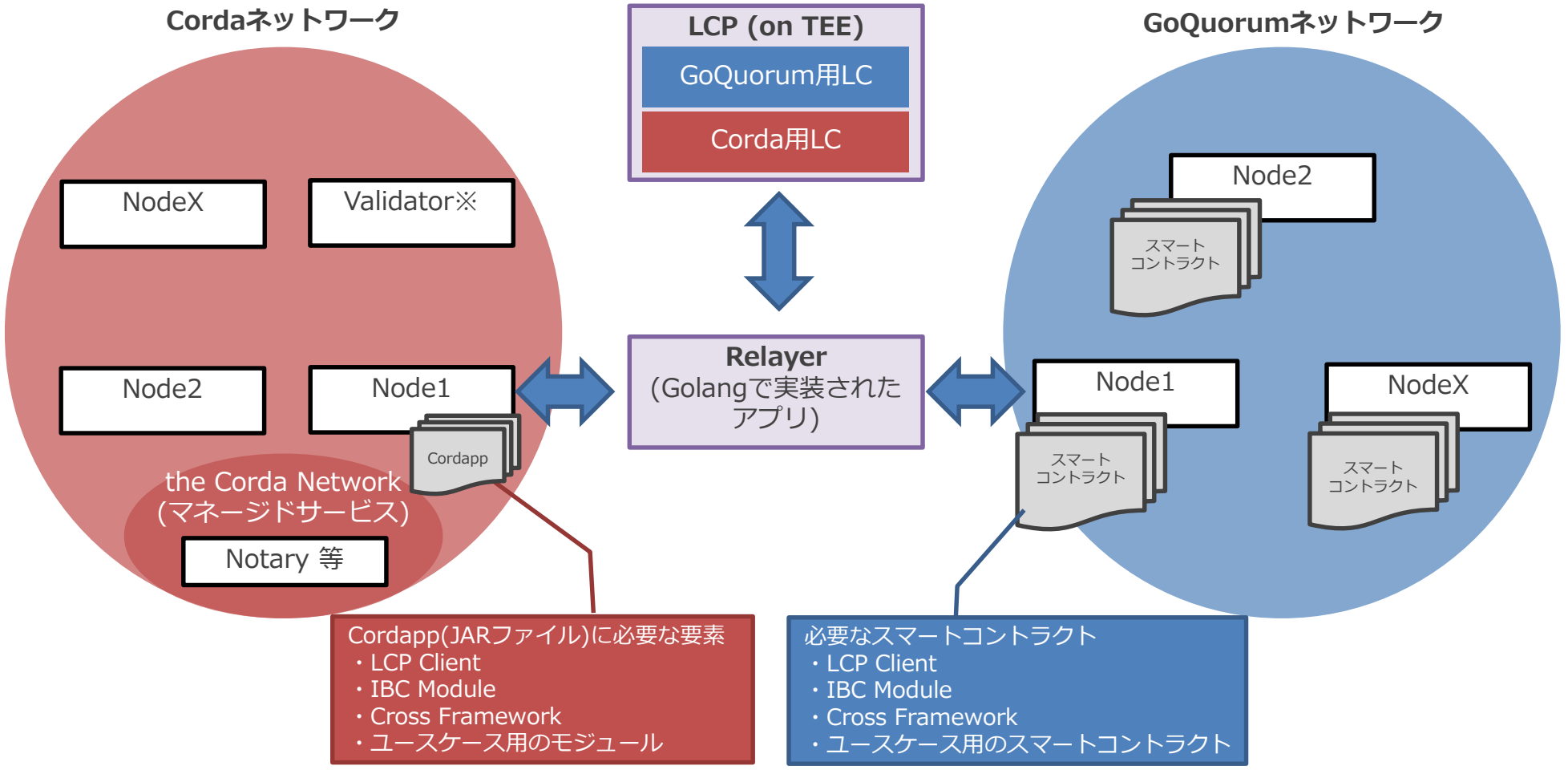
課題解決
① 各ブロックチェーンではProxyの検証結果を検証するLCのみを管理することで、管理コストを効率化
② 各ブロックチェーン上スマートコントラクトの実装言語特性を加味せずにLCを実装可能になるため、開発難易度が改善
③ ガスコストの高い検証工程をProxyに代替することでガスコストを効率化

LCP(on TEE)とは

- Light Client Proxy方式をTEE(Trusted Execution Environment)のEnclave領域内で処理させることで、セキュリティを確保した状態でトランザクション検証を実現する方式
  - ✓ TEEはメモリ上に暗号化された隔離領域を作り、暗号学的に処理の機密性と完全性を保証する技術。
- Light Client Proxy方式をBC上で処理させる方式が他案となるが、Proxy用BCのセキュリティ担保コストや性能不足リスクの課題があるため、本取組みではLCP(on TEE)を対象とする実現方式に選定。



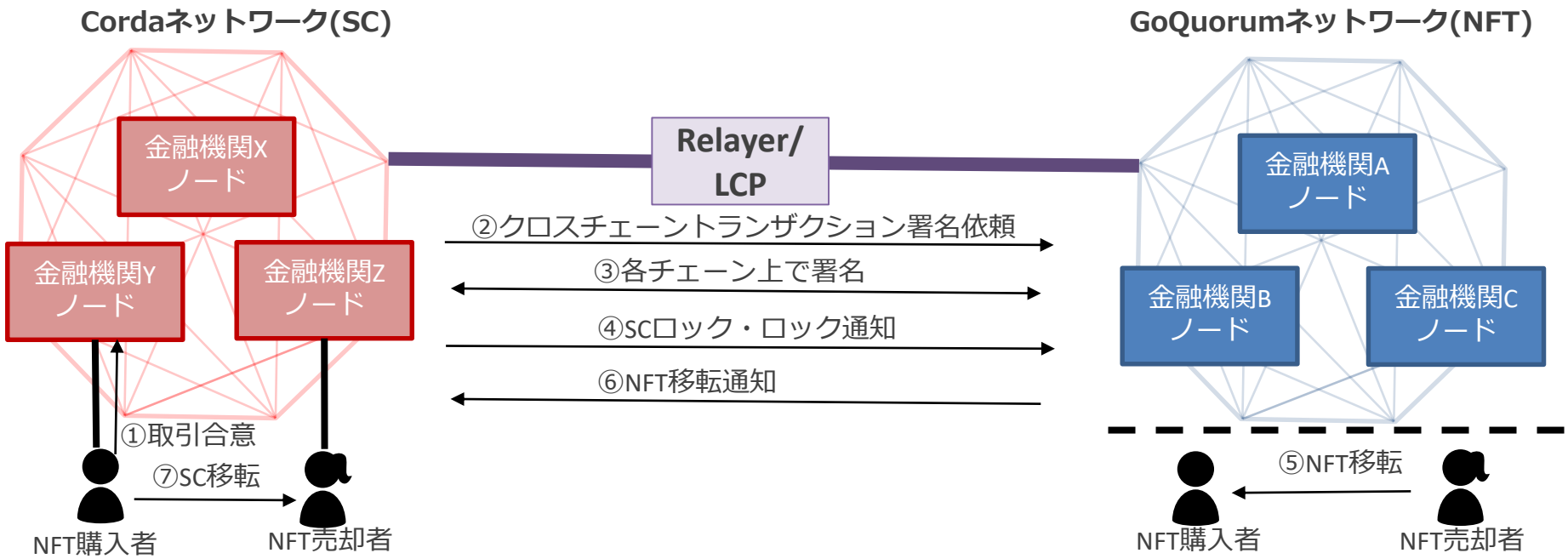
■ 本取組みにおけるアーキテクチャは以下の通り。



※このValidatorは、クロスチェーントランザクション実行時のCorda側トランザクションの正当性を強化するために、
取引当事者ノードとは別に設置される第三者のノードを指す。（取引当事者ノード間のトランザクションに対して第三者として署名を行う）

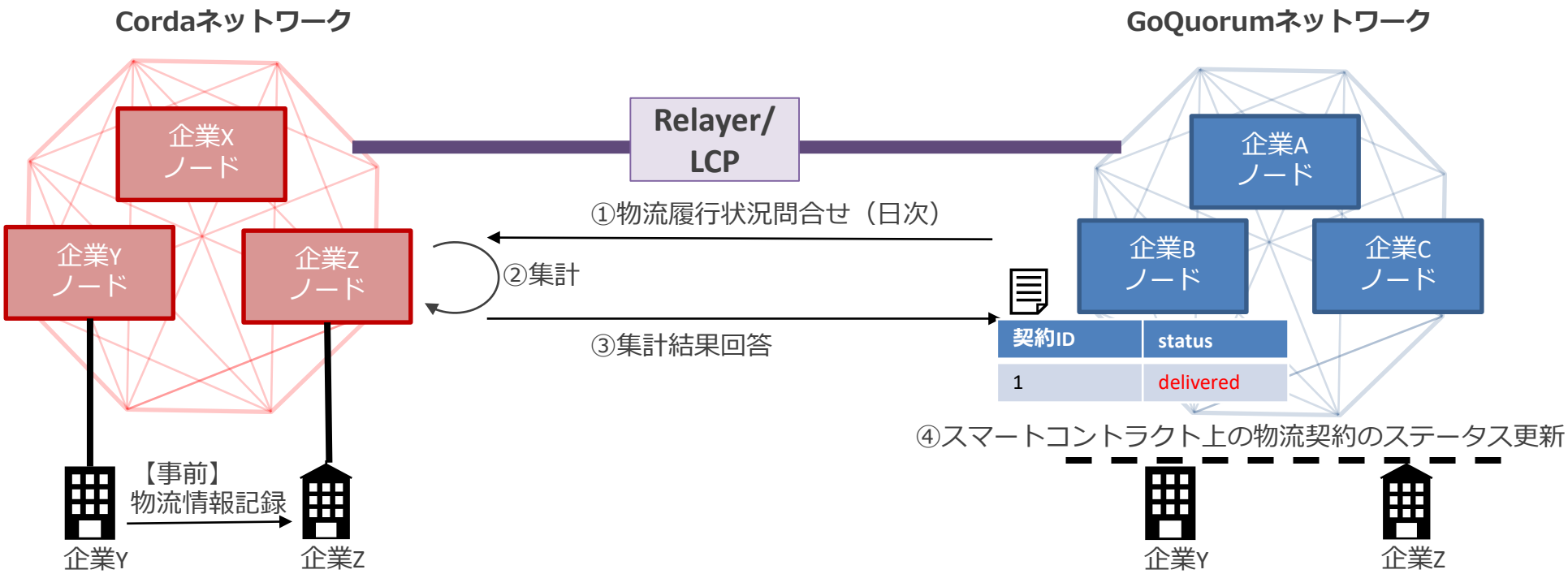
想定ユースケース①（金融）

- Corda上で発行・管理されているステーブルコイン(SC)とGoQuorum上で発行・管理されているERC721準拠のトークン（NFT）を用いたDVP決済を行う。
- 前提
  - ✓ ノードの保有者：金融機関等の企業・組織
  - ✓ Corda側エンドユーザー：企業のノード内で管理されノード内で一意のアイデンティティを持つ
  - ✓ GoQuorum側エンドユーザー：企業のノード内では管理されず、ネットワーク内で一意のアイデンティティ(アドレス)を持つ



想定ユースケース②（非金融）

- Corda上で管理されている物流情報を参照して、GoQuorum上で管理されている契約情報を日次で更新する。
- 前提
  - ✓ ノードの保有者：企業・組織
  - ✓ Corda側エンドユーザー企業：企業ノードに紐づく
  - ✓ GoQuorum側エンドユーザー企業：必ずしも企業のノードに紐づかない
  - ✓ 1つの契約に複数の物流が紐づいていることを想定（物流情報には契約IDが紐づいて記録されている）



■ 想定ユースケースで設定されうる機能要件や非機能要求グレードに基づく一般的な非機能要件のうち、本書ではクロスチェーントランザクションにおいて特別な考慮が必要と考えられる以下機能・非機能要件についてポイントやプラクティスを記載する。

機能要件	両ブロックチェーンでのアトミックな更新（想定ユースケース①）
	他方ブロックチェーンへの参照（集計）を伴う更新（想定ユースケース②）
	プライバシー
	アイデンティティ指定
非機能要件	可用性（耐障害性）
	可用性（回復性）
	性能
	運用・保守性(監視・異常検知時対応)
	運用・保守性(リリース運用)
	セキュリティ

■ チェーン間の責任分解について

- ✓ 各ブロックチェーンが独立したコンソーシアムであることを想定し、各ブロックチェーンの可用性や監視レベルは独立して設定されていると仮定。（互いに干渉不可）
- ✓ 本取組みでは以下を中心にプラクティスを記載
  - インターオペラビリティによって新たに追加される構成要素の可用性等について
  - 例外処理・異常発生時の運用対応など、各チェーンが協調して運用を行わなければならない事項について

■ 実行環境

- ✓ GoQuorum
  - 実行バージョン：v22.4.0
- ✓ Corda
  - 実行バージョン：Corda Enterprise 4.10
  - ネットワーク：the Corda Network(マネージドサービス)

the Corda Network(tCN)の制約について

tCNはトークンの2重消費を検証するNotary Node等をR3社がマネージドサービスとして提供するネットワークである。
プライベートなNotary Node等の運用は柔軟性のメリットを享受できる一方でコスト・運用面の負担が大きいため、コスト・運用等の観点からtCNの利用は有力な選択肢となる。そこで本取組みではCorda側ネットワークはtCNで運営されていると仮定し、tCN利用に伴う以下制約事項を前提とする。

- ✓ トランザクションサイズ制限 10MB
- ✓ Notaryの時点バックアップによるリストア不可

■ Corda5について

- ✓ 本書ではCorda4の利用を前提にプラクティスを記載。
- ✓ 2023年には非機能要件をさらに高めた次世代Corda(Corda5)がリリース予定。
- ✓ 本書で記載しているブロックチェーンインタオペラビリティは次世代Cordaでも実現可能であり、非機能要件の高度化に寄与する想定。

次世代Corda(Corda5)アップデート内容例

✓オーケストレーションツールを用いたワーカークラスタを構築し、より堅牢で変化に柔軟なシステムを実現可能に

✓マルチテナンシーやDappsインストーラーの導入により、今まで以上に簡単にビジネスネットワークを構築可能に

✓PKIやコンセンサス・アルゴリズムがカスタマイズできるようになり、よりビジネスニーズに適したアプリケーション設計および運用が可能に

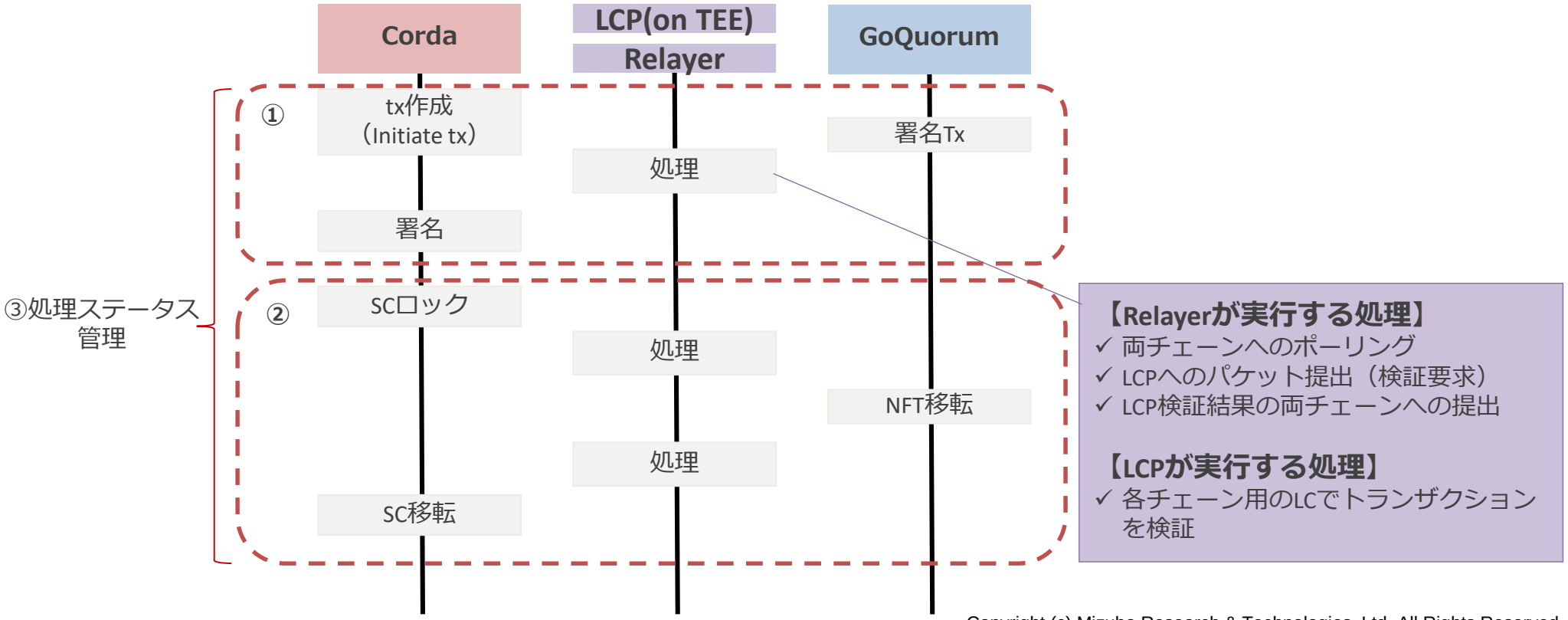
【機能要件】両ブロックチェーンでのアトミックな更新（想定ユースケース①）

ポイント

- ✓ 想定ユースケース①におけるCorda上で発行・管理されているステーブルコイン(SC)とGoQuorum上で発行・管理されているERC721準拠のトークン（NFT）とのDVP決済のような両ブロックチェーンでのアトミックな更新には、Cross Frameworkが具備する機能を活用することで効率よく開発が可能。

Cross Frameworkがサポートする処理

① エンドユーザーによる認証・署名管理
② Simple Commitフローによるアトミックなトランザクションの管理
③ 処理フロー全体の処理ステータス管理



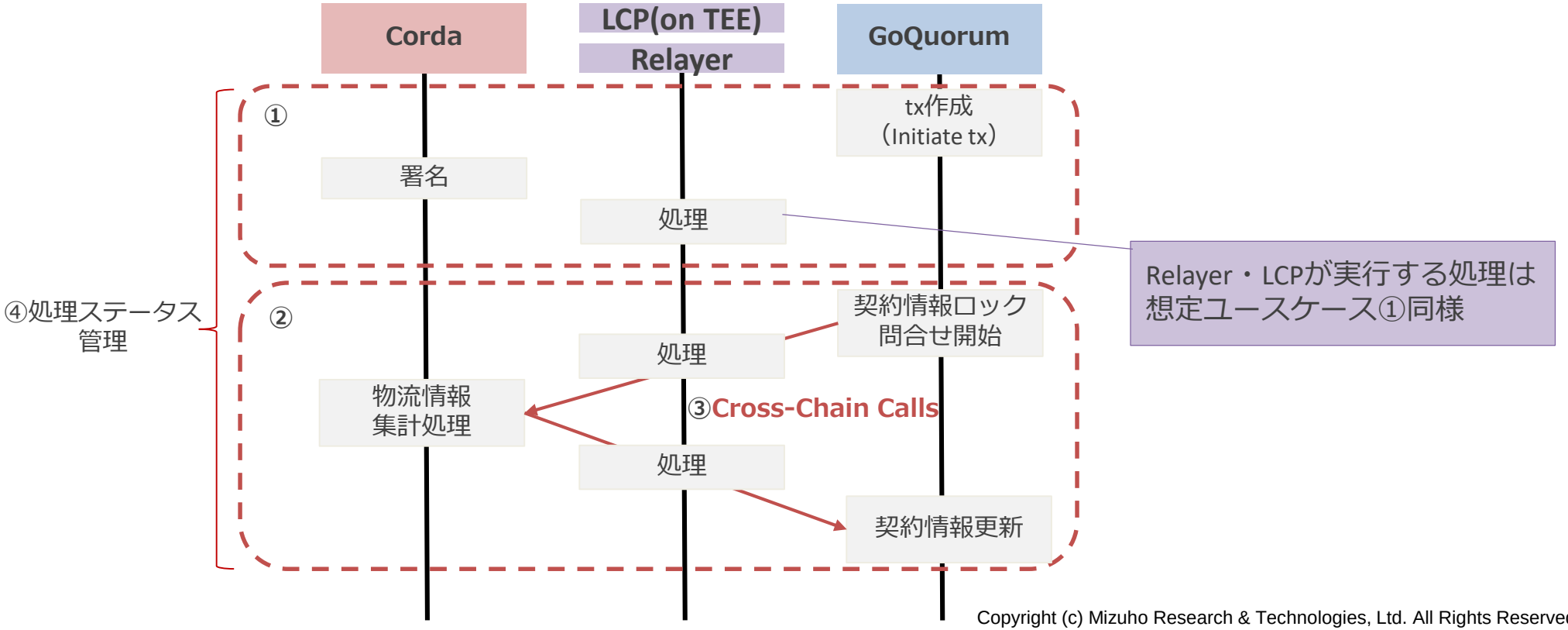
【機能要件】他方ブロックチェーンへの参照（集計）を伴う更新（想定ユースケース②）

ポイント

✓ 想定ユースケース②におけるCorda上で管理されている物流情報を参照してGoQuorum上で管理されている契約情報を日次で更新するような他方チェーンへの参照（集計）を伴う更新処理においても、Cross Frameworkが具備する機能を活用することで効率よく開発が可能。

Cross Frameworkがサポートする処理

① エンドユーザーによる認証・署名管理
② Simple Commitフローによるアトミックなトランザクションの管理
③ Cross-Chain Callsによる他チェーンへの参照処理
④ 処理フロー全体の処理ステータス管理



【機能要件】プライバシー①

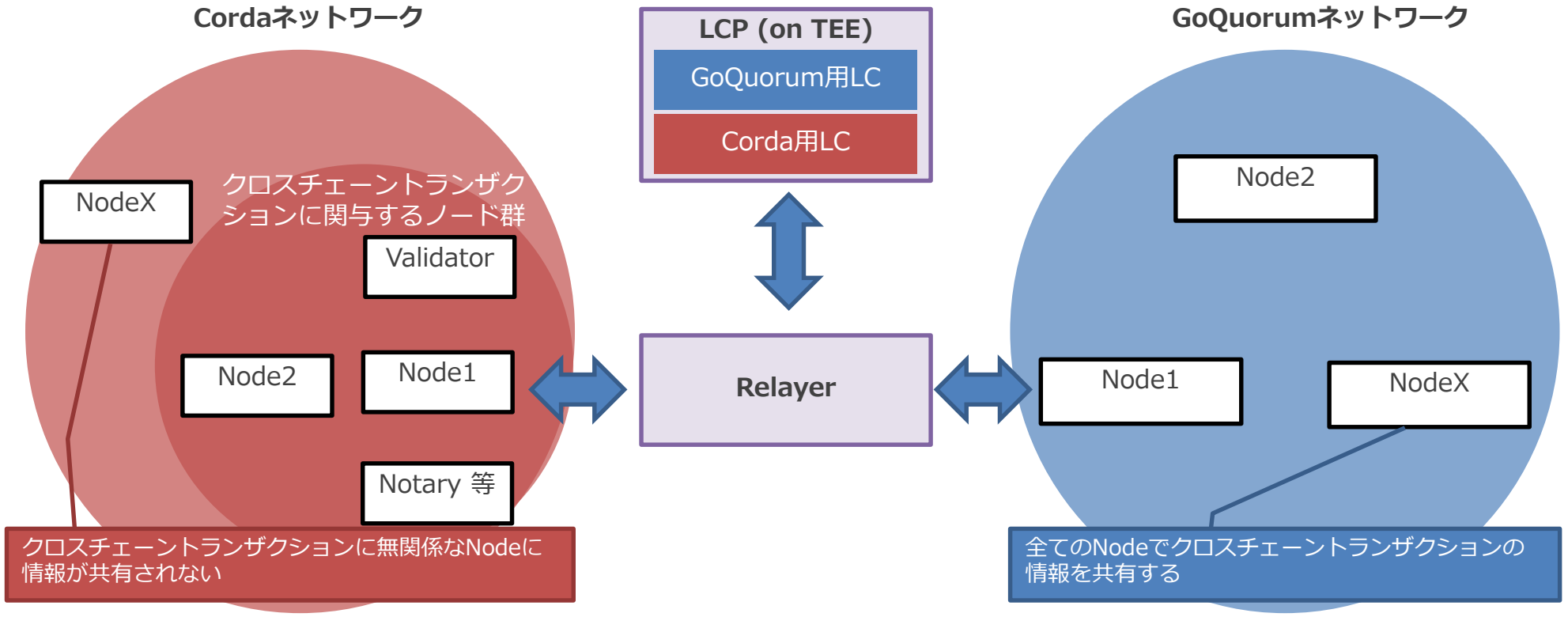
ポイント

✓ 情報の取扱い方に関するチェーン間の差異を把握し、以下観点を踏まえてプライバシー要件を設計する。
① 伝播した情報に関する他方チェーン内での共有範囲
② 他方チェーンへ伝播させる情報

伝播した情報に関する他方チェーン内での共有範囲

■ ブロックチェーン基盤毎に情報の共有範囲が異なる。
✓ Corda: トランザクションに関与するノードのみで情報を共有
✓ GoQuorum: プライベートトランザクションを除き全ノードで情報を共有

■ 情報共有範囲を踏まえ、他方チェーンへ伝播させる情報を設計する必要がある。



他方チェーンへ伝播させる情報

■ IBCプロトコルにおいて他方チェーンに伝播する情報は以下の2種類である。

① IBCの通信に係る制御用データ（非業務データ)

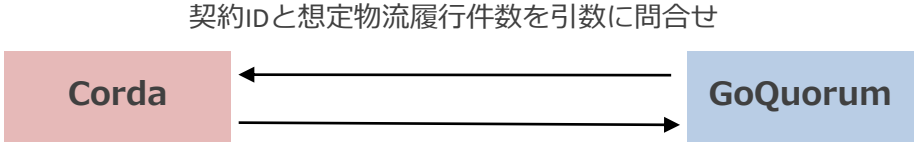
② クロスチェーントランザクションのデータ
（たとえ同一ブロックに格納されるデータであっても、クロスチェーントランザクションと関係のないトランザクションデータは伝播しない）

■ 上記②に関して、Cross-Chain Calls実行時の引数・レスポンス結果は任意の形態をとれる。
→レスポンスに含まれる回答の信頼性、プライバシー要件のバランスを踏まえた設計が可能。

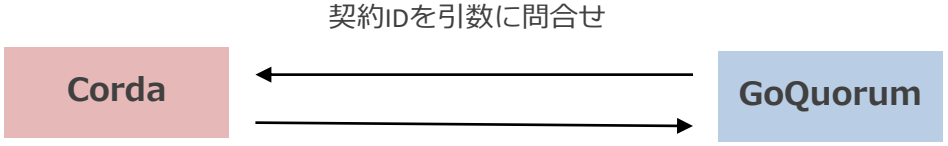
※信頼性、プライバシー要件以外にもパフォーマンス等は考慮が必要

想定ユースケース②における設計例

設計例①：伝播する情報を限定しプライバシーを高めるパターン



設計例②：伝播する情報を増やし回答の信頼性を高めるパターン



【機能要件】アイデンティティ指定

ポイント

- ✓ エンドユーザー等に紐づくアイデンティティ体系に関するチェーン間の差異を把握し、以下観点を踏まえて認証等の機能要件を設計する。
  - ① エンドユーザー等に紐づくアイデンティティ指定方法
  - ② クロスチェーントランザクションにおける認証のための署名主体

エンドユーザー等に紐づくアイデンティティ指定方法

- ■ ブロックチェーン基盤毎にアイデンティティ体系・指定方法が異なる。
  - ✓ Corda:ノードを識別するLegal Identity(LI)とノード内のUUIDの組合せでエンドユーザーをCordaネットワーク内で一意に識別。
  - ✓ GoQuorum: エンドユーザー毎に作成する秘密鍵から導出されるアドレスでエンドユーザーをGoQuorumネットワーク内で一意に識別。
- ■ クロスチェーントランザクション作成時に上記のアイデンティティを指定する。

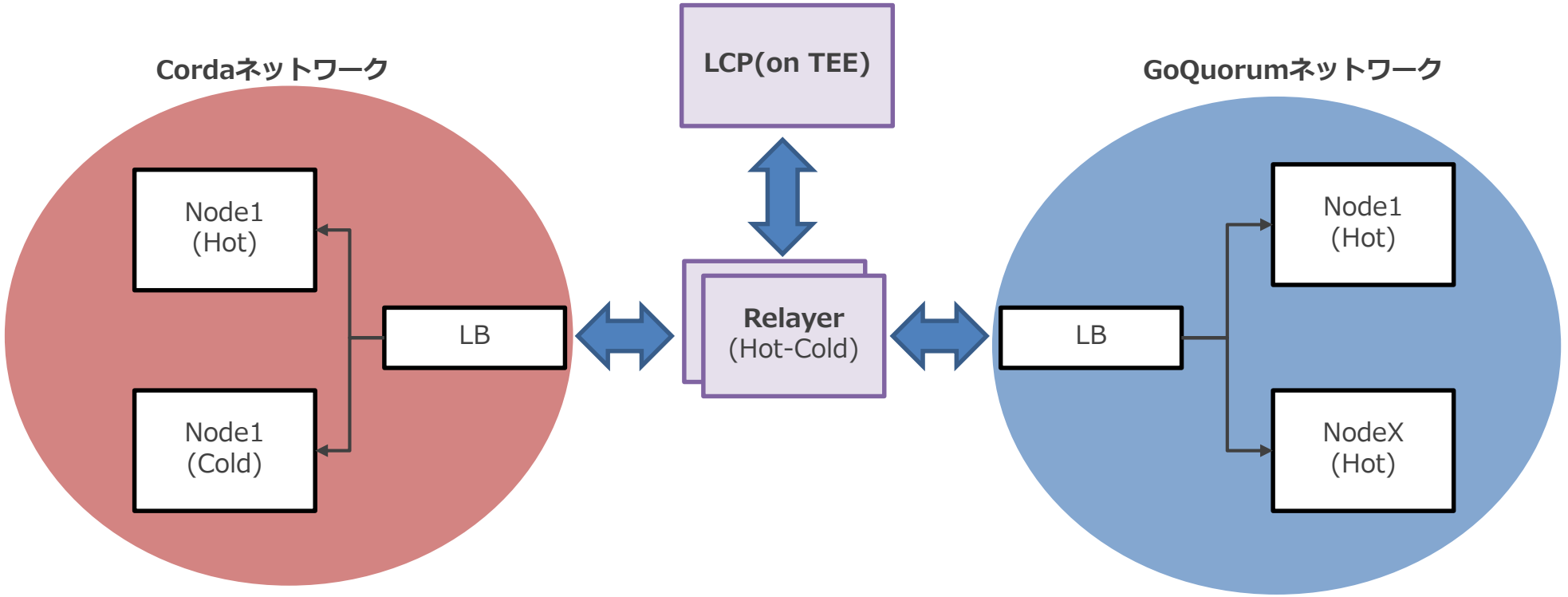
クロスチェーントランザクションにおける認証のための署名主体

- ■ 上記アイデンティティ体系の違いから、クロスチェーントランザクション実行時のエンドユーザーの認証を目的とした署名処理では、以下のような署名主体が考えられる。
  - ✓ Corda:エンドユーザーに紐づくノードが署名を行う。エンドユーザーとノードの紐付けは、R3社提供の拡張ライブラリであるAccounts Libraryの使用が候補として挙げられる。（ノード内で個別エンドユーザーに紐づいた鍵が署名に使用される。）
  - ✓ GoQuorum: エンドユーザーが自身の鍵で署名を行う。

【非機能要件】可用性（耐障害性）①

ポイント

✓ 各構成要素の特徴を踏まえ、ホットスタンバイ/コールドスタンバイの冗長化設計を行う。



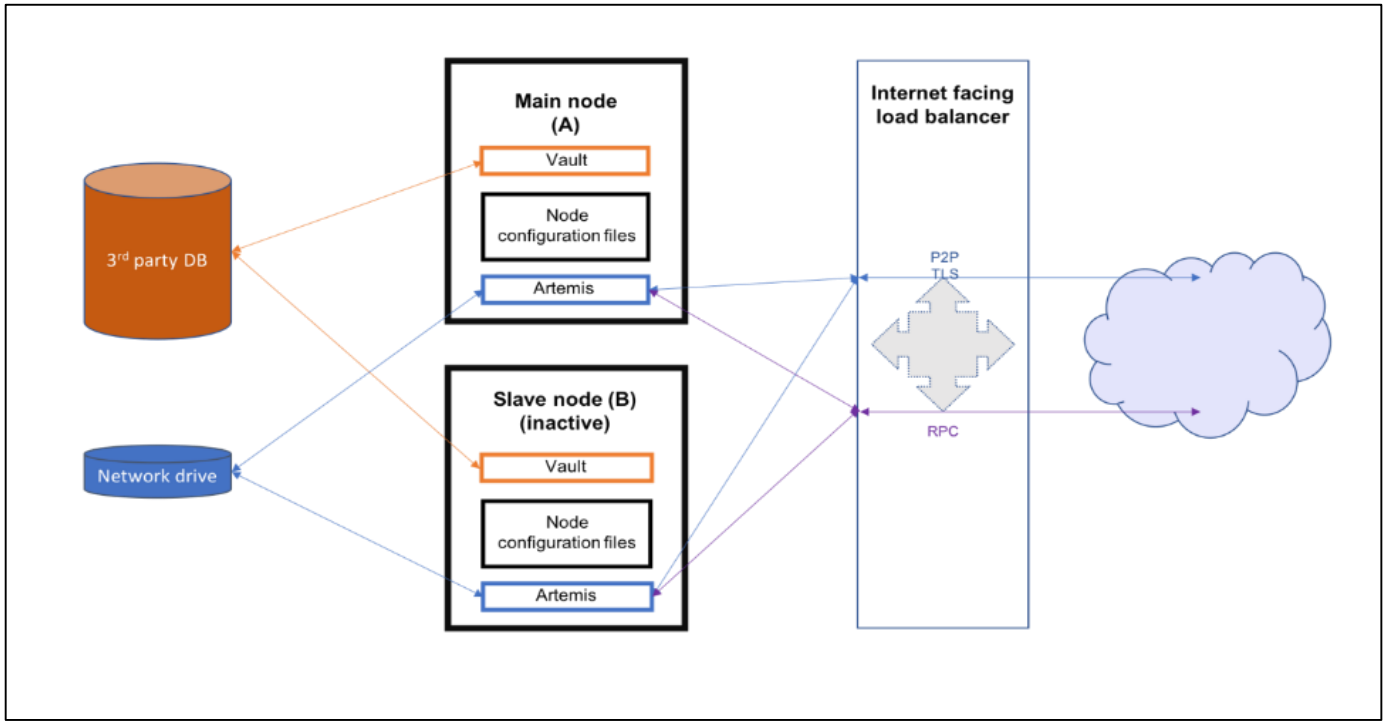
GoQuorum

- IBCに関するデータを含め全ノードでデータを同期しているため、ノード前段にロードバランサを配置しホットスタンバイさせる。
- ノード障害発生時は切替先ノードでクロスチェーントランザクションを続行する。

【非機能要件】可用性（耐障害性）②

Corda

- コールドスタンバイによる高可用性構成を設計する。
- 上記構成ではノード内の処理におけるチェックポイントがDBに書き込まれる。また、メッセージキュー（artemis）でネットワークドライブにマウントし、マシン間でキューの共有が可能となる。
- ノード障害発生時は、コールドスタンバイノードがチェックポイントとキューを引継ぎクロスチェーントランザクションを続行する。



Relayer

- Relayerは全ての通信のリクエスト送信元となるため、リクエストの重複が起きないようにコールドスタンバイによる冗長化が考えられる。

LCP

- 冗長化構成に向けたアップデート中であるため、今後のアップデートを注視しつつワークアラウンドを検討する。

【非機能要件】可用性（回復性）

ポイント

- ✓ 両チェーンの状態を特定時点のバックアップに同時に戻すことは現実的に困難である可能性があり、その場合はデータ毀損発生時にロールフォワードによる復旧は困難となる。
- ✓ 上記を踏まえ、各チェーンで業務データの復旧を行い、IBCの通信に係る制御用データ（非業務データ)については再構築する対応を行う。

GoQuorum

- ■ IBCに関するデータを含め全ノードでデータを同期しているため、基本的にはノード間同期により業務データや IBCの通信に係る制御用データの復旧を実施する。
- ■ ネットワーク全体でデータ毀損が発生した場合は、上記の通りクロスチェーントランザクションの再実行が困難であるため、特権等を用いた運用オペレーションによりスマートコントラクトの状態を更新して業務データを復旧し、業務データ復旧後にブロックチェーン間のChannelを再構築する。

Corda

- ■ 基本的にはノードが使用するDB等のバックアップから業務データや IBCの通信に係る制御用データの復旧を実施する。
- ■ 大規模災害等によりDB等が復旧不可能な場合は、 Collaborative Recovery機能を用いて他ノードのDBと同期し業務データを復旧する。 （Collaborative Recovery機能はトランザクションが共有されているノード間でのみ実行可能。そのため、Collaborative Recoveryの活用には設計段階からの考慮が必要。）
  業務データ復旧後にブロックチェーン間のChannelを再構築する。

Relayer・LCP

- ■ Relayerはステートレスであるためデータ復旧の考慮は不要である。
- ■ LCPはIBCの通信に係る制御用データ（非業務データ)のみを保持するため、ブロックチェーン間のChannelを再構築することよって復旧を行う。また、LCP障害によりホストが変更される場合は、各チェーンが保有するEnclave公開鍵の再登録を実施する。（鍵保管についてはセキュリティ要件の頁を参照）

【非機能要件】性能①

ポイント

- ✓ 各想定ユースケースにおけるラウンドトリップタイムは以下式となる。（ネットワークの伝送時間含まず）

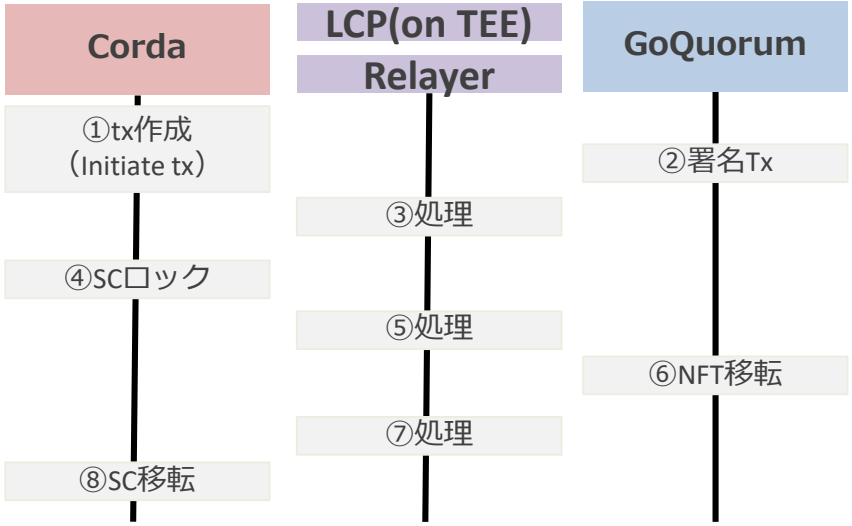
  ユースケース①：(Cordaトランザクション実行時間×3回) + (GoQuorumトランザクション実行時間×2回) + (Relayer・LCP処理時間×3回)
  ユースケース②：(Cordaトランザクション実行時間×2回) + (GoQuorumトランザクション実行時間×3回) + (Relayer・LCP処理時間×3回)

- ✓ 上記式の通り各構成要素の処理時間がラウンドトリップタイムに影響を与えるため、各構成要素における処理性能向上の工夫が有効である。また、中長期稼働において各構成要素の性能劣化を低減させることも有効である。
- ✓ 想定されるラウンドトリップタイムを踏まえ、エンドユーザーへのトランザクション完了通知を非同期で行う等のUX向上の工夫が考えられる。

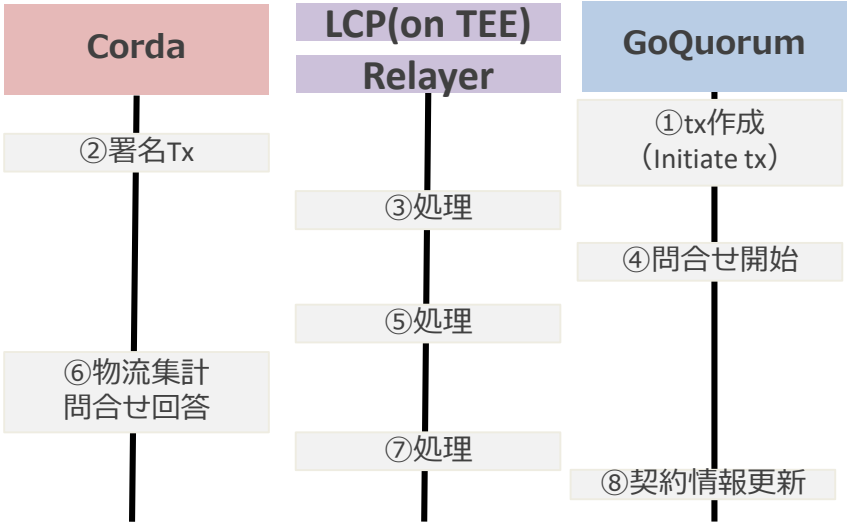
各ユースケースにおけるランドトリップタイム

■ 各ユースケースの処理シーケンスに基づき、ラウンドトリップは以下式であらわせる。

ユースケース①ラウンドトリップタイム
Cordaトランザクション実行時間(①,④,⑧)
＋ GoQuorumトランザクション実行時間(②,⑥)
＋ Relayer・LCP処理時間(③,⑤,⑦)



ユースケース②ラウンドトリップタイム
Cordaトランザクション実行時間(②,⑥)
＋ GoQuorumトランザクション実行時間(①,④,⑧)
＋ Relayer・LCP処理時間(③,⑤,⑦)

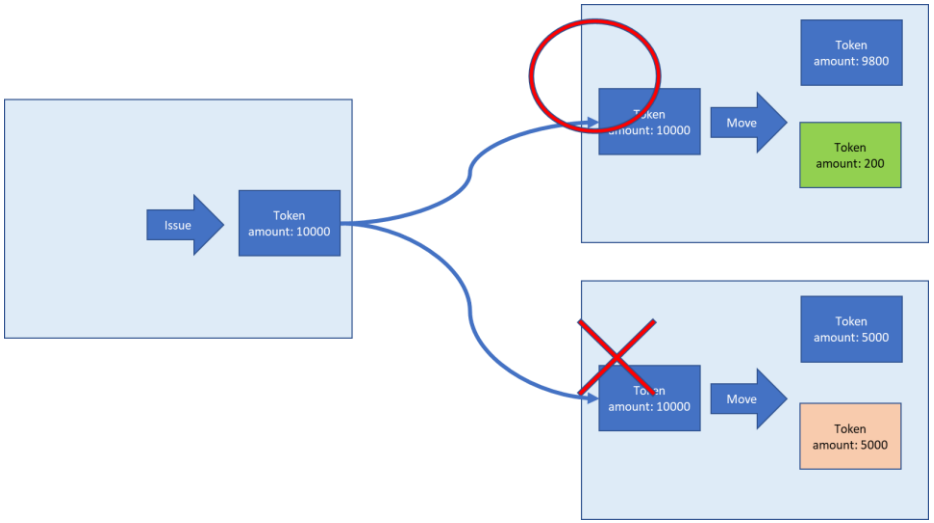


処理性能向上の工夫

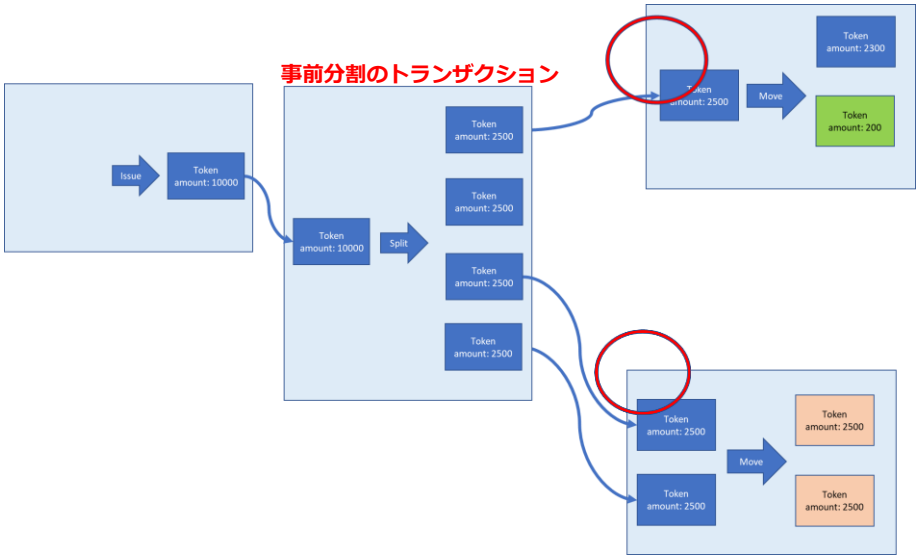
- 各構成要素の処理時間がラウンドトリップタイムに影響を与える。また、Relayerが各チェーンへのポーリングに要する処理時間はRelayer・LCPの処理時間全体の大部分を占めるため、各チェーンの処理性能の向上がラウンドトリップの短縮に有効である。
- 各チェーンの処理性能向上はクロスチェーントランザクション固有の要素ではないが、工夫の一例として本書ではCordaにおける対応例を3点記載する。
  - ✓ マルチスレッド処理による複数トランザクション発生時の処理待ち低減
  - ✓ トランザクションに含まれるStateを事前分割することによる並列処理容易性の向上。（下図イメージ）
  - ✓ ノードに含まれるRDBのインデックスメンテナンスや、 Corda Enterpriseの機能（Archive service） を用いたトランザクションのライフサイクル上不要になったRDBレコードの削除による中長期の性能劣化低減。

State事前分割イメージ

事前分割を実施しない場合
CordaはUTXOモデルを採用しており、同タイミングで同じStateを使うことができない。



事前分割を実施する場合
Stateを事前に分割しておくことで並列でStateを使用できる。



ポイント

- ✓ 例外・異常が発生する可能性のある箇所について網羅的に例外・異常を検知可能とする。
- ✓ 例外・異常が発生するタイミングにより必要な対応が異なるため、機能または運用による対応を定める。

例外・異常が発生する可能性のある箇所、検知方法について

■ 例外・異常が発生する可能性のある箇所及び検知方法は以下の通り

構成要素	例外・異常の検知方法
GoQuorum	① 各ブロックチェーン内で例外を捕捉 ② 各ブロックチェーン内で例外を捕捉できない場合等は、Relayerがタイムアウトにより異常を検知。 (Cross-Chain Calls実行時等に想定される処理時間に応じたタイムアウト条件を設定可能)
Corda	
Relayer	ログ監視により検知 （両チェーンのいずれかの主体による監視が必要）
LCP	

例外・異常検知時の対応方法について

■ 例外・異常が発生するタイミングにより必要な対応が異なるため、以下例のような機能や運用による対応案が考えられる。

例外・異常の発生タイミング例	対応案
GoQuorumでトランザクション完了後に他構成要素で例外発生 （例：想定ユースケース①におけるNFT移転完了後）	特権を用いた運用オペレーションによる逆取引対応など
Cordaでロック処理後に他構成要素で例外発生 （例：想定ユースケース①におけるSCロック完了後）	ロック解除処理の機能を実装し機能による自動対応をするなど

【非機能要件】運用保守性(リリース運用)

ポイント

- ✓ アプリ・インフラリリースにおいては、各構成要素の更新による影響範囲を把握して運用設計を行う必要がある。
- ✓ アプリリリースにおいては、以下構成要素更新時のための運用設計が必要。
  - ① IBC Module、Cross Framework、ユースケース用スマートコントラクト等の更新
  - ② LCPプログラムの更新
- ✓ インフラリリースにおいては、以下構成要素更新時のための運用設計が必要。
  - ① TEEに構築したLCPにおけるホストの変更を伴うインフラ更新

IBC Module、Cross Framework、ユースケース用スマートコントラクト等の更新について

- ■ 各プログラムの関係は、Cross Frameworkがユースケース用スマートコントラクト等を呼び出す関係となっているため、IBC ModuleやCross Frameworkの更新はユースケース用スマートコントラクト等に影響を及ぼさない。（いわゆる業務ロジックに影響を与えない）
- ■ IBC Module更新の際はデータ移行は不要だが、ブロックチェーン間のChannelを開局する運用を実施する。
- ■ ユースケース用スマートコントラクト等の更新については、インターフェースレベルで破壊的な変更がありCross Frameworkからの呼び出しに影響があるケースのみCross Frameworkの修正が必要。

LCPプログラムの更新について

- ■ LCP内のLight Clientを更新する場合は更新後にブロックチェーン間のChannelを開局する運用を実施する。

TEEに構築したLCPにおけるホストの変更を伴うインフラ更新

- ■ プログラム・設定・データをホストマシン間で引き継ぐことが可能だが、 TEEのEnclaveが用いる秘密鍵の再登録を実行する必要があるため、鍵更新の運用を実施する。（鍵保管についてはセキュリティ要件の頁を参照）

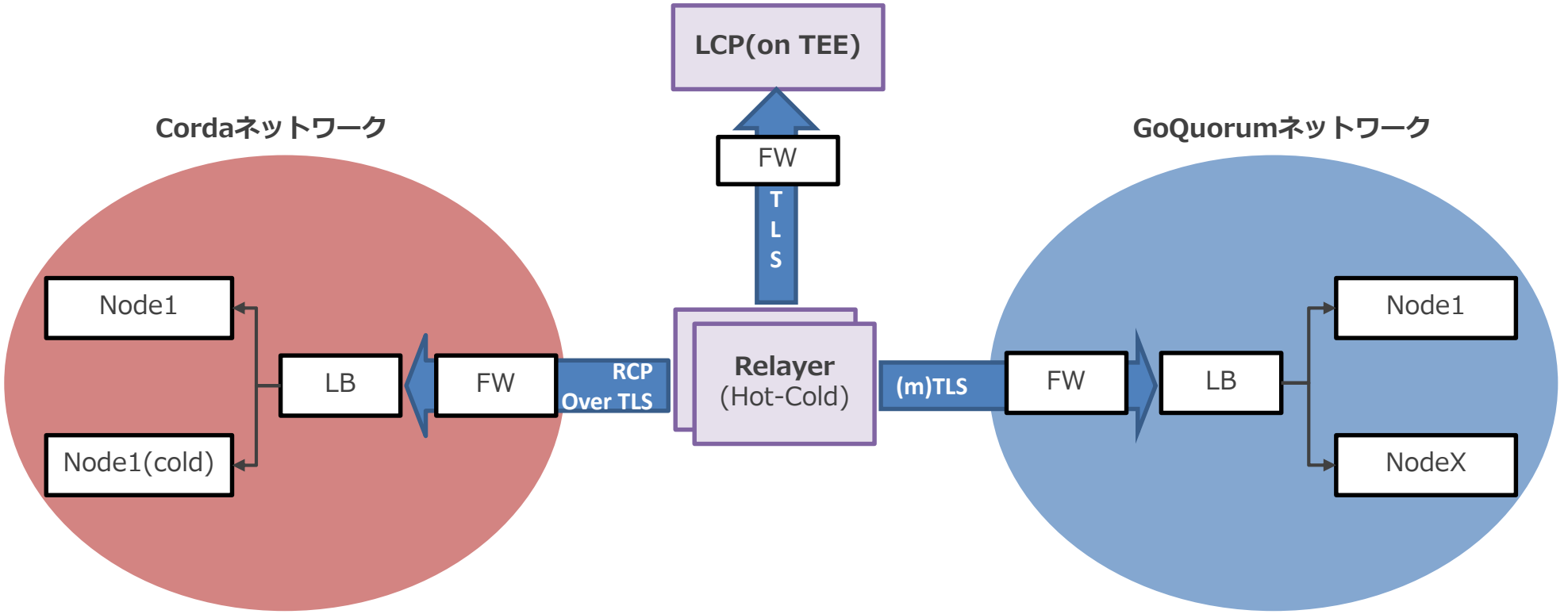
【非機能要件】セキュリティ ①

ポイント

- ✓ クロスチェーントランザクションに係るアプリケーションレイヤーのセキュリティはIBCプロトコルによって担保（他方チェーンへの連携時にトランザクションを改竄するようなケースなど）
- ✓ インフラレイヤーのセキュリティは原則的に各構成要素で実施
- ✓ インターオペラビリティ導入による追加要素としてネットワークと鍵管理の考慮が必要

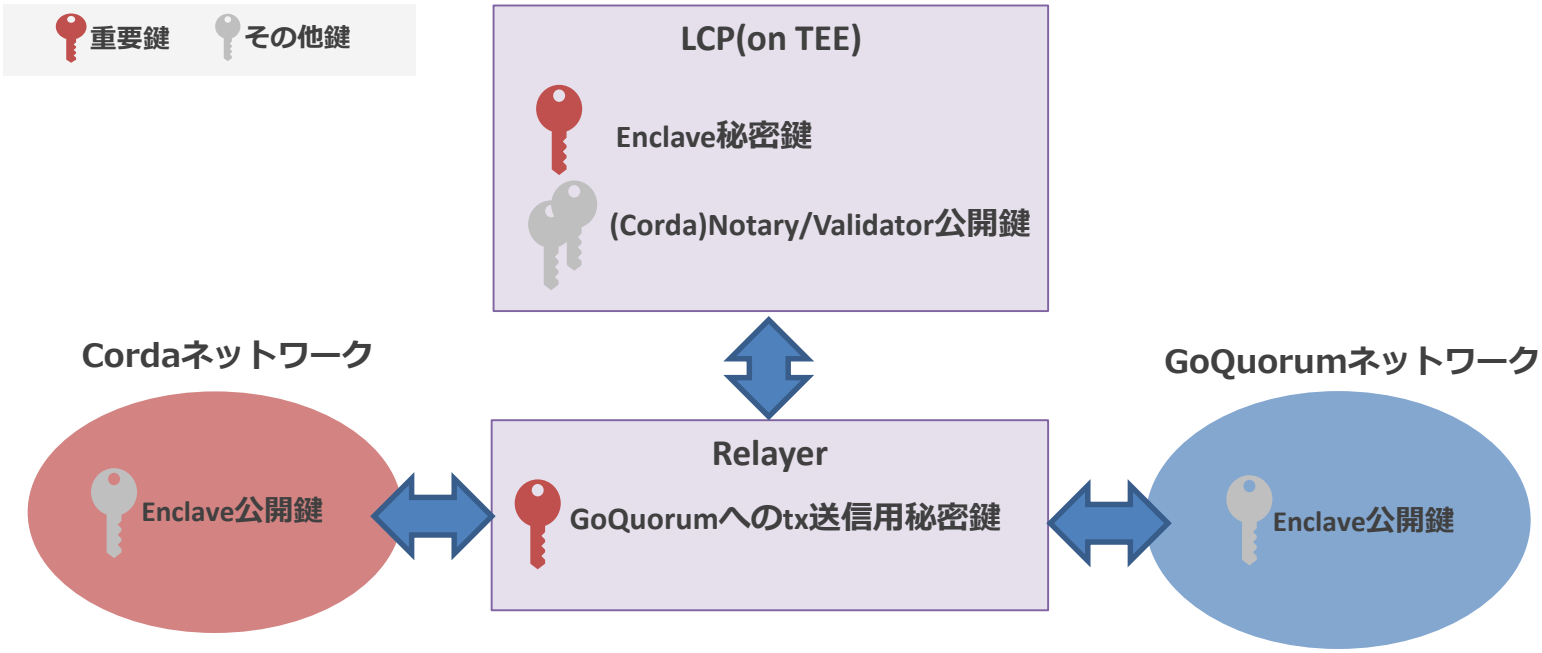
ネットワークの制御について

- ■ 本書が想定するアーキテクチャでは各構成要素間に新たな通信が発生する。
- ■ Relayerがリクエスト送信元となり全ての通信を実施。
- ■ 上記通信において、証明書による認証やファイアウォールによる通信制御、通信経路の暗号化を行う。



鍵管理について

- インターオペラビリティ導入により厳格な保管が新たに必要となる秘密鍵等は以下2つであり、リスク、コスト、性能等を踏まえ管理方式を設計する必要がある。
  - ① TEEのEnclaveが用いる鍵ペア
  - ② RelayerがGoQuorumにトランザクションを送信するための秘密鍵
- ①については、本書が想定するアーキテクチャではLight Client ProxyをTEE上に構築するため、TEEのEnclaveが用いる鍵ペアの管理が必要であり、秘密鍵はTEEの仕組みのもとセキュアに保管される。
- ②については、漏洩した場合に一時的なチェーン間相互連携の停滞等が起きる可能性はあるものの、不正なトランザクションを実行しようとしてもIBCプロトコルにより各チェーンで不正が検知されるため、漏洩した際のリスクが比較的に低い鍵である。
- その他以下図に示すような各種公開鍵は厳格な保管が求められる鍵ではないものの、鍵のローテーションが発生するケースについては一部未サポートであり、ワークアラウンドを検討する必要がある。



※GoQuorum・Cordaの各基盤では、上記以外にも各基盤固有の鍵が管理される

- 本書ではHyperledger YUI及びLCP on TEEを採用したGoQuorum・Corda間のインターオペラビリティ実現方式における機能・非機能要件実現のためのプラクティスを記載。

- Hyperledger YUI等のインターオペラビリティ技術や各ブロックチェーン技術は技術進展が続いているため、インターオペラビリティを導入する際には導入時点の技術水準を踏まえた機能・運用の設計を行いつつ、各技術のアップデートを適宜取り込むことが肝要であり、それを実施できる体制も重要である。

- 異なるブロックチェーン間のインターオペラビリティ実現はビジネス拡大やユースケース進展において必須だと考えられ、ブロックチェーンを活用するエンタープライズ領域のプロジェクトにおいても重要な検討テーマであるといえる。本書ではエンタープライズ領域のブロックチェーン活用を念頭に許可型ブロックチェーンのGoQuorum・Corda間のインターオペラビリティを題材としているが、エンタープライズ領域においてもパブリックチェーンの利用が広がっており、パブリックチェーンも視野に入れたインターオペラビリティの実現についても今後ますます重要になると考えられる。

本資料は情報提供のみを目的として作成されたものであり、商品の勧誘を目的としたものではありません。

本資料は、当社が信頼できると判断した各種データに基づき作成されておりますが、その正確性、確実性を保証するものではありません。また、本資料に記載された内容は予告なしに変更されることもあります。

本資料記載の製品、サービス名は各社の商標または登録商標です。