A study on universal standard for cross-ledger intercommunication

Classification, analysis and comparison of cross-chain projects

YUAN FAN

Master Thesis in Communication System
Supervisor(s): Henrik Gradin, György Dán
Examiner: Viktoria Fodor
School of Information and Communication Technology
KTH Royal Institute of Technology
Host Company: Centiglobe
August 23, 2019
Stockholm, Sweden 2019

Abstract

Write your abstract here...
Keywords: Keyword1, keyword2, ...

Sammanfattning

Write your Swedish summary (popular description) here...
Keywords: Keyword1, keyword2, ...

Acknowledgements

Write your professional acknowledgements here...

Acknowledgements are used to thank all persons who have helped in carrying out the research and to the research organizations/institutions and/or companies for funding the research.

Name Surname,
Place, Date

Contents

List of Figures

List of Tables

List of Acronyms

ICT	Information Communications Technology
BTC	Bitcoin
ETH	Ethereum coin
HTLA	Hash Time-locked Agreement
HTLC	Hash Time-locked Contract
MPC	Multi-Party Computation
SPV	Simplified Payment Verification
RSMC	Recoverable Sequence Maturity Contract
addr	address
Tx	Transmit(Tx) Data
ILP	Interledger Protocol
DApps	Decentralized Applications
ELA	Elastos coin
HHCM	Hierarchical Hybrid Consensus Mechanism
PoW	Proof-of-Work
PoS	Proof-of-Stake
BFT	Byzantine Fault Tolerance
DAG	Directed Acyclic Graph
DPoS	Delegated Proof-of-Stake
PoI	Proof-of-Intelligence
API	Application Programming Interface
IBC	Inter-Blockchain Communication protocol
UDP	User Datagram Protocol
TCP	Transmission Control Protocol
PBFT	Practical Byzantine Fault Tolerance
LFT	Loop Fault Tolerance
AUXPoW+DPoS	Auxiliary Proof of Work and Delegated Proof of Stake

Listings

Chapter 1

Introduction

1.1 Motivation

Originally, the ledger is the foundation of accounting. However, it is not difficult to find that there are many shortcomings in the long-term use of traditional ledgers. For example, low efficiency, high cost, opacity and easy to cause fraud and abuse issues.

With the development of information technology, these ledgers have gradually evolved into digital technology. Distributed ledger is a major leap after the digitization of ledger-based technology. A distributed ledger is a database that is shared, replicated, and synchronized between network members [**?**]. It records transactions between network participants, such as the exchange of assets or data. From a technical point of view, the Distributed Ledger Technology (DLT) not only inherits the traditional bookkeeping philosophy but also has its unique innovations, which have some advantages that traditional ledgers cannot reach.

Based on the background above, there exist many distributed ledgers and for them to be fully distributed they need to communicate with each other and also traditional ledgers. Otherwise, the consistency between the ledgers of different entities across the chain would not be guaranteed. There have been many different approaches such as cross-chain protocols and platforms released to realize the cross-chain transactions, so it is worthy to find out the differences between them.

1.2 Research Objectives/Research Questions, Research Methodology

There is one technical issue that affects the blockchain developers, that is the communication. A single blockchain network is a relatively closed system that does not actively interact with the outside world. The assets of each chain are also an inde-

pendent value system. If we can break through the interoperability among different ledgers and let the value circulate in the wider world, it will inevitably promote the rapid development of the blockchain industry. Cross-chain technology is dedicated to building a bridge of trust between ledgers, breaking the situation of an isolated value system, and realizing asset interoperability in order to achieve a true win-win situation.

This project makes uses of qualitative research strategy, where the research approach always implemented as interpretivism. To gain a standard requirement/need for cross-ledger communication model, the variety of factors are governed by all valuable findings through the research, which is not easily quantifiable (measurable). Therefore qualitative approach was found to be most applicable for this study.

For the purpose of this research, I decided to use literature review first to gain a deep understanding of cross-ledger history and realization of communications through published papers. Then the key point is the case study among popular projects. By studying the different cross-chain implementations, exacting the main idea of them, we can summarize and categorize them into different groups. Hence, identify patterns and commonalities in the cross-chain field. Even help more and more developers to consummate the blockchain design.

1.3 Research Contributions

My research has contributed to the universal demanding and requirements on cross-ledger communication towards blockchain areas. In particular, I have focused on the problems that the realization of cross-chain communication is facing.

For a long time since Bitcoin open the blockchain era, the blockchain world is like the single-machine time back in the 1960s. Every blockchain is highly independent and difficult to communicate with each other. Hence, the data and services in the blockchain world are confined to the individual blockchain, as a result, this phenomenon will discourage development. As if we could find a standardized cross-chain protocol/platform that could link all blockchain systems, and the services could be more specific and complete due to the co-operate of blockchains. The popularization and mature of cross-chain technology will lead a revolutionary development in the blockchain field. And different from the Internet to achieve the circulation of information, cross-ledger could realize the circulation of value.

To understand different patterns or implementations of cross-chain technology, we need to start with the history of cross-chain and grasp the main idea of chain interoperability based on literature review. Based on those findings, we could summarize the key problems the blockchains now facing, according to those difficulties I will

give several examples through the case study in the following paragraph. This is where my research contributes significantly to the existing literature. The comparison from various aspects will next lead to a standard and universal needs of the cross-chain area. In my thesis, I have considered the following three major aspects of this study as shown in Figure **??**

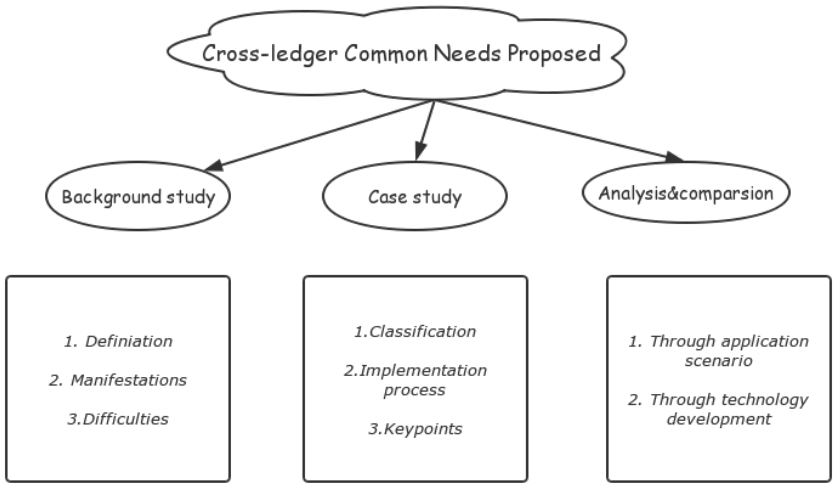

Figure 1.1: Research components of cross-chain communication

1.4 Thesis Organization

This thesis is organized into five chapters as follows:

• Chapter **??** concentrates on the research value and market meaning of this thesis, then briefly introduces the concept of cross-chain communication.

• Chapter **??** studies the background of the cross-chain project by classifying different manifestations of cross-chain communication as well as pointing out the problems cross-ledger communication facing.

• Chapter **??** focuses on the theoretical solutions that will address the difficulties, discusses the communication process of various cross-chain projects based on different group rules.

• Chapter **??** put efforts on analyze the market demand situation and discuss the applications that could be adopted, compare the technology development of some representative projects.

• Chapter **??** summarizes the findings during the research and suggests several ideas for related future work.

• Appendix **??** contains a comparison table that evaluating 20 cross-chain projects from several valuable aspects.

• **??** lists implementation code pieces with essential functions explained and the utilization of smart contracts, for further test use.

Chapter 2

Literature Study

2.1 Background

2.1.1 What is cross-ledger?

Since the development of blockchain technology, many different chains have been born, and the information isolation of many chains inevitably forms the value island effect of blockchain. There is a need for a technology that can work with different blockchains and become the bridge connecting them.
Cross-ledger (or cross-chain) in the narrow sense is the process of asset/data interoperability between two relative independent blockchains.
Traditional ICT field defines the *interoperability* as the ability to share and utilize information between different ICT systems or modules in a reliable way [**?**]. While Vitalik Buterin mentioned in Chain Interoperability [**?**] that interoperability in the blockchain area mainly describes the ability of assets exchange, information intercommunication and atomic transactions between various blockchains. This can be achieved by introducing the third party without modifying the original chains.

2.1.2 Evolution of Cross-chain

As shown in Figure **??**, it has not been a short period for Ripple to release the *Interledger Protocol*(ILP) [**?**] in 2012 since the advent of Bitcoin Network in 2009. Interledger protocol proposed a cross-ledger interoperability scheme for the first time in the blockchain area, which enables cross-ledger transfers through third-party notaries. In 2014, the BlockStream team, founded by the Bitcoin core developer group, first proposed a *Pegged Sidechains* cross-chain interaction scheme, introducing a sidechain with a two-way peg to achieve cross- chain asset transfer. Soon in 2015, the Bitcoin Lightning Network [**?**] adopted a *Hashed Timelock* mechanism to implement a fast transaction channel off the Bitcoin main chain. In 2016, the BTC-Relay [**?**] solution was released, and the one-way cross-chain communication

from Bitcoin to Ethereum was realized based on the relay cross-chain scheme. In the same year, Chain Interoperability [?] by Vitalik Buterin made a comprehensive and in-depth analysis of blockchain interoperability issues. In 2017, Polkadot and Cosmos first proposed a solution for building a cross-chain network infrastructure platform. Now the Cosmos hub blockchain has launched in March 2019.

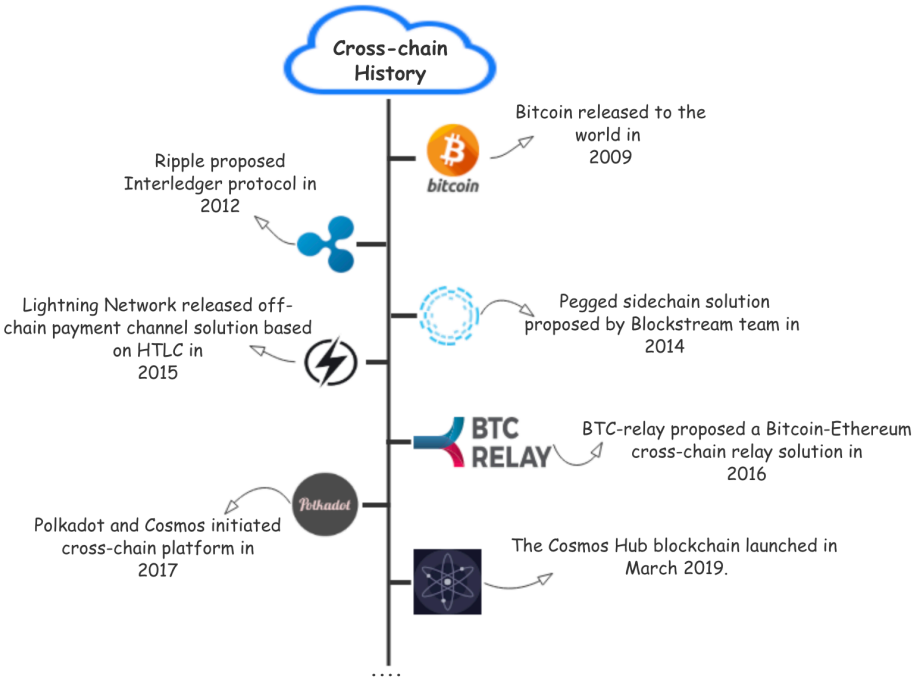

Figure 2.1: Cross-chain development tree

2.2 Cross-chain manifestation

So far we know, current public blockchain is a self-adaptive, relatively closed distributed system. Although the system allows new nodes to join and old nodes exit, it also has its own fault-tolerant mechanism. It is hard to be compatible with external systems.
In a way, there are two main value/data interoperability implementations between chains:

1. **Inter-chain asset exchange**:
   It usually refers to asset swapping between different users on both chains. However, the total amount of assets in each chain does not increase or decrease, instead, the ownership of the assets has changed. The process of this

change needs to occur synchronously on both chains.

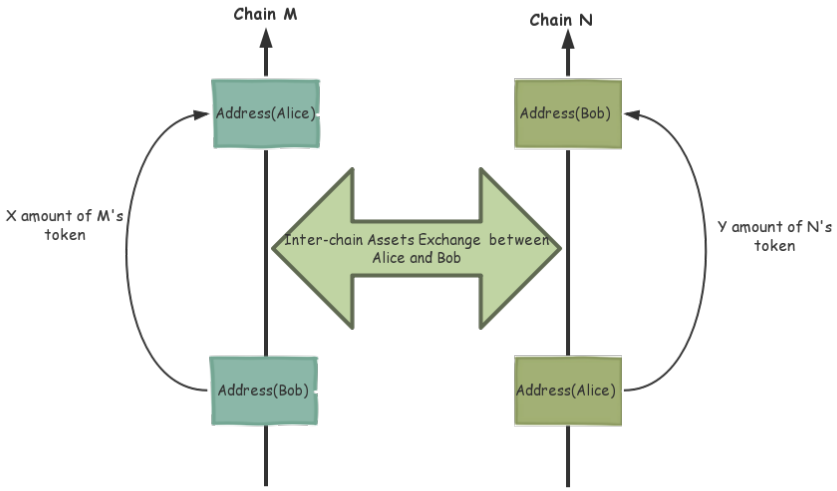

Figure 2.2: Inter-chain exchange diagram

This diagram briefly describes the situation when Alice wants to use X amount of chain M's token exchange for Bob's Y amount of chain N's token. Eventually, Bob's token in chain N swaps to Alice's address in chain N, similarly, Alice's token swaps to Bob's address located in chain M.

2. **Inter-chain asset transfer (one/two-way)**:
Compared with the consistency in the total number of assets in the asset exchange, the transfer has a transfer of asset value, which is manifested in the increase or decrease of the available assets in each chain. For example in Figure **??**, the scenario is Alice wants to transfer X number of chain M's token to chain N, as a result, chain M send X amount of token to a locked address in original chain, in turn, chain N generates an equal amount of tokens of itself in a certain address that can be used. Thus, the transfer of this asset succeed.

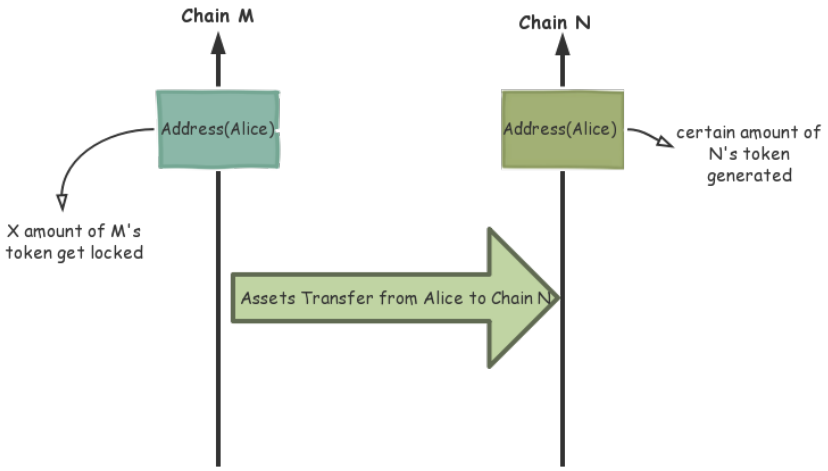

Figure 2.3: Inter-chain transfer diagram

For now, researches and applications of cross-chain are mainly focused on these two methods. Some projects have proposed the concept of cross-chain smart contracts, which is similar to the realization of cross-chain asset transfer in terms of technical implementation.

2.3 Difficulties

The number one feature in the blockchain is immutability, every record must be accurate to protect value. Hence, in the cross-chain project, the key point is to ensure the accuracy of each transaction. There are many obstacles cross-chain need to encounter. To sum up, the following are several key points:

- **How to ensure the atomic of transactions.**
  That is, cross-chain transactions either occur or do not occur. Otherwise the inconsistency and "out-of-sync" status of the two chains will become the biggest system vulnerabilities in cross-chain transactions, and the security of both systems will be threatened. This is the basic requirement for realizing cross-chain transactions, and it is also a difficult point that must be solved in cross-chain transactions.
- **How to complete the confirmation of the transaction for other chains.**

The confirmation including the transaction has occurred and is wound up and written into the right block as well as the transaction has been confirmed by enough blocks in the whole system so the probability of invalidation of the transaction due to system reconfiguration will be very low. Blockchains are lack of a mechanism to actively obtain external information, so it is not an easy task to confirm the trading status of another chain.

- **How to ensure that the total assets of the two chains remain unchanged.**
  In the scenario of asset exchange, the assets of the two chains are not substantially exchanged, so this type of situation does not change the total assets of each chain. However, in the scenario of asset transfer, the number of available assets in each chain changes, total assets can remain unchanged only when the cross-chain transactions are accurately recorded, and the accounting of the two chains is completely atomic, either at the same time or not.

- **How to ensure the independent security of the two chains.**
  When two chains inter-operate, it is inevitable that they will affect each other. How to ensure the security of their own chains and the others in the process of cross-chain transactions is a problem worth considering. If the security issue cannot be isolated, then one attacked chain will affect the entire cross-chain network.

- **How to realize multiple chains interoperability.**
  Take the history of the computer network as a reference, the independent blockchain network will eventually embark on the future of interconnection. How to link these existing and future blockchain networks to be unified into one whole network will be one of the most important issues of the future cross-chain network.

2.4 Summary

This chapter introduces a thorough background study of the cross-chain research area. It briefly reviews the history of cross-chain development and outlines the importance of the growing technology of cross-chain. Two main manifestations of cross-chain transactions are proposed to give a classification standard for Chapter **??** project study. Chapter **??** also describes the key difficulties that cross-chain projects facing. Some of them will be discussed in the following chapter.

Chapter 3

Project Classification

3.1 Solutions

Based on what we discussed about the difficulties in previous section, there are some solutions that came up with multiple cross-chain projects. They will be illustrated in following paragraphs.

3.1.1 Ensure the atomic of transactions

3.1.1.1 Atomic swaps

An atomic cross-chain swap [**?**] is the basic theoretical framework for multiple parties exchange assets across multiple blockchains. Atomic operations in computer science ensures every exchange either success or failure, no third intermediate state. For a more intuitive introduction to the atomic swap protocol, we assume an exchange scenario:

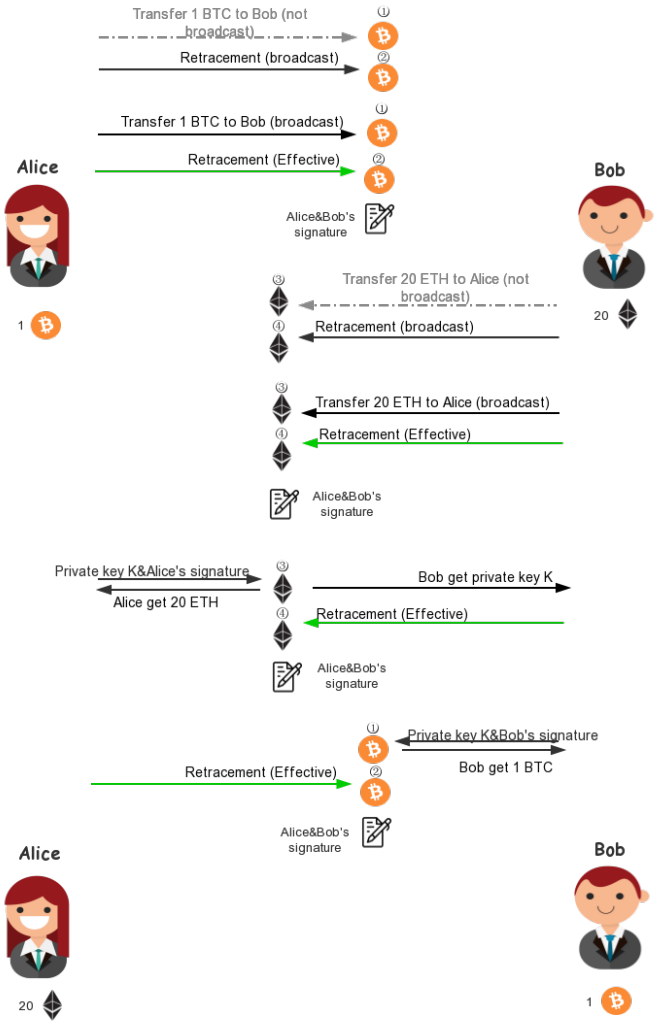

Figure 3.1: Atomic swaps diagram

1. Assume Alice has 1 BTC on chain A while Bob has 20 ETH on chain B, and Alice wants to change Bob's 20 ETH with one BTC. Both Alice and Bob have wallet addresses on both chains A and chain B.

2. In order to initiate this transaction, Alice needs to randomly generates a key K, which is known only to Alice and then initiates a 1 BTC on chain

A transaction (transaction ①) to Bob. The transaction can be only finished when obtains the signature of Bob and provides the key K.

3. Before broadcast transaction ①, Alice will first broadcast a retracement (transaction ②). If transaction ① does not receive the correct key and signature within 48 hours, the amount paid by that will be returned to Alice. Transaction ② must be signed by Alice and Bob after the broadcast to take effect. At the same time, Alice will only broadcast the transaction ① to the network if transaction ② is successfully validated.

4. Bob now sees the transaction ② sent by Alice. If Bob agrees, he will sign the transaction ②, of course, Alice will also complete the signature so that the retracement will take effect. Then Alice will broadcast the transaction ① to the whole network.

5. Bob can only get the value K after he pays Alice with 20 ETH. Hence Bob initiates transaction ③ on chain B to pay Alice 20 ETH. These 20 ETH are only available if Alice enters the decrypted key K and attaches Alice's signature. To prevent Alice from denying, Bob also issues a retracement transaction ④ that requires Alice and Bob to sign together before the broadcast transaction ③, when Alice does not provide the correct key or the signature within 24 hours. then activate the retracement,20 ETH will be returned to Bob.

6. After Alice sees the transaction ④, both Alice and Bob need to attach their signature to this transaction to take effect. At this time, Bob will broadcast transaction ③ to network.

7. In order to get 20 ETH, Alice will sign transaction ③ with the correct value K. For now, transaction ③ succeeds, Alice obtains 20 ETH, and Bob obtains key K.

8. After Bob gets the key K, he goes back to chain A, enters the key K and his signature, and finally gets 1 BTC from Alice.

From the diagram, we could obtain that the atomic swap protocol does not transfer the assets of Chain A to Chain B, but only the assets ownership of both chains. The total assets of Chain A and Chain B have not changed, so it can only achieve asset exchange between chains and cannot achieve asset transfer.
This solution not only can be applied to the decentralized ledger system, but also to the centralized ledgers. As long as the two systems provide the functions of retracement, time lock, and key lock.

3.1.1.2 Hash Time-Locked Contracts(HTLC)

HTLC is a very technical implementation of the atomic swap protocol. It guarantees the atomicity of the transaction through the hash lock and the time lock mechanism.

In different systems, whether it is a blockchain system or a centralized ledger system, despite the ways of implementing the lock, the principle behind it is the same, that is, only certain hash conditions or time met, the transaction is allowed to take effect.

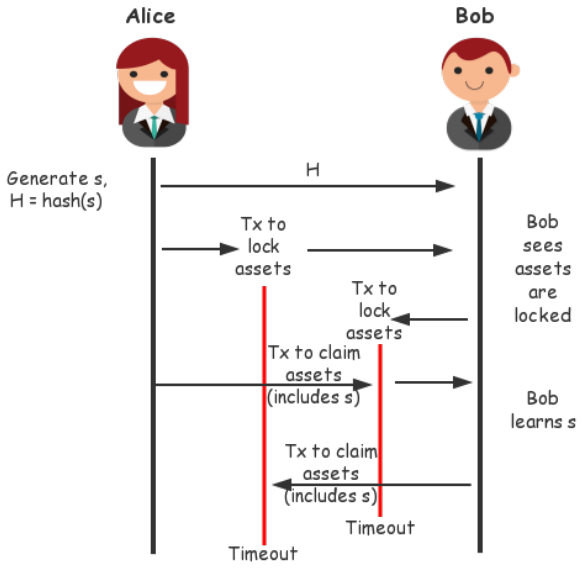

Figure 3.2: Hash Time-lock Contract diagram

Using only hash time locks is not enough when you want to achieve cross-chain asset transfer, you also need to cooperate with other cross-chain technologies to ensure the authenticity of cross-chain transactions.

3.1.1.3 Hash Time-locked Agreements(HTLA)

HTLA is one HTLC generalization protocol came up by Interledger [**?**], regardless whether the system support HTLC or not, whether it's a distributed or centralized ledger. HTLA can be used to implement cross-chain exchange between system or even support multi-hop cross-chain interchange between multiple systems, as shown in the Figure **??** below.

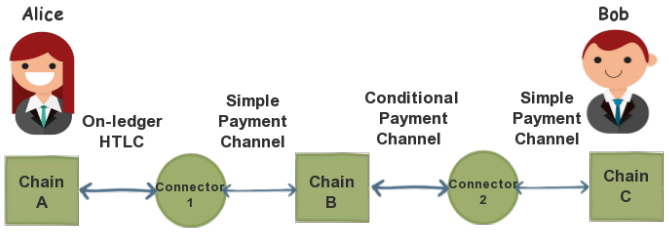

Figure 3.3: Interledger HTLA diagram

Alice and Bob can across between blockchains A, B, and C via HTLA, and each blockchain supports different cross-chain protocols. The connector here plays a role in connection and isolation. Linking blockchains that support different cross-chain protocols together, and again isolate them, so that blockchains do not interfere with each other.
HTLA supports multiple cross-chain protocols based on HTLC, some of them are mentioned in Figure **??**.

3.1.2 Complete the transaction confirmation

As we all know, blockchain systems are relatively independent and closed, there's no direct communication way for them to confirm every piece of records that happened. So no matter how it evolves, there will always be a "middle-man" between the two chains, taking on the role of information exchange between the two chains. Here the "middle-man" represent any entity that could interact with two chains, it may be one or a group, maybe the centralized or distributed agency, maybe a separate chain or even a functional module. The "middle-man" usually acts as a node for two blockchains at the same time, so that only one application software can be deployed on the same node to obtain the others' system data.
After the "middle-man" completes the data collection, how to confirm the transaction, where to confirm, and who confirms becomes the key point of this problem. According to different schemes, this process can be summarized in three ways:

- **Notary** [**?**]:
  In the notary scheme, a trusted one or group is used to declare to the chain X that an event has occurred on the chain Y, or that the statement is correct. These groups can both automatically or requested to listen and respond to events. There are 3 different child-schemes came up in the evolution of this model:

– `Centralized Notary schemes`
The centralized notary mechanism is also called the single-signature notary mechanism, usually played by a single designated independent node or institution, which is the simplest mode. Its purpose is instead of letting Alice and Bob, two strange individuals deal, it's not as reliable as indirect transactions with third-party institutions with credit endorsements (such as Alipay). Since Alice and Bob exist in different ledger systems, the notary is technically required to be compatible with two or more systems at the same time.

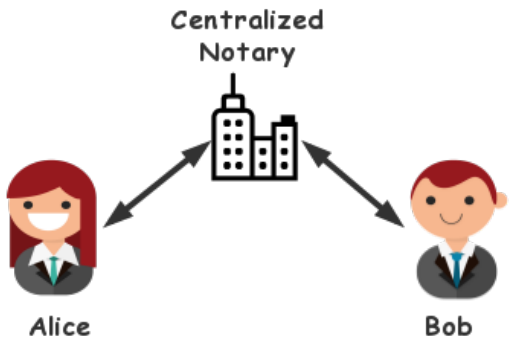

Figure 3.4: Centralized Notary Scheme diagram

To some extent, the use of centralized institutions has replaced technical credit guarantees, from technical credibility to traditional credit intermediaries. Although this kind of mode has fast transaction processing, strong compatibility, and simple technical architecture, the security of the central node has become a key bottleneck for system stability.

– `Multi-sig Notary schemes`
The multi-signature notary mechanism is accomplished by multiple notaries that can sign a common agreement on their respective ledgers to complete the cross-chain transaction. Each node of the multi-signature notary group has its own key, and cross-chain transactions can only be confirmed when a certain number or proportions of notary signatures are reached.

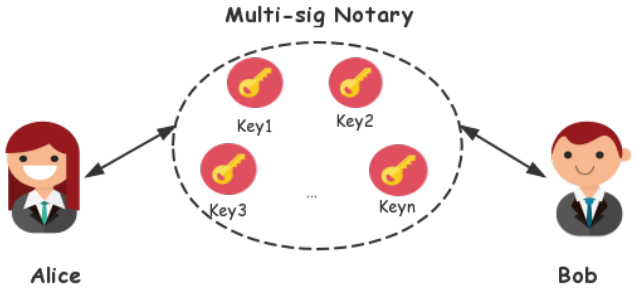

Figure 3.5: Multi-sig Notary Scheme diagram

This method is more secure than the single-signature mode, and a few notaries who are attacked or do evil will not affect the normal operation of the system. However, this approach requires both chains to have the ability to support multiple signatures.

– `Distributed signature Notary schemes`
The main difference between distributed signature and multi-signature is the signature generation. Distributed signature using *Multi-Party Computation*(MPC), which will enhance the security as well as the implementation difficulty.

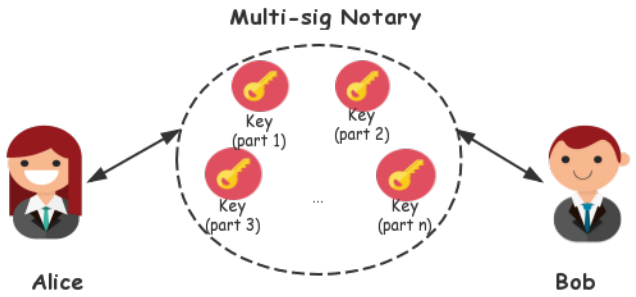

Figure 3.6: Distributed signature Notary Scheme diagram

As Figure **??** shows, distributed signature based on cryptography, the key point is that for cross-chain transactions, the system generates one and only one key, and no one in the notary group will have a complete key. The key is randomly sent to each notary node in the form of fragments. Meanwhile, the fragment is the processed ciphertext, so even if all the notaries put together the pieces, the complete key cannot be known, and the security of the key is fully guaranteed.

- **Relay** [**?**]:
  Relay is one flexible and easy-to-expand cross-chain technology that does not rely on trusted third parties to help with transaction verification. Instead, it is self-verified by the receiving chain after receiving the send chain data. Self-verification methods are depending on the system structure. For example, BTC-relay [**?**] based on *Simplified Payment Verification*(SPV), and Cosmos [**?**] rely on verify the number of nodes' signature.
  Vitalik mentioned Relay in his Chain Interoperability paper [**?**], pointing out that chain A and chain B can use the other party's block data for information synchronization and cross-chain calls. Currently, information synchronization can be done, but there is no mature technical solution for cross-chain calls. Two chains cannot verify the validity of each others block at the same time, otherwise, they will fall into an infinite loop of nesting. If chain A owns the block data of chain B, then chain A needs to be confirmed in the case of chain B transaction confirmation, and chain B needs to wait for chain A's transaction confirmation because it also has block A's block data, it goes on as a loop.

- **Sidechains**:
  The concept of a sidechain as defined in white paper [**?**] is: *sidechain is a blockchain that validates data from other blockchains.* However, this explanation was considered to be too broad and not rigorous by Vitalik Buterin in Chain Interoperability [**?**]. "Sidechain" is more frequently used to refer to what Blockstream calls a "pegged sidechain", where the functionality of a blockchain is of an anchored asset of another asset, which chain is regarded as the parent chain. In this way, this relationship is based on assets, not the blockchain itself. This is a strong coupling cross-chain structure using directly embeds part of the data of the original chain into its own block or storage space. In the case of cross-chain transactions, verification can be completed directly through the original chain data stored in the system. This method is generally considered bidirectionally at the beginning of the system design. Compared to notary and relay, the sidechain is more direct. The state of one chain will be directly reflected in the data of the other chain. When one chain is attacked, the other chain may also be affected. This model is more suitable for the design of the same system, which allows the two sides to become a whole without losing the relative independence of the ledgers.

3.1.3 Realize multiple chains interoperability

The computer network connects the original independent computers into a local area network, the local area network develops into a metropolitan area network, the metropolitan area network evolves into the Internet, and the Internet connects the people of the world like never before.
However, for the emerging technology/industry of blockchain, it is still in the "single-machine" era, and the interactions demand between chains and chains will become increasingly strong with the application of blockchain.
To realize interoperability among multiple chains, there are two potential aspects of difficulties that need to overcome:

- How to achieve interoperability among blockchains system that has already developed.
- How to prepare/setup the way for the interconnections among the new blockchains in the future.

3.1.3.1 Active compatibility

This solution is mainly aimed at the existing blockchain system. First, there are different blockchain application systems in the upper layer, and then the underlying cross-chain mechanism is developed.
Usually these systems are heterogeneous chains and need to be docked one by one, but there is also a different solution for a pair of connections.

1. Direct interconnection between the two chains

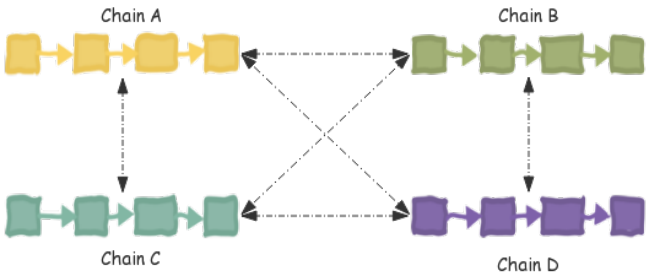

Figure 3.7: Direct interconnection network architecture diagram

This method is the most time-consuming and laborious without the support of the unified underlying protocol. It is necessary to establish 6 paths between

the 4 chains to realize the interconnection between them. And each path needs to be customized. Although this method is not scalable, it can guarantee better security and independence. Once an attack occurs, it is difficult to affect the entire network.

2. Third-party cross-chain platform

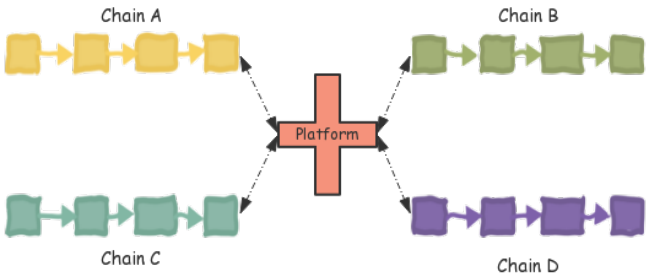

Figure 3.8: Cross-chain platform network architecture diagram

To establish a cross-chain platform, blockchains can indirectly interconnect with each other. Thus, only 4 paths are required to build a cross-chain network. However, in this method, the cross-chain platform will become the key point and performance bottleneck of the entire cross-chain network (not necessarily the bottleneck, but it may be in the future). Once the cross-chain platform is attacked, the entire cross-chain network will be paralyzed.

3.1.3.2 Passive compatibility

Passive compatibility is mainly aimed at the blockchain system that has not been developed. It first builds the underlying cross-chain platform, allowing other blockchain systems to be easily, conveniently and securely accessed. Cross-chain platforms will prioritize the development of systems and protocol standards that apply to interoperability between the various chains. The subsequent development of standards-compliant development on existing platforms allows for the creation of blockchains that naturally have cross-chain functionality within the system. However, the cross-chain mentioned here refers to the chain that conforms to the protocol standard can be easily connected to each other. If it is to interoperate with other chains outside the system, it is necessary to develop a separate middleware to communicate.

In addition, different cross-chain platforms can support different types of blockchains, such as Cosmos supporting isomorphic chains and Polkadot supporting heterogeneous chains, both of which are highly scalable, and will be later discussed in Section **??**.

3.2 Project study

3.2.1 Lightning Network

In general, we can not say lighting network realize the cross-chain function, though it provided a classic application towards atomic swaps and HTLC. The design idea of lighting network is very simple, it put a large number of high-frequency small-value transactions off-chain to expand the transaction processing capability of the blockchain.
Lightning network [**?**] is a fast and scalable Bitcoin transaction project, it has two main technical points:

A. `Recoverable Sequence Maturity Contract`
   RSMC is similar to a reserve mechanism in which both parties trade in an off-chain trading pool. A certain amount of assets is used as the mutual funds for the transactions between the two parties, and the share of the assets is recorded off-chain. This trading pool is a "micro-payment channel". When a transaction occurs between two parties, the proportion of the common assets in the trading pool will change. The new proportional data needs to be signed and confirmed by both parties, and the old proportional share version is invalid. The entire process is done off the chain, so it does not occupy the resources of the main chain. The final proportion of assets will be confirmed and record to the main-chain after one of the transaction party requires a withdraw.
   It could happen anytime as long as both parties signed for this. To ensure the security, if someone submitted the old share of assets to make profits, others could protect themselves by proving this balance sheet is not the latest one. Then the asset of the counterfeit party will be confiscated to the challenger.

B. `Hash Time-locked Contract`
   Lightning Network uses HTLC to guarantee the atomicity of transaction as shown in Figure **??**. This diagram simply illustrates how HTLC works to provide limited time transfer function. The basic process is: Bob and Alice can reach an agreement that specifies Alice to lock a certain amount of assets and provide a hashed value H. Before the arrival of time T, if Bob can learn an appropriate s (secret) where its hash value matches with H and sent to Alice, Bob can get the corresponding amount of assets value. Conversely, the asset will unlock and return to Alice.

When there are "micro-payment channels" between multiple users, these channels are connected to each other to form a "channel network", which is the lightning

network. The mutual transfer of the two parties does not require a direct payment channel to connect to each other, but also through the intermediary to achieve mutual transfer.

Disadvantages:

- Users cannot pay off-line.
- More suitable for small transactions. Large transfer amount needs to open multiple channels.
- Easy to face the dis-matched situation, if there's no response from one party, the other one may need hours to close the channel and substitute with another route.

3.2.2 Notary Scheme

3.2.2.1 Ripple Interledger protocol

Earlier we focused on cross-chain, this project is more focused on cross-ledger, which means that the agreement not only supports decentralized blockchains but also supports various centralized ledgers, which is broader support for cross-chain applications. Ripple ILP [**?**] uses the HTLA and Notary scheme to implement this technology.
Ripple is the first project to propose the use of blockchain technology to achieve cross-ledger exchange of assets, with a focus on resolving cross-border remittances, enabling faster and more economical international remittances via the Ripple network.
Interledger Protocol(ILP) is compatible with any online ledger systems. Specifically, the ILP will establish a two-way pegged relationship between the trader's account and a Ripple local account, enabling simultaneous changes between the two to ensure transparency in the transaction process. At the same time, for two ledger systems that do not have a direct payment channel, multi-hop indirect cross-ledger transactions can be realized through ILP.
The main idea of ILP is to secure cross-ledger transactions by setting up *escrow account* on Ripple. So the process will need the preparation of escrow account of several parties in the transaction. As an example in Figure **??**, Alice, Bob, and one selected market maker should have their own Ripple escrow accounts set up on two bank systems before the transaction.

- Alice first selects a market maker with the most suitable exchange rate, and fill in the remittance information, receipt address and timeout period on the Ripple application.

- This information will be packed by the Interledger Module and sent to the Ripple Account 1, Ripple Account 1 records the changed amount of currency in the escrow account 1 and sends the transfer certificate to the **Validator**

- For Bob, Company B fills in the Ripple application with information such as the remittance address and timeout period and broadcasts it on the Ripple network. At this time, the liquidity provider selected by A will transfer a certain amount of assets in B's currency from Ripple Acc.3 to Ripple Acc.2 in advance, then send the transfer certificate to the **Validator**

- Validator checks the two transfer certificates; after the verification is passed, the ILP ledger will be liquidated simultaneously according to the Hashed Time Lock Agreement.

- The final step is when liquidation completed, Ripple will synchronize all account changes through an interledger module, thus realizing the cross-ledger transactions.

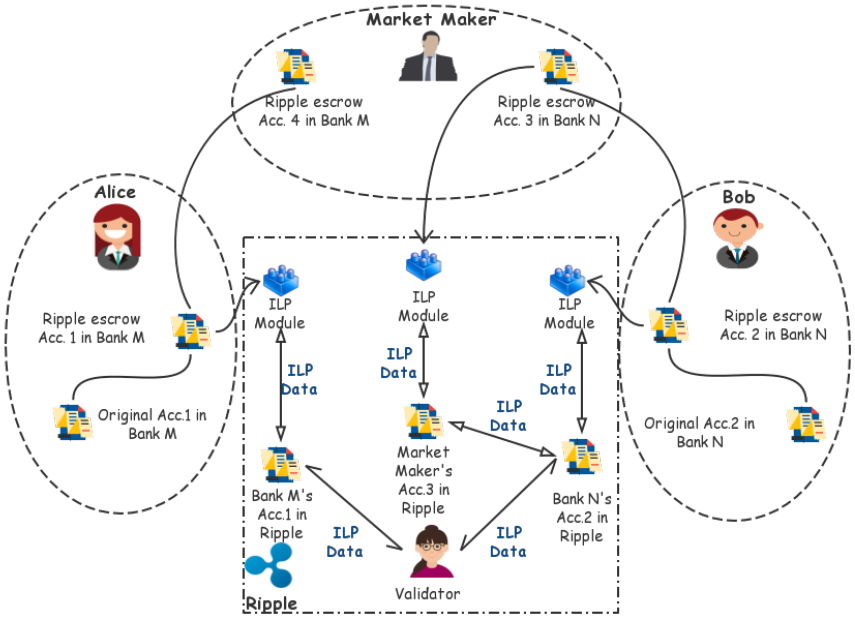

Figure 3.9: ILP example process

3.2.2.2 Liquid

Liquid [**?**] is a sidechain of BTC and a typical representative of the multi-signature notary mechanism. It is designed to meet the BTC fast transfer needs of exchanges, market makers and brokers. Therefore, Liquid uses the multi-sig federation mechanism to confirm the transaction block, which can greatly improve the transaction speed.
Liquid based on Element codebase and uses Strong Federation technology to support 1:1 Bitcoin exchange. The basic process of the transaction is described as Figure **??**. The confirmation of this asset transfer requires multi-signature from the majority of notaries.

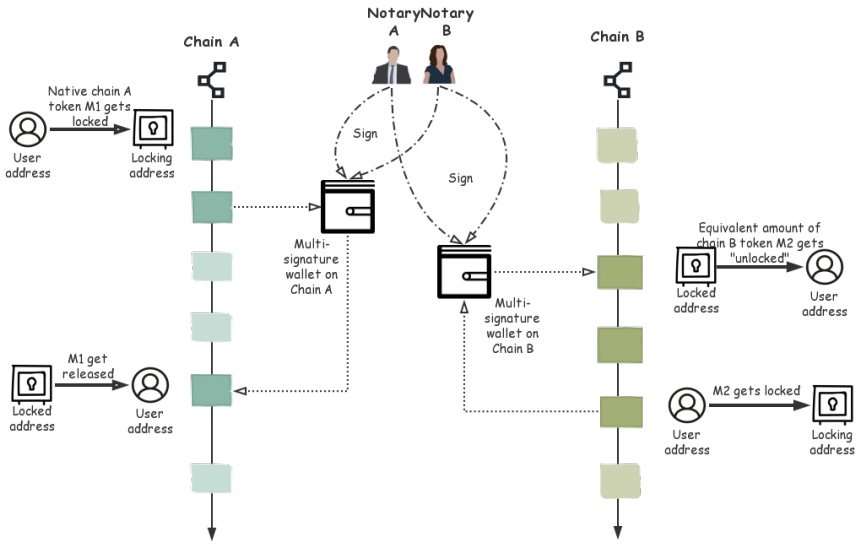

Figure 3.10: Multi-signature federation scheme

In Strong Federation, there are 2 types of node role in the network:

- Blocksigners: Signature verification for transactions in the sidechain to achieve block consensus.

- Watchmen: When the asset is transferred from the sidechain to the main chain, it is responsible for signature verification of the transaction on the main chain, indicating that the sidechain asset has indeed been destroyed, and the main chain can unlock the corresponding number of assets.

3.2.2.3 Wanchain

Wanchain [**?**] is a cross-chain platform project initiated in 2016. It is a heterogeneous cross-chain framework that implements cross-chaining based primarily on distributed notary scheme. This model mainly uses cryptography "Secure Multi-Party Computation" and "Threshold Key Sharing Scheme" to implement the Authenticator distributed signature.

Wanchain provides the infrastructure for asset cross-chain transfer channels for different blockchain networks, realizing the transfer of assets between Wanchain and original chain. Wanchain3.0 now launches bridges from Bitcoin to the Ethereum network. The transaction reliability verification is completed by multiple Storeman of Wanchain. The following figures**????** are shown the transfer process between Wanchain and Ethereum.

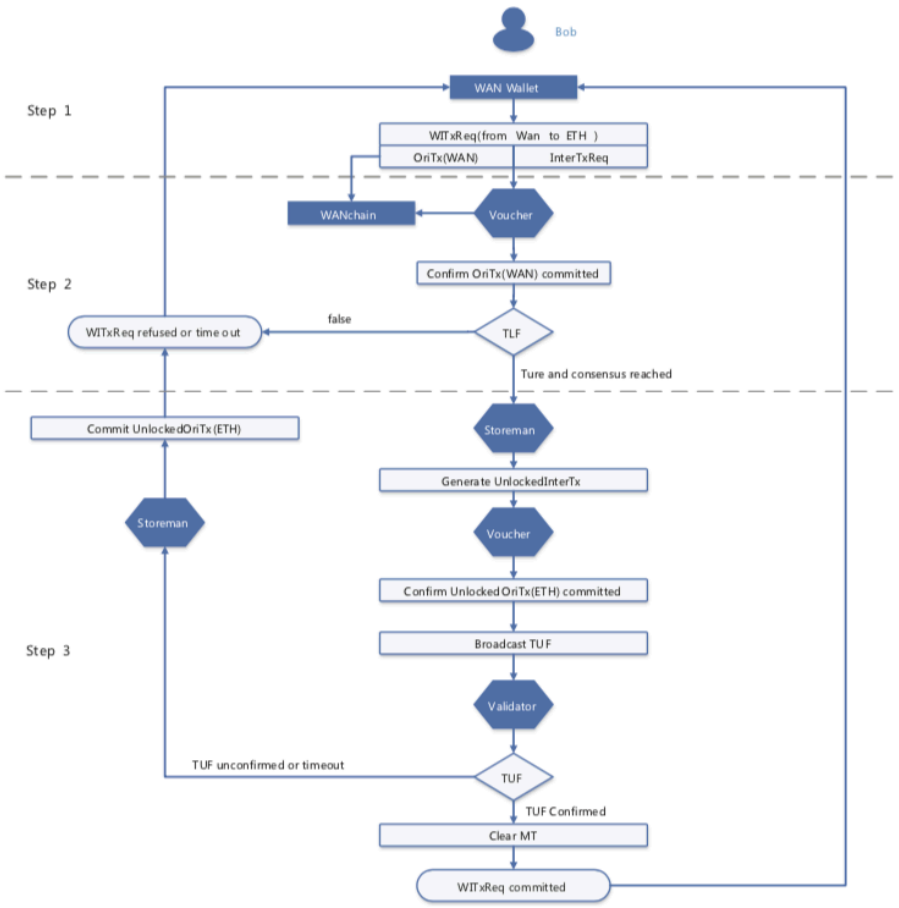

Figure 3.11: Data transfer process from Ethereum to Wanchain[1]

[1]Image courtesy of Wanchain white paper [**?**]

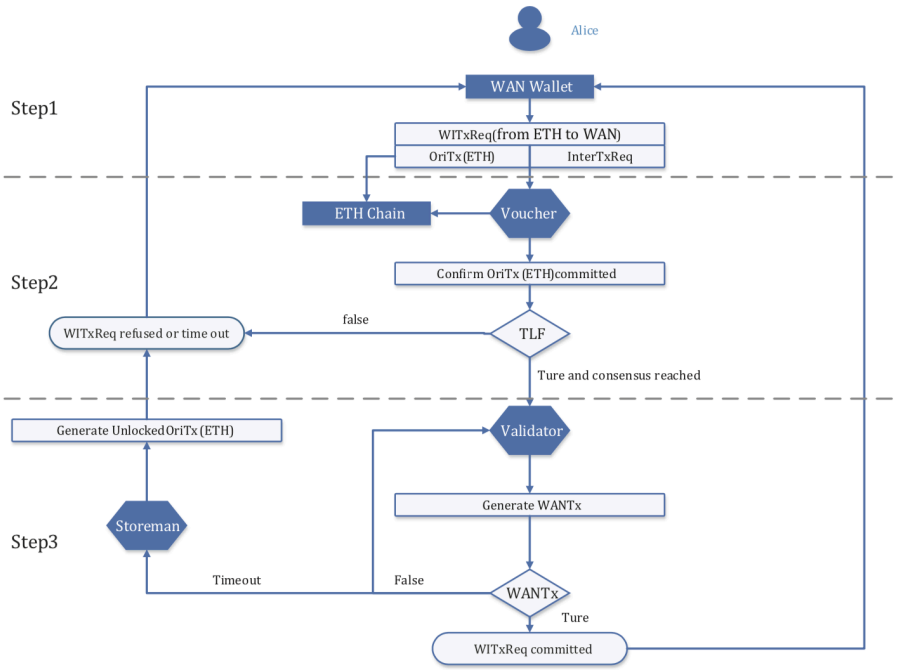

Figure 3.12: Data transfer process from Wanchain to Ethereum[2]

1. The token of the user in the original chain will be sent to Wanchain in the locked account, and the transaction is locked by Hash Time Lock;

2. After the **Voucher** verified the transaction on the original chain, the **Storeman** will initiate a cross-chain contract transaction on the Wanchain, and transfer the mapping token in wanchain to the user's cross-chain account on Wanchain, and locked;

3. After the user's wallet detects the transaction locked by the cross-chain contract, release the Secret to the cross-chain contract;

4. Storeman obtains the control of the original chain token through the secret number, thus achieving confirmation of the original chain transaction.

5. If the user does not release the secret number within the scope of the hash time lock, the hash time lock expires

[2]Image courtesy of Wanchain white paper [**?**]

6. The transaction of the post-cross-contract is automatically invalidated, and the user regains control of the original token.

In Wanchain, when Storeman locked an account, the private key of the locked account is scattered into multiple pieces and send to Storeman, and it requires more than a certain percentage of Storeman complete the signature before the final confirmation. In order to avoid conspiracy, Storeman has to pay a certain amount of token to participate in the verification in case of doing evil. To ensure atomicity, Wanchain uses a hash time lock to lock cross-chain transactions, ensuring that no user or Storeman will complete a one-sided transaction.

Since the Wanchain mechanism does not change the original chain, it is necessary to adapt the development according to the characteristics of the original chain, which is also the difficulty of heterogeneous cross-chain solution. The transaction speed is affected by the confirmation speed of the original chain.

Fusion [**?**]

Similar to the principle of Wanchain, Fusion's goal is to build a basic platform for the operation of encrypted financial applications based on blockchain technology. On this platform, a variety of tokens can be freely interacted through smart contracts to achieve value interoperability. Support multi-platform cross-chain asset transfer, using a distributed signature notary model for cross-chain transaction processing.

Based on Hierarchical Hybrid Consensus Mechanism(HHCM), Fusion combines the advantages of PoW and PoS consensus mechanism, which balancing the safety, efficiency, and scalability. It is the only cross-chain platform that offers parallel computing, multiple triggering mechanisms, and off-chain data support.

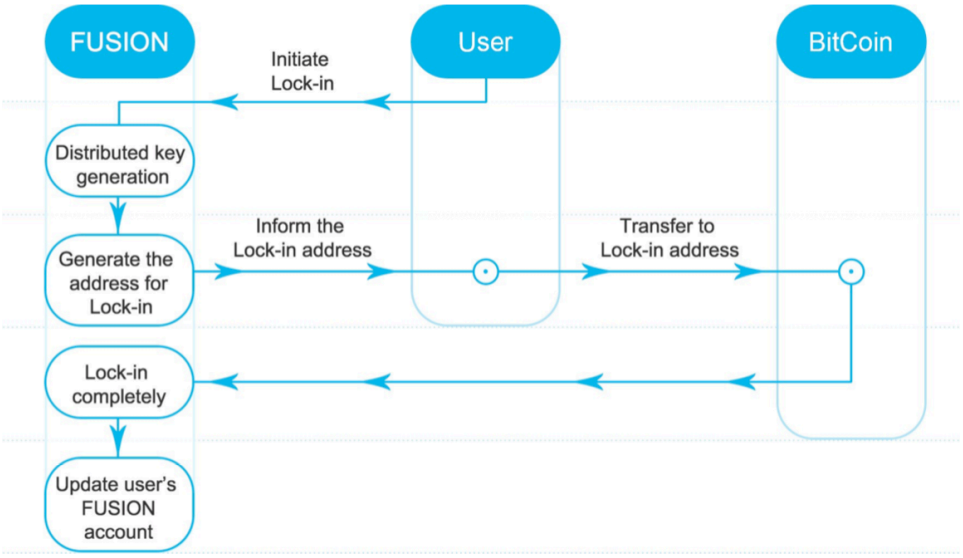

Figure 3.13: Fusion Network Architecture, lock-in process[3]

The realization of cross-chain assets transaction is based on the lock-in&lock-out process as shown in Figure **??**.

3.2.3 Relay Scheme

3.2.3.1 BTC-Relay

In 2016, the BTC-Relay released by the Consensys team is the most classic relay cross-chain solution, enabling cross-chain transactions between Ethereum and Bitcoin, as well as realizing Ethereum's DApp applications to support BTC payments. Since Bitcoin scripts are non-turing complete and difficult to support complex applications, BTC-Relay only implements one-way cross-chain from Bitcoin to Ethereum.

[3]Image courtesy of Fusion white paper [**?**]

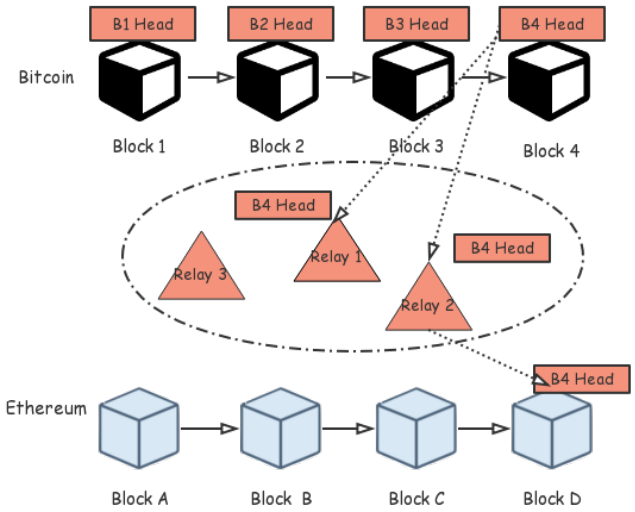

Figure 3.14: BTC-Relay cross-chain process diagram

Like Figure **??** shows, BTC-relay itself is a smart contract for Ethereum. The function of the contract is to verify certain transactions on Bitcoin and provide verification information to other DApp users on the Ethereum. Relay is a group of users who obtain block header data from Bitcoin and has the account address of the Ethereum network. The Relay that submits the block header data to the BTC-Relay contract as soon as possible can get the ETF transaction fee reward. After obtaining the block header data, the BTC-Relay smart contract can verify a transaction according to the principle of SPV proof. When a transaction in the Bitcoin network does occur, it can trigger the specific transaction or smart contract execution of the Ethereum network.

3.2.3.2 Cosmos

Cosmos [**?**] is a cross-chain platform project initiated by the Tendermint team in 2017. It supports the modular establishment of Cosmos isomorphism chain and also supports the external heterogeneous chain through Bridge. Its most important feature is that all the chains in the Cosmos system are isomorphic chains and can more easily support the flow of assets across the chain. All the zones share a set of network protocols, consensus mechanisms, and data storage methods. Assemble the new Zone blockchain through the API interface.
The overall architecture of Cosmos network are shown in Figure**??**:

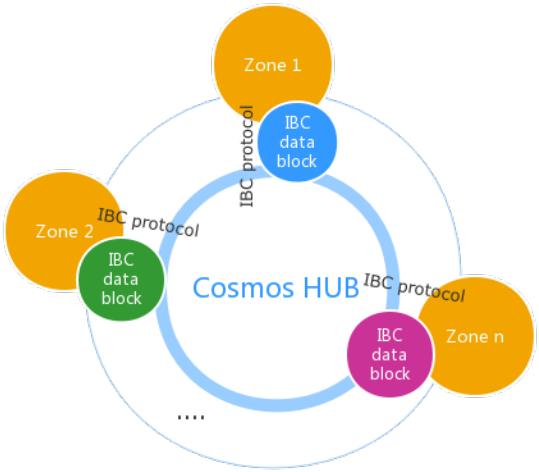

Figure 3.15: Cosmos network architecture

There are many Zones connected to the Hub (Hub is a chain, and each Zone is also a chain). Cosmos Hub maintains a multi-asset distributed ledger and masters the asset status of all the Zones that connected to it. Each zone will synchronize the status of the Hub, but the communication between the Zone and the Zone can only be done indirectly through the Hub. Each cross-chain asset transfer requires a common confirmation of the zone, hub, and receiving zone to be successful.

The information is transmitted between zones through packets based on the IBC (Inter-Blockchain Communication) protocol. The blocks in a space pack the data to be delivered into standard IBC data packets and finally complete the transmission through the network layer UDP or TCP protocol.

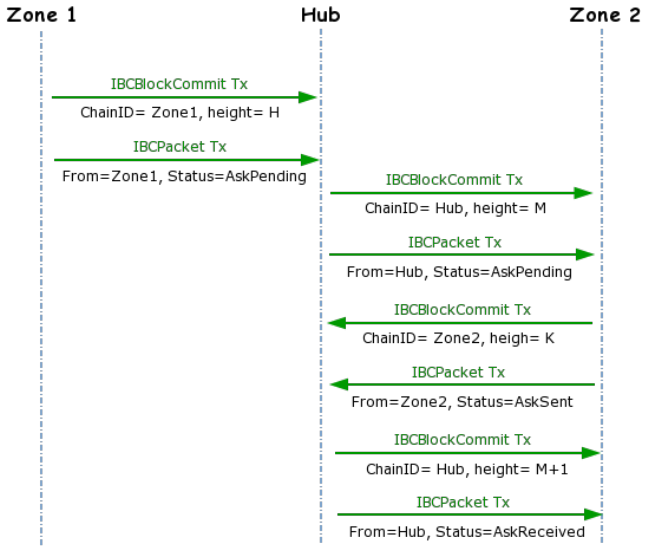

Figure 3.16: IBC sequence diagram

1. Zone 1 initiates an IBCBlockCommitTx transaction, passing the new block header information (including all Validator's public key) to the HUB;

2. Zone 1 initiates an assets transaction;

3. Zone 1 verifies the transaction;

4. Send the valid transaction into the message queue for HUB;

5. Zone 1 listens to a new message in the queue, which generates Merkle proof, and sent to HUB as IBCPacketTx's Payload. (In each space there is an independent third-party relay program that generates Merkle Proof from the original chain and assembles it into a Packet and initiates the transaction, passing it to the receiving chain);

6. HUB verifies that Merkle Proof is valid, if it is valid, send a message to Zone2 (The process of sending a message to Zone2 by HUB is the same as steps 1 6);

7. Zone2 verifies that Zone1 is a valid transaction after receiving the HUB message. Send a message to HUB confirming that it can receive assets from Zone1;

8. The HUB sends a message to Zone2, sends the asset to Zone 2, and the asset transaction is completed. Due to an attack or network error during the transfer process, it is possible for the message sent by Zone 2 to the HUB is lost, as shown in the figure**??** below, after waiting for a period of time, the HUB sends a message telling Zone1 that the current transaction is timeout and the transaction fails.

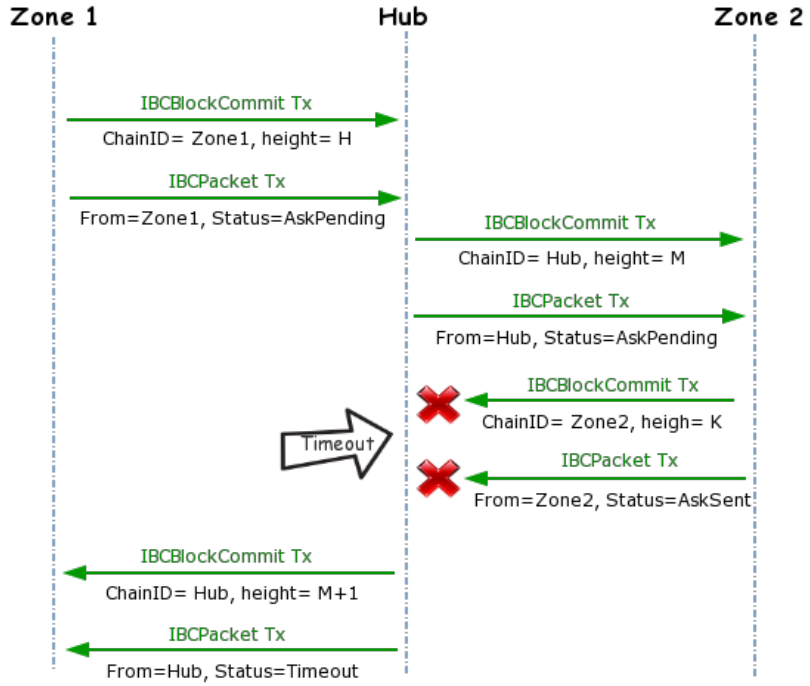

Figure 3.17: IBC sequence diagram, timeout

Cosmos based on **Tendermint** consensus which is the combination of PBFT and PoS, it improves the processing power of Cosmos Network. The cross-chain transaction between Cosmos and other heterogeneous blockchains need to be carried out through Cosmos Bridge, Bridge-Zone will be responsible for the docking with the original chain, including the confirmation of the original chain transaction, create or destroy the corresponding tokens.

Overall, Cosmos is a blockchain development framework, allowing developers to focus on their own business without having to consider the underlying technology of the blockchain so the plug-in design can use Cosmos as needed.

3.2.3.3 Polkadot

Polkadot [**?**] is a cross-chain project supported by the Web3 Foundation. Creating a scalable, heterogeneous, multi-chain architecture designed to address blockchain interoperability, scalability, and security.
The overall architecture of Polkadot is shown in the figure below, consisting of a Relay Chain, Parachains and Bridges.

- **Relay chain** is the central system of the Polkadot network, which coordinates the consensus and transactions between the chains, records account information and transaction status;
- **Parachains** can be built by developers to collect and process transactions, and transfer to Relay chain;
- **Bridges** connect other heterogeneous blockchains (such as Ethereum) with Polkadot network.

There are four roles in Polkadot's network:

1. Collators (collecting user transactions, validating and submitting to Validators),
2. Validators (validating and broadcasting transaction data committed by Collators, verifying blocks and paying deposits)
3. Nominators (investment deposits selected for trust Validators)
4. Fishermen (supervising the evil behavior in the network).

Collators and Validators are the performers of the main transaction when doing cross-chain operations, while Nominators and Fishermen are the participants in maintaining system trust.

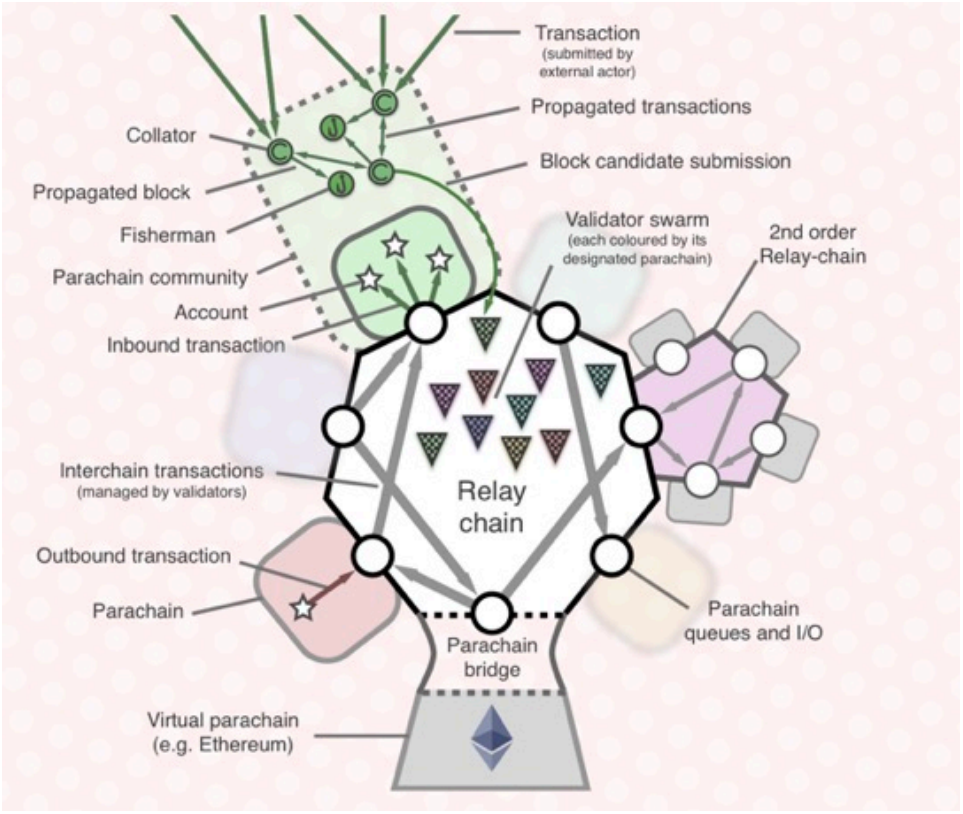

Figure 3.18: Polkadot network architecture [4]

Parachain can share trust of the entire Polkadot network, but it also gives a certain confirmation right to the Relay chain; when the user's transaction information on Parachain A is transmitted to Parachain B, the process is as follows:

1. The initiated transaction is sent to the Collator on Parachain A;
2. Collator validates the transaction and packs it into the block;
3. Collator submits the block and state transition proof to the Validator on Parachain A;
4. The Validator verifies the block it receives that contains only valid transactions and pays a certain amount of deposit;
5. After enough Nominators have paid a deposit for the Validator, they broadcast the block to the Relay Chain;

[4]Image courtesy of Polkadot white paper [**?**]

6. The transaction of data has been executed.

Unlike Cosmos, Polkadot supports cross-chain interoperability between heterogeneous chains. It does not only support asset transactions, but also the data transfer. Its system complexity and implementation are more difficult than Cosmos.

3.2.3.4 ICON

ICON [**?**] is committed to building a cross-chain network that connects all types of blockchain systems, enabling DApps to be interconnected across all types of blockchains. The cross-chain transactions primarily handled through notary mechanism. Figure **??** below shows the overall conceptual model of ICON. With ICON, blockchains are connected around the **Nexus**, which is a loopchain-based blockchain. The whole system based on loop fault tolerant mechanism which is an enhancement of BFT-based algorithm.

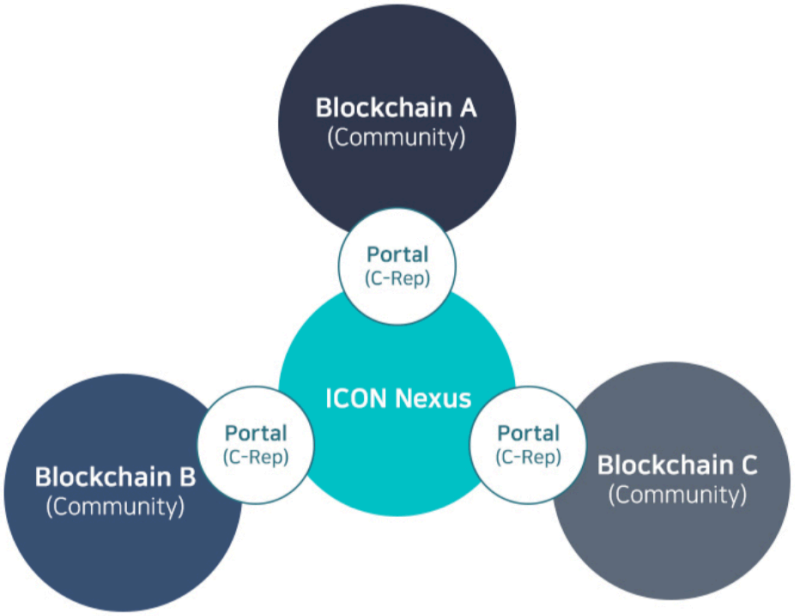

Figure 3.19: Conceptual model of ICON[5]

[5]Image courtesy of ICON white paper [**?**], Nexus is a Multi-Channel blockchain comprised of Light Client of respective blockchains. Each blockchain connected to Nexus via a portal.

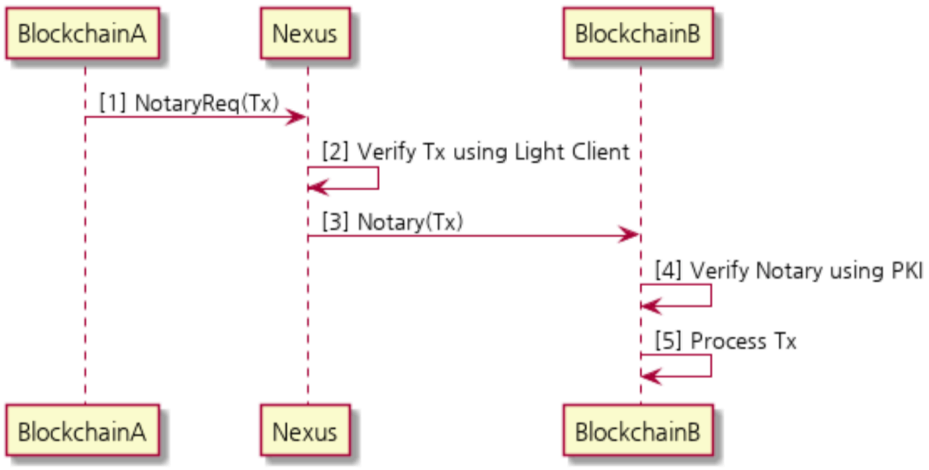

Figure 3.20: Assets transaction process through Nexus[6]

Although rated as one slow progress cross-chain project, ICON still has the ambition to not only connect blockchains together but also aiming at realizing communication between the traditional ledger system and the blockchain world.

3.2.3.5 AION

Among the Interoperability Alliance[7] , AION differs from Wanchain and ICON. While the other two projects still focus on the assets and value transactions cross-chain, AION also expanding the business logic and interoperability between chains. Figure**??** represents a simple multi-tier network architecture based on AION.

[6]Image courtesy of ICON white paper [**?**]

[7]Wanchain, ICON and AION form an alliance to overcome the technical difficulties towards cross-chain technology. reference: https://medium.com/helloiconworld/blockchain-interoperability-alliance-icon-x-aion-x-wanchain-8aeaafb3ebdd

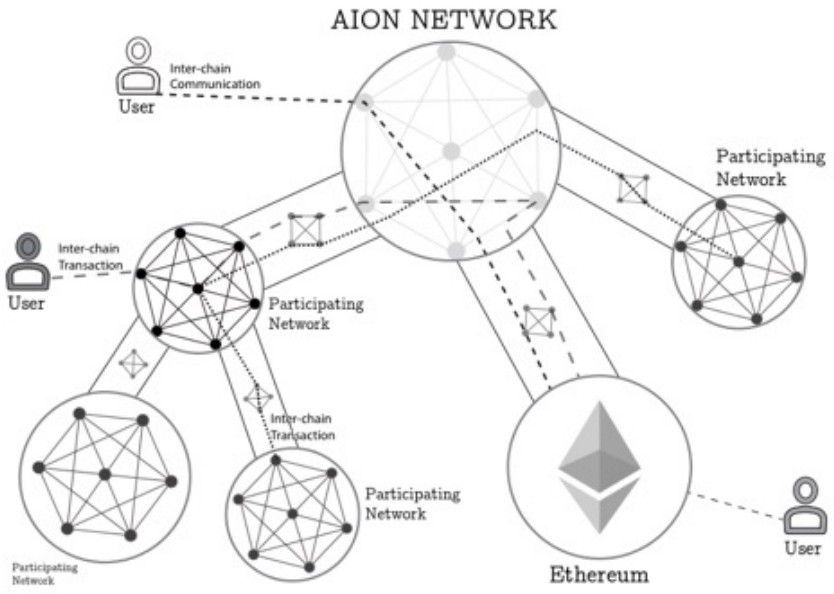

Figure 3.21: Multi-tier network structure of AION[8]

AION aims to establish a multi-tier cross-chain platform that supports interoperability of heterogeneous chains. By connecting networks and bridges to blockchain systems, the route of cross-chain transactions is a multi-stage process.
At each stage, the Validators verifies the transaction and agrees on whether the transaction is forwarded or rejected. Bridge validators will use a lightweight BFT-based algorithm to reach consensus. If a transaction is rejected at any time, any state changes due to cross-chain transactions will be revoked, at least in the connected network.
AION has different consensus mechanisms at different product stages, the latest AION 3.0 using mixed DPoS and PoI algorithm.
AION is an emerging project and still in the early stage of development.

3.2.3.6 Quant Overledger

Developed by Quant Network, Overledger is the one Blockchain Operating System that facilitates the development of multi-chain smart contracts [**?**]. In order to not limit the inter-communication between 2 blockchains at the same time, Overledger enables reading the transactional, contract and script information and map them into one "over layer". And in most cases where the transaction requires multiple hops during the route to the destination, Overledger will create a common interface

[8]Image courtesy of AION white paper [**?**]

among ledgers to solve this issue.
Different from other projects who devote their cross-chain solution into the transactional layer,

3.2.4 Sidechain Scheme

3.2.4.1 Plasma

As the second-layer expansion framework of Ethereum, Plasma has been proposed by Joseph Poon (founder of Lightning Network) and Vitalik Buterin (founder of Ethereum) in 2017 [**?**]. The first thing to be clear is that Plasma is essentially a set of frameworks instead of a separate project. It provides an off-chain solution for a variety of different project real projects.
Layer 2 expansion of blockchains often known as off-chain expansion, similar to lightning network, this kind of expansion scheme does not need to modify the underlying protocol of the blockchain, but by transferring a large amount of frequent calculation work "off-chain", and submitting the calculation results to the "main chain" guarantee its finality as we discussed previously. Plasma works like blockchains in blockchains where anyone can create different Plasma on top of the underlying blockchain to support different business needs.
As an example of sidechain, Plasma derived from the general concept of symmetric 2-way pegged scheme to realize the transfer solution. The overall process of asset exchange is shown in figure **??**.

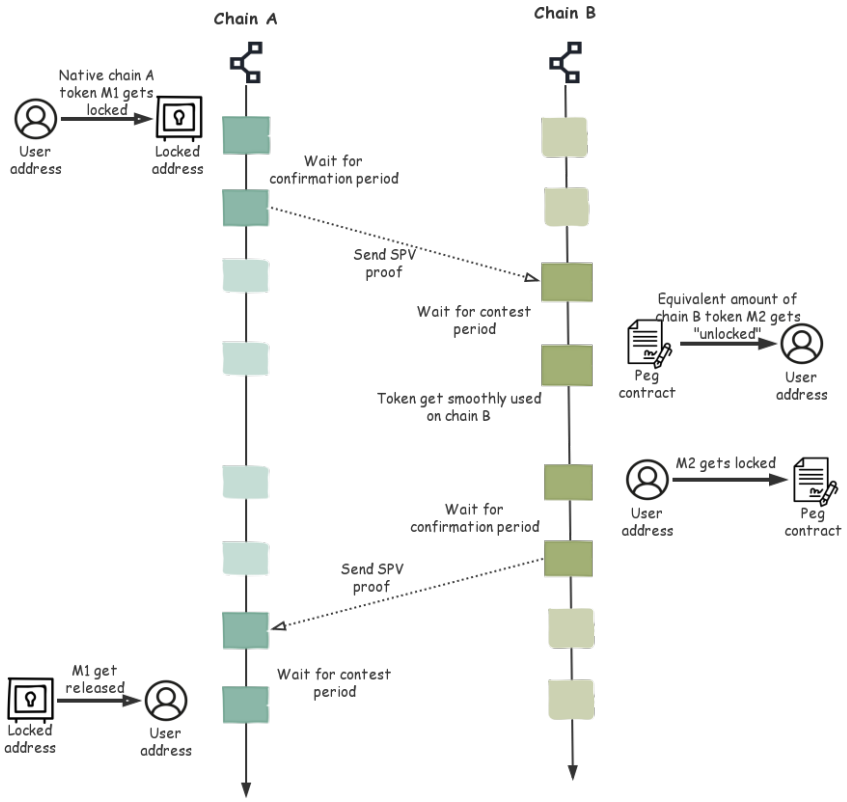

Figure 3.22: 2-way pegged sidechain diagram[9]

1. When chain A wants to transfer the asset to chain B, it first needs to initiate a transfer transaction Tx1 (chain A's locking addr1 + chain B's receiving address addr2), and the asset M1 is locked on the addr1.

2. After Tx1 transaction is submitted, it is necessary to wait for a *confirmation period* so that there are enough blocks and calculations to ensure that the cross-chain transaction Tx1 is confirmed, reducing the impact of refactoring on cross-chain transactions.

3. After the confirmation period, the **SPV** certificate containing Tx1 will be sent to chain B. B knows that chain A has indeed initiated and locked asset

[9]SPV to verify that the transaction exists (recognized by other nodes on the blockchain)

M1, so it generates a corresponding amount of M2 on chain B according to a certain ratio. The value of M1 is transferred to M2 means the assets on chain A are transferred to chain B.

4. After M2 is generated on B, it is necessary to wait for the *competition period* before unlocking M2 to avoid double-spend attack in chain A reconstruction.

5. After unlocking, M2 can freely circulation on chain B.

6. The process of a transaction from chain B to chain A is similar to previous steps.

Plasma supports multi-level sidechains and uses MapReduce mode to perform parallel computing, which greatly improves sidechain performance. The block header and hash data of the side chain will be sent to the main chain, and *Proof of Fraud* can be used to ensure the correctness of the sidechain transaction.

3.2.4.2 Elastos

In order to ease the pressure on the main chain as well as provide better user experience for DApps, Elastos [**?**] adopted the main chain + sidechain architecture, main chain only responsible for the circulation of the ELA while the DApps run on the sidechain.
In this scenario, the transfer of the assets between the main chain to sidechain is in a one-to-many relationship. It is feasible that the side chain only saves all the block header information of the main chain. If the main chain needs to save the block header information of all the side chains, it will lead to poor scalability. So Elastos uses asymmetric 2-way peg scheme based on SPV to realize the cross-chain function.
Assets from the main chain to the sidechain, Elastos using SPV proof to prove the transactions, while it secures the transfer using multi-signature notary scheme when transfer from sidechain to the main chain. We can regard this process as a combination of Plasma and Liquid.

3.2.4.3 OneLedger

As one cross-chain consensus protocol, OneLedger [**?**] uses sharding and improved practical Byzantine fault-tolerant consensus. By creating sidechain, it can easily realize cross-chain interactions between individuals or business in OneLedger. OneLedger defines a three-layer consensus protocol to integrate different blockchain applications more efficiently.
Different from what we have discussed before, OneLedger as a sidechain, it realizes the synchronization of assets and values between the main chain and sidechain by applying multi-sig federation and drive-chain.
A drivechain [**?**] gives custody of the locked coins to the miners, allowing them to

(algorithmically) vote on when to unlock coins and where to send them. As Figure **??** shows:

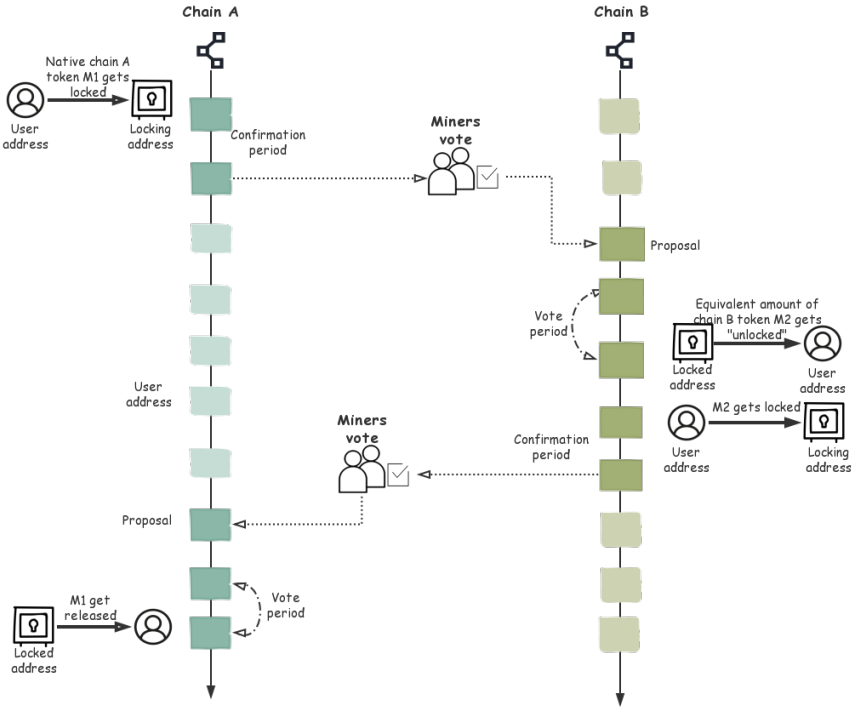

Figure 3.23: Drivechain working diagram

According to OneLedger's white paper, the core of OneLedger is a set of consensus protocols that enable OneLedger to effectively integrate different blockchain products. As far as I understand that the protocol mentioned here is not a specific consensus protocol algorithm in the traditional sense, but a series of concepts and application scenarios. Among all 3 layers, I specifically studied the *Public Chain Consensus* which apply on the atomic transfer between blockchains through OneLedger Network on the base layer.

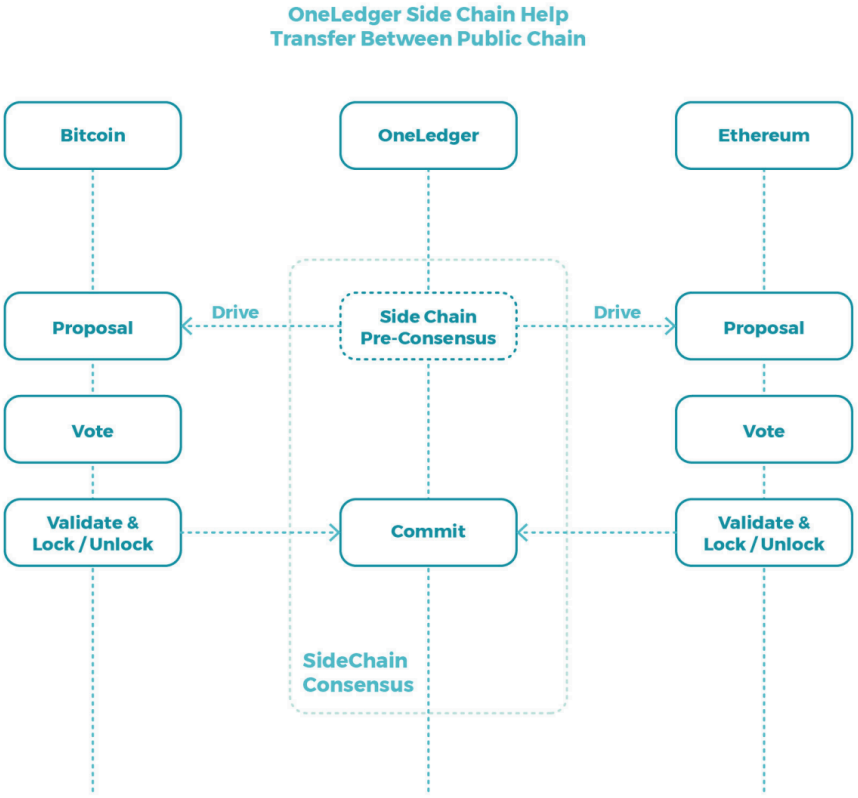

Figure 3.24: OneLedger sidechain architecture[10]

There's 2 steps in sidechain consensus algorithm:

- **Round base pre-consensus**: Use to obtain a consensus proposal with more than 2/3 participants' votes.
- **Commit**: When the purposed block has reached a pre-consensus stage, it needs to drive to the public chain when necessary, and accepts the verification process. Once the proposal is accepted by both public chains, the new block will be officially 'committed' to the OneLedger network, and once more than 2/3 of the participants complete the commits, the block is finalized.

3.2.5 PalletOne

To face with the lack of interoperability in the blockchain world, PalletOne aims to become the "IP protocol" of the blockchain network, which provides an operating

[10]Image courtesy of OneLedger white paper [**?**]

environment for decentralized applications via abstract interfaces. [**?**]

On the one hand, PalletOne encapsulates all underlying blockchains into adapters in the form of interfaces so that it could exchange values and data with different blockchains. On the other hand, the PalletOne virtual machine provides a secure and stable smart contract running environment for common programming languages. Developers can write cross-chain blockchain applications using their common development language without paying attention to the details of the blockchain. At the same time, PalletOne's original jury mechanism and the Mediator mechanism of DAG Data Storage and DPoS enable both contract execution and data storage to be processed in parallel, enabling a high-performance "super-chain."

According to their technical yellow paper, they try to address scalability issues, enhance user-friendliness, and transaction issues in different chains. However, the core problem that PalletOne solves is the problem of cross-chain interoperability. This is also the place where we focus on. The most creative feature in PalletOne is the consensus mechanism, it is more complicated and divide the nodes into two roles:

- **Jury**
  Instead of the traditional consensus model, PalletOne delegates the operation of smart contracts and the management of multi-signature accounts to the jury. The jury randomly selected by the candidate jurors will use the consensus to reduce the occurrence of network congestion, and at the same time use the deposit penalty mechanism to ensure that the jurors do not do evil.

- **Mediator**
  Mediator in PalletOne is responsible for the overall security of the whole PalletOne network, so Mediator needs to ensure that all decisions are correct. It is similar to the function of miners in traditional blockchains to create blocks. The nodes elect Mediator by using DPoS consensus. In order to prevent Mediator from becoming a performance bottleneck for PalletOne, most of the work only needs to be done by the jury without calling Mediator.

The following Figures**??** and **??** provided by the official yellow paper are the vivid examples of the cross-chain process between BTC and ETH with different endorsement.

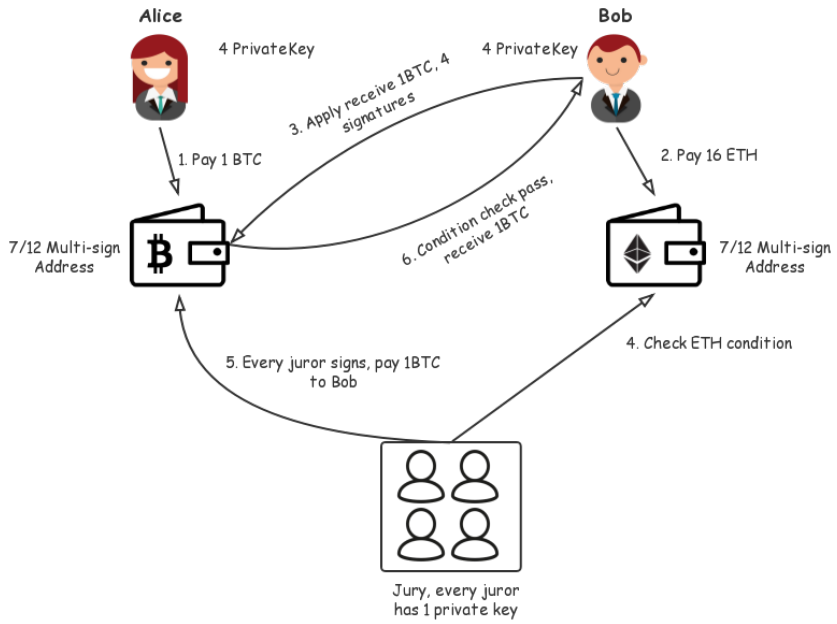

Figure 3.25: The normal process of executing the exchange contract (with Jury endorsement) [11]

1. Jury members and two user accounts first executed the proper contract that generated 7/12 multi-signature address for both crypto-currency.

2. Then both parties transferred a certain amount of exchange currency to corresponding multi-signature address.

3. Each party will initiate the signature request with their own 4 private keys to collect the tokens.

4. Jury verified the validity of both contracts and checked the status of two multi-signature address. If failed, go back to step 2.

5. After verifying the satisfaction, each juror will sign with the private key and invoke the transactions.

[11] Image courtesy of PalletOne yellow paper [?]

6. If one party of this transaction failed or timed-out, the jury will terminate the contract and there will be a compensation to the other by the one caused the fault.

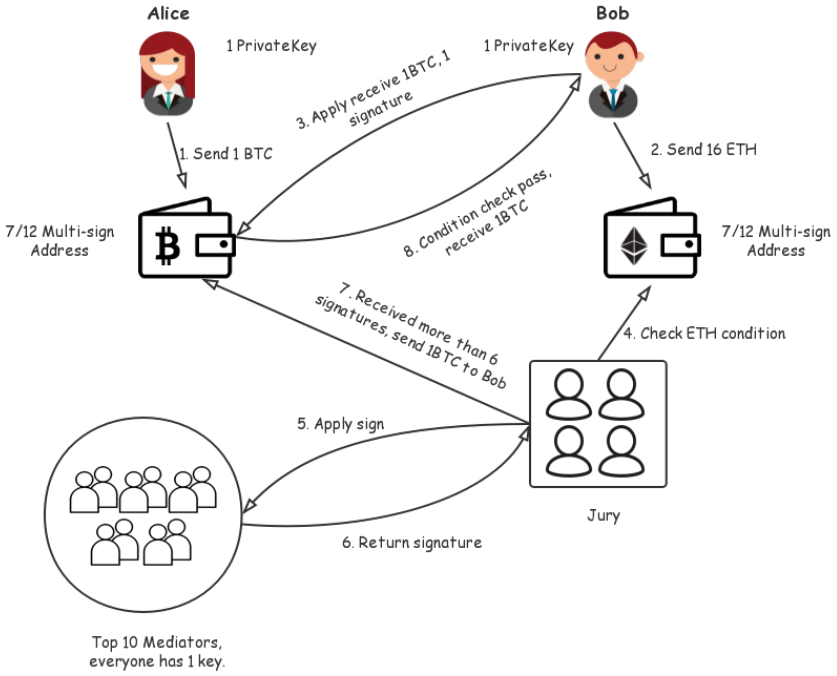

Figure 3.26: The normal process of executing the exchange contract (with Mediator endorsement) [12]

The process of Mediator endorsement during the preparation stage differs from the previous one is, the signed contract will inquire the current votes of the Mediator and select the first 10 nodes to form the 7/12 multi-signature address. And since the limited number of juror in the Jury endorsement, Mediator mode guarantees the persistence of a multi-signature wallet, so there is additional step for the jury that has already reached the consensus to send the execution results to Mediators to verify, then the jury leader can broadcast the result once received the majority of Mediators' signature.

[12]Image courtesy of PalletOne yellow paper [?]

Among the projects I have studied so far, PalletOne has the most ambitious and powerful team with a purely technical project that does not have much community operation.

3.3 Summary

This Chapter have presented a thorough case study of 14 representative cross-chain projects and classified them into several working schemes. Through the official white paper and other documents, the preceding sections explained the working process and theory of them using diagrams to help the understanding. Including some of my personal comments and thoughts towards them. The total work of comparison will be shown as a table in the Appendix **??**.

Chapter 4

Implementation and analysis

In previous chapter, we mainly identified several subsets of cross-ledger communication schemes and explained the corresponding projects in details. Nevertheless, the clear logic levels does not necessarily ensure a good user experience or functional applications. Thus, to evaluate the performance of different protocols/platforms, As we discussed in *Solutions* **??**,

4.1 HTLC Atomic-swaps implementation based on Bitshares and Ethereum

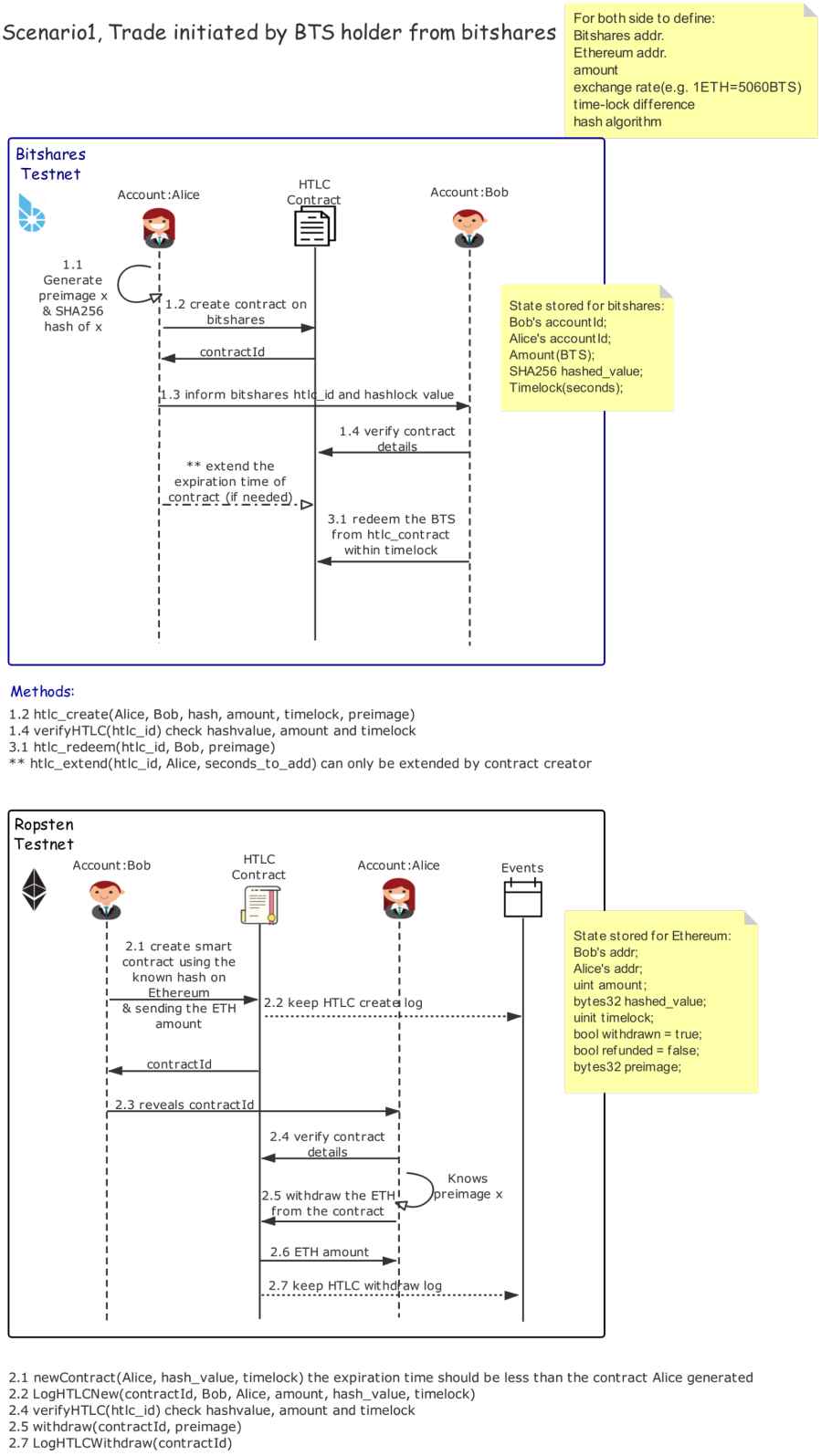

Figure 4.1: BTS->ETH swap diagram

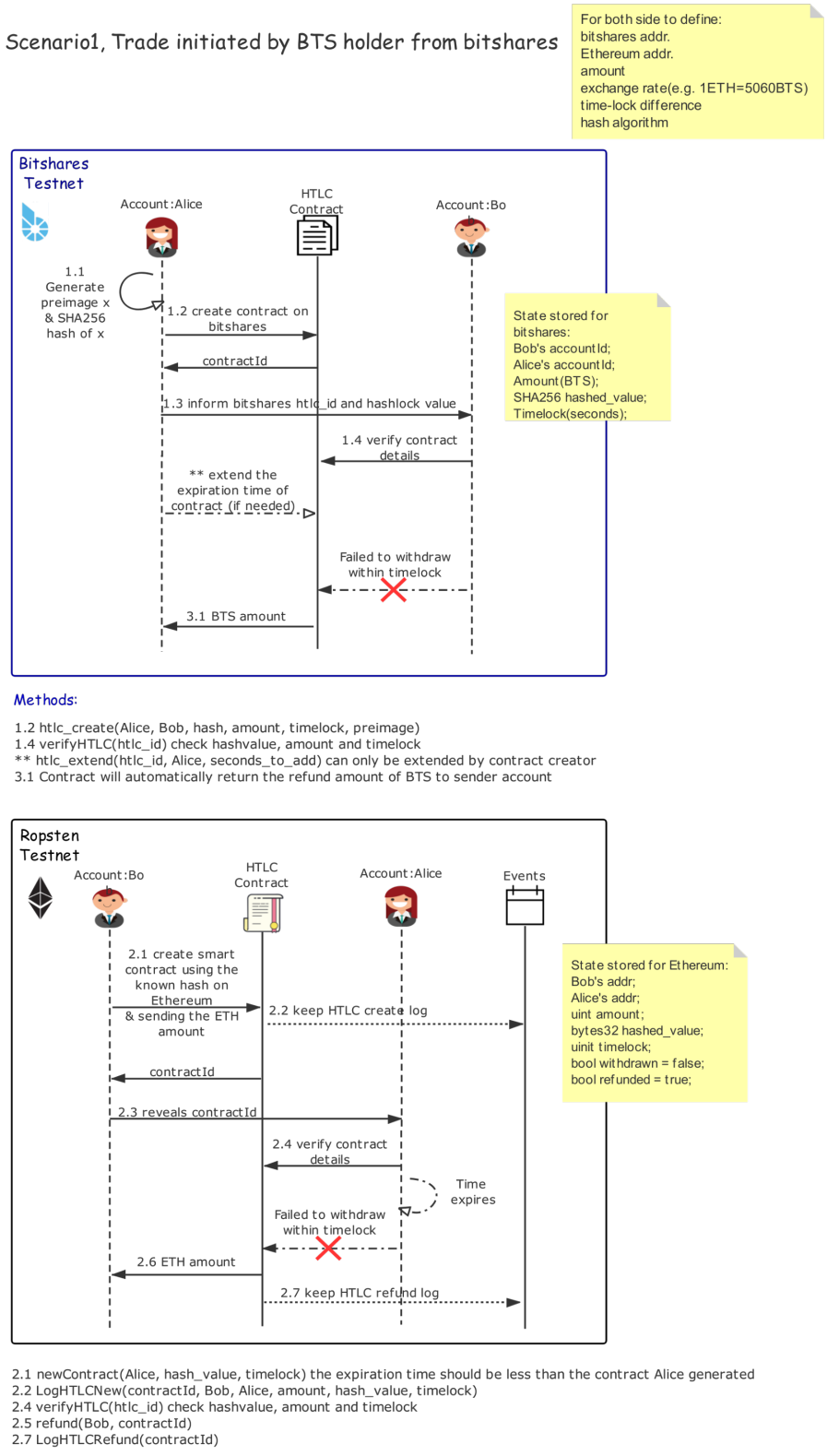

Figure 4.2: BTS->ETH swap diagram

4.2 Interledger switch API study

4.3 Summary

Write here...

Chapter 5

Conclusion and Future Work

5.1

Write your conclusion here...

Appendix A:
Crosswise Comparison Table

Project	Description	Solution	Consensus	Interoperability	Scheme	Application	Compatibility with traditional ledger
Ripple	Ripple aims to build a global payment network based on blockchain, which proposes the Interledger protocol for establishing links with traditional financial institutions.	Notrary	ILP	1-c-1	Protocol	Most situation	Yes
BTC-Relay	In 2016, the Consensys team introduced BTC-Relay, the most classic relay cross-chain solution that enables cross-chain transactions between Ethereum and Bitcoin, as well as realizes Ethereum DApps to support BTC payments.	Relay	PoS	1-c-1	Layer 2	Asset exchange	No
Cosmos	Cosmos is a cross-chain platform project initiated by the Terdermint team in 2017. It supports the modular establishment of Cosmos isomorphism chains and also supports the external heterogeneous chain through Bridge.	Relay	Tendermint	N-c-N	Blockchain	Decentralized exchange	No
Polkadot	Polkadot is a cross-chain platform project initiated by Parity Technologies that links the blockchain networks through relaychains and parachains, and links other chains outside the Polkadot network via BridgeChain.	Relay	PoA	N-c-N	Blockchain	Scalability of decentralized computation	No
Lighting network	Lightning Network is a project running on Bitcoin. There are two main technical points, one is the RSMC (Recoverable Sequence Maturity Contract) and the other is the Hash Time Lock Contract. The former solves the problem of confirmation of the chain transaction, and the latter solves the problem of the payment channel.	Hash Timelock	PoW	1-1	Layer 2	Real-time payment	No
Aion	Aion aims to build a multi-tiered cross-chain platform that supports cross-chain interoperability of heterogeneous chains and connects the blockchain systems by connecting networks and bridges.	Relay	DPoS+PoI	N-c-N	Platform	Information exchange	No
ICON	ICON is committed to building a cross-chain network that connects all types of blockchain systems, enabling DAPPs to be interconnected across all types of blockchains. ICON handles cross-chain transactions primarily through a notary mechanism.	Notary	LFT	N-c-N	Blockchain	Most situation	Yes

Wanchain	It is a cross-chain platform project that provides a cross-chain platform for interoperability of existing heterogeneous chains. It adopts a distributed signature notary mechanism to protect the accuracy of cross-chain transactions.	Notary	WANPos	N-c-N	Platform	Asset exchange	With chainlink
Fusion	Fusion supports multi-platform cross-chain asset transfer, using a distributed signature notary model for cross-chain transaction processing.	Notary	HHCM	N-c-N	Platform	Transactional	No
Chainlink	The ChainLink project was launched in 2014 to address an important issue of blockchain interaction with external data by creating a secure link to link real-world data to blockchain systems.	Notary	N/A	N-N	Protocol	Most situation	Yes
ArcBlock	ArcBlock is a platform dedicated to the development and deployment of decentralized blockchain applications using the Open Chain Access Protocol to realize the cross-chain ability of most applications.	Relay	PoS	N-c-N	Platform	DApps	No
PalletOne	It is a high-performance distributed cross-chain protoco, and has become the IP protocol of the blockchain world. By unifying the interface of each blockchain, it provides a multi-language smart contract running environment, uses randomly elected jurors to manage multi-signed public keys.	Notary	DPoS+Jury	N-N	Protocol	Most situation	No
Quant Overledger	Quant Network creates the Overledger system that operates on top of blockchains, providing the interface for enabling the interoperability between blockchain and traditional network. It provides unlimited possibilities for blockchain data and applications	Relay	N/A	N-N	Platform	DApps	Yes
Plasma	Plasma is a side chain design model of Ethereum. Its main idea is to provide a model that can perform cross-chain transactions, and rely on the Ethereum blockchain to ensure its safety. The MapReduce mode is used to perform parallel computing, which greatly improves the performance of the sidechain.	Sidechain	PoS	1-1	Layer 2	Smart contract framework	No

Elastos	It is a public chain designed with a sidechain scheme. The main chain relies on the Bitcoin POW mechanism to ensure credibility without increasing energy consumption. Sidechain increases the computing power in the form of cluster services to avoid overloading the main chain.	Sidechain	AUXPoW+DPoS	1-N	Platform	DApps	No
Liquid	Liquid is a sidechain of the BTC and is a typical representative of the multi-signature notary mechanism. It is specifically designed to meet the BTC fast transfer needs of exchanges, market makers, and brokers.	Notary	PoW	1-1	Layer 2	Bitcoin-related	No
OneLedger	OneLedger is a cross-chain consensus protocol that enables individuals or businesses using OneLedger to easily implement cross-chain interactions by creating sidechains. All transactions are performed on the sidechain, which greatly improves the efficiency.	Sidechain	3-layer consensus	1-1	Protocol	Decentralized exchange	No
Bytom	A multi-asset swap platform that operates on other chain assets. Developers can create a smaller version of relayer of other chain, and API calls can be made from smart contracts to the relayer of other chain to validate network activity for cross-chain communication.	Relay	PoW	1-c-N	Platform	Asset exchange	No
Aelf	Aelf adopts the architecture of sidechain, which mainly solves the problem of resource isolation. Different applications have different requirements on resources and performance. Therefore, their operation in separate spaces is the optimal asset allocation of system resources.	Sidechain	DPoS	1-1	Layer 2	Sidechain expansion	No
Zipper	Zipper is a decentralized transit network that implements peer-to-peer messaging and transaction settlement across multiple blockchain networks. The Zipper Cross-chain Gateway (CCG) will be responsible for connecting all external blockchains.	Notary	BFT	N-c-N	Platform	Transactional	Yes

Table 1: A comparison table of 20 cross-chain projects

In the table above, the level of interoperability[1] is defined by using the following notation:

- 1-c-1: Two connected blockchains per time with a connector;
- N-c-N: Many connected blockchains per time with connectors;
- 1-1: Two connected blockchain connected per time without connector;
- N-N: Many connected blockchains per time without connectors.

[1] This concept is adopted from Quant Overledger White paper [**?**]

Appendix B: Code Pieces

a. Atomic swaps based on HTLC

Listing 1: Bitshares side

```
/*
 * Here we need to specify the rpc_endpoint_url since I am using the
    local private testnet
 * you can change to your own network ws address
 * Keep the both side private key to perform the signature generate
 * could be update to input parameters.
 */

const rpc_endpoint_url = "ws://127.0.0.1:8090";
var sprivKey = "5Kecd9SoyHEYbSrUnadGzSokptuTWNMKi4M4CgXh7dSNSzLkNLq";
let spKey = PrivateKey.fromWif(sprivKey);
var rprivKey = "5Hwv9FXXrMd4o3FaHFJRLwuMmsihLz29bAGQYon4arK6ZzXCQhB";
let rpKey = PrivateKey.fromWif(rprivKey);
Apis.instance(rpc_endpoint_url, true).init_promise.then(
  res => {
    console.log("Successfully connected to BTS local test network.")
    return ChainStore.init(false);
  });

/*
 * Instance the connection using the api
 * since I use the local private testnet the network_name is not
    defined
 * script adopted from bitsharesjs/examples/createHtlc.js
 * check your environment before use
 */


async function deployHTLC(sender, recipient, Hash, amount, timelock,
    secret) {

  let fromAccount = sender;
  let toAccount = recipient;

```

```
32    let time_lock = parseInt(timelock);
33    let hash = Hash;
34
35    return Promise.all([
36      ChainStore.FetchChain("getAccount", fromAccount),
37      ChainStore.FetchChain("getAccount", toAccount)
38    ]).then(res => {
39
40      let [fromAccount, toAccount] = res;
```

Listing 2: Ethereum Side

```
1
2
3     /*
4      * the function connectAcc() and getAcc() aim to
5      * connect to ropsten testnet using metamask account info
6      * and get the send/receive account as desired
7      * the first account id is 0
8      * the commented line are meant for other users
9      * till now is test only
10     */
11
12  // async function connectAcc(mnemonic, api_key, id) {
13  //   MY_SECRET_MNEMONIC = mnemonic;
14  //   env_api = api_key;
15  //   provider = new HDWalletProvider(MY_SECRET_MNEMONIC, env_api);
16  //   const web3 = new Eth(provider);
17  //   web3.eth.getAccounts().then( function(e) => {
18  //     getAcc(e,id);
19  //   });
20  // }
21
22
23  async function connectAcc(id) {
24
25    let  account = await web3.eth.getAccounts();
26    address = account[id];
27    return address;
28
29  }
30
31  const sha256 = x =>
32    crypto
33      .createHash('sha256')
34      .update(x)
35      .digest()
36  const txLoggedArgs = txReceipt => txReceipt.events.LogHTLCNew.
      returnValues
37  const txContractId = txReceipt => txLoggedArgs(txReceipt).contractId
```

b. Ethereum smart contracts

This contract provides a way to create and keep HTLCs for ETH. Detail protocol:

1. `newContract(receiver, hashlock, timelock)`- sender calls this to create a new HTLC and gets back a 32 byte contract id
2. `withdraw(contractId, preimage)` - once the receiver knows the preimage of the hashlock hash they can claim the ETH with this function
3. `refund()` - after timelock has expired and if the receiver did not withdraw funds the sender / creator of the HTLC can get their ETH back with this function.

Listing 3: Hashed Timelock Contracts (HTLCs) on Ethereum

```
 *
 *  1) newContract(receiver, hashlock, timelock) - a sender calls this to create
 *      a new HTLC and gets back a 32 byte contract id
 *  2) withdraw(contractId, preimage) - once the receiver knows the preimage of
 *      the hashlock hash they can claim the ETH with this function
 *  3) refund() - after timelock has expired and if the receiver did not
 *      withdraw funds the sender / creator of the HTLC can get their ETH
 *      back with this function.
 */
contract HashedTimelock {

    event LogHTLCNew(
        bytes32 indexed contractId,
        address indexed sender,
        address indexed receiver,
        uint amount,
        bytes32 hashlock,
        uint timelock
    );
    event LogHTLCWithdraw(bytes32 indexed contractId);
    event LogHTLCRefund(bytes32 indexed contractId);

    struct LockContract {
        address payable sender;
        address payable receiver;
        uint amount;
        bytes32 hashlock; // sha-2 sha256 hash
        uint timelock; // UNIX timestamp seconds - locked UNTIL this time
        bool withdrawn;
```

Listing 4: Ethereum micro-payment channel – Machinomy

```
    mapping (bytes32 => PaymentChannel) public channels;

    event DidOpen(bytes32 indexed channelId, address indexed sender, address indexed receiver, uint256 value);
    event DidDeposit(bytes32 indexed channelId, uint256 deposit);
    event DidClaim(bytes32 indexed channelId);
    event DidStartSettling(bytes32 indexed channelId);
    event DidSettle(bytes32 indexed channelId);

    /*** ACTIONS AND CONSTRAINTS ***/

    /// @notice Open a new channel between `msg.sender` and `receiver`, and do an initial deposit to the channel.
    /// @param channelId Unique identifier of the channel to be created.
    /// @param receiver Receiver of the funds, counter-party of `msg.sender`.
    /// @param settlingPeriod Number of blocks to wait for receiver to `claim` her funds after the sender starts settling period (see `startSettling`).
    /// After that period is over anyone could call `settle`, and move all the channel funds to the sender.
    function open(bytes32 channelId, address receiver, uint256 settlingPeriod) public payable {
        require(isAbsent(channelId));

        channels[channelId] = PaymentChannel({
            sender: msg.sender,
            receiver: receiver,
            value: msg.value,
            settlingPeriod: settlingPeriod,
            settlingUntil: 0
        });

        DidOpen(channelId, msg.sender, receiver, msg.value);
    }

    /// @notice Ensure `origin` address can deposit money into the channel identified by `channelId`.
    /// @dev Constraint `deposit` call.
```

Appendix B: Code Pieces

a. Atomic swaps based on HTLC

Listing 1: Bitshares side

```
/*
 * Here we need to specify the rpc_endpoint_url since I am using the
    local private testnet
 * you can change to your own network ws address
 * Keep the both side private key to perform the signature generate
 * could be update to input parameters.
 */

const rpc_endpoint_url = "ws://127.0.0.1:8090";
var sprivKey = "5Kecd9SoyHEYbSrUnadGzSokptuTWNMKi4M4CgXh7dSNSzLkNLq";
let spKey = PrivateKey.fromWif(sprivKey);
var rprivKey = "5Hwv9FXXrMd4o3FaHFJRLwuMmsihLz29bAGQYon4arK6ZzXCQhB";
let rpKey = PrivateKey.fromWif(rprivKey);
Apis.instance(rpc_endpoint_url, true).init_promise.then(
  res => {
    console.log("Successfully connected to BTS local test network.")
    return ChainStore.init(false);
  });

/*
 * Instance the connection using the api
 * since I use the local private testnet the network_name is not
    defined
 * script adopted from bitsharesjs/examples/createHtlc.js
 * check your environment before use
 */


async function deployHTLC(sender, recipient, Hash, amount, timelock,
    secret) {

  let fromAccount = sender;
  let toAccount = recipient;

```

```
  let time_lock = parseInt(timelock);
  let hash = Hash;

  return Promise.all([
    ChainStore.FetchChain("getAccount", fromAccount),
    ChainStore.FetchChain("getAccount", toAccount)
  ]).then(res => {

    let [fromAccount, toAccount] = res;
```

Listing 2: Ethereum Side

```


  /*
   * the function connectAcc() and getAcc() aim to
   * connect to ropsten testnet using metamask account info
   * and get the send/receive account as desired
   * the first account id is 0
   * the commented line are meant for other users
   * till now is test only
   */

// async function connectAcc(mnemonic, api_key, id) {
//   MY_SECRET_MNEMONIC = mnemonic;
//   env_api = api_key;
//   provider = new HDWalletProvider(MY_SECRET_MNEMONIC, env_api);
//   const web3 = new Eth(provider);
//   web3.eth.getAccounts().then( function(e) => {
//     getAcc(e,id);
//   });
// }


async function connectAcc(id) {

  let  account = await web3.eth.getAccounts();
  address = account[id];
  return address;

}

const sha256 = x =>
  crypto
    .createHash('sha256')
    .update(x)
    .digest()
const txLoggedArgs = txReceipt => txReceipt.events.LogHTLCNew.
    returnValues
const txContractId = txReceipt => txLoggedArgs(txReceipt).contractId
```

b. Ethereum smart contracts

This contract provides a way to create and keep HTLCs for ETH. Detail protocol:

1. `newContract(receiver, hashlock, timelock)`- sender calls this to create a new HTLC and gets back a 32 byte contract id
2. `withdraw(contractId, preimage)` - once the receiver knows the preimage of the hashlock hash they can claim the ETH with this function
3. `refund()` - after timelock has expired and if the receiver did not withdraw funds the sender / creator of the HTLC can get their ETH back with this function.

Listing 3: Hashed Timelock Contracts (HTLCs) on Ethereum

```
 *
 *  1) newContract(receiver, hashlock, timelock) - a sender calls this to create
 *      a new HTLC and gets back a 32 byte contract id
 *  2) withdraw(contractId, preimage) - once the receiver knows the preimage of
 *      the hashlock hash they can claim the ETH with this function
 *  3) refund() - after timelock has expired and if the receiver did not
 *      withdraw funds the sender / creator of the HTLC can get their ETH
 *      back with this function.
 */
contract HashedTimelock {

    event LogHTLCNew(
        bytes32 indexed contractId,
        address indexed sender,
        address indexed receiver,
        uint amount,
        bytes32 hashlock,
        uint timelock
    );
    event LogHTLCWithdraw(bytes32 indexed contractId);
    event LogHTLCRefund(bytes32 indexed contractId);

    struct LockContract {
        address payable sender;
        address payable receiver;
        uint amount;
        bytes32 hashlock; // sha-2 sha256 hash
        uint timelock; // UNIX timestamp seconds - locked UNTIL this time
        bool withdrawn;
```

Listing 4: Ethereum micro-payment channel – Machinomy

```
    mapping (bytes32 => PaymentChannel) public channels;

    event DidOpen(bytes32 indexed channelId, address indexed sender, address indexed receiver, uint256 value);
    event DidDeposit(bytes32 indexed channelId, uint256 deposit);
    event DidClaim(bytes32 indexed channelId);
    event DidStartSettling(bytes32 indexed channelId);
    event DidSettle(bytes32 indexed channelId);

    /*** ACTIONS AND CONSTRAINTS ***/

    /// @notice Open a new channel between `msg.sender` and `receiver`, and do an initial deposit to the channel.
    /// @param channelId Unique identifier of the channel to be created.
    /// @param receiver Receiver of the funds, counter-party of `msg.sender`.
    /// @param settlingPeriod Number of blocks to wait for receiver to `claim` her funds after the sender starts settling period (see `startSettling`).
    /// After that period is over anyone could call `settle`, and move all the channel funds to the sender.
    function open(bytes32 channelId, address receiver, uint256 settlingPeriod) public payable {
        require(isAbsent(channelId));

        channels[channelId] = PaymentChannel({
            sender: msg.sender,
            receiver: receiver,
            value: msg.value,
            settlingPeriod: settlingPeriod,
            settlingUntil: 0
        });

        DidOpen(channelId, msg.sender, receiver, msg.value);
    }

    /// @notice Ensure `origin` address can deposit money into the channel identified by `channelId`.
    /// @dev Constraint `deposit` call.
```