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
September 10, 2019
Stockholm, Sweden 2019

Abstract

The phenomenon of isolated value in each blockchain system has become a distinct issue of blockchain field. To address this problem, the demand of cross-chain intercommunication came up. In a narrow sense, cross-chain refers to the process of asset interoperability between relatively independent blockchains. In this thesis, we mainly analyze the design principles, technical difficulties, and solutions of cross-chain intercommunication in a narrow sense. With the introduction of distributed ledger technology(DLT), we describe the interaction with other ledgers as the fundamental problem of current blockchain technology.

The implementation form of cross-chain is mainly manifested as asset swap and asset transfer. There are existing applications in the industry so far, and various cross-chain application scenarios can evolve from this. This paper will focus on these two implementations, illustrate their principles, located the realization difficulties, and put forward corresponding possible solutions. Then we elaborated 8 popular cross-chain projects underlying mechanism listed with three main categories. A detailed comparison according to their interoperability level, consensus algorithm and application scenarios of the overall overview of 20 cross-chain projects is presented as a table in the Appendix A.

During the implementation process, we performed a simple atomic swap cross-chain framework based on Hash Time Lock Contract between Bitshares and Ethereum. Compared with the performance of one wallet application test case using Interledger protocol. These two applications are represented the two manifestations of cross-chain realization.

With limited projects to test out, our conclusion was reached after a discussion with relative merits of two approaches practically. Interledger protocol has a better solution from the aspects of the decentralization, scalability, and whether it supports with traditional ledgers.

Keywords: Distributed Ledger Technology, Cross-chain study, Atomic swaps, Interledger protocol, Relay, Sidechains

Sammanfattning

Fenomenet isolerat värde i varje blockchain-system har blivit en distinkt fråga om blockchainfält. För att hantera detta problem kom kravet på interkommunikation mellan kedjor upp. I en smal mening hänvisar tvärkedjan till processen för interoperabilitet mellan tillgångar mellan relativt oberoende blockchains. I denna avhandling analyserar vi huvudsakligen designprinciper, tekniska svårigheter och lösningar för interkommunikation mellan kedjor i en smal bemärkelse. Med introduktionen av distribuerad huvudboksteknologi (DLT) beskriver vi interaktionen med andra bokar som det grundläggande problemet med den nuvarande blockchain-tekniken.

Tvärkedjans implementeringsform manifesteras huvudsakligen som tillgångsbyte och överföring av tillgångar. Det finns hittills befintliga applikationer i branschen och olika tvärkedjiga applikationsscenarier kan utvecklas från detta. Detta dokument kommer att fokusera på dessa två implementeringar, illustrera deras principer, lokalisera förståelsessvårigheterna och lägga fram motsvarande möjliga lösningar. Sedan utarbetade vi åtta populära tvärkedjeprojekt underliggande mekanism listade med tre huvudkategorier. En detaljerad jämförelse beroende på deras driftskompatibilitetsnivå, konsensusalgoritm och tillämpningsscenarier av den övergripande översikten över 20 tvärkedjeprojekt presenteras som en tabell i Appendix A.

Under implementeringsprocessen utförde vi en enkel atomväxlingskedja baserad på Hash Time Lock-kontraktet mellan Bitshares och Ethereum. Jämfört med prestanda för ett testfall i plånbokstillämpning med Interledgerprotokollet. Dessa två applikationer representerar de två manifestationerna av förverkligandet av kedjan.

Med begränsade projekt att testa, nåddes vår slutsats efter en diskussion med relativa fördelar med två metoder praktiskt taget. Interledger-protokollet har en bättre lösning med avseende på decentralisering, skalbarhet och huruvida det stöder traditionella bokar.

Keywords: Distribuerad Huvudbokstеknik, Tvärkedjestudie, Atombyte, Interledger-protokoll, Relä, Sidokedjor

Acknowledgements

This thesis work proceed smoothly with the kind support and help of many individuals whose names may not all be enumerated. I would like to extend my sincere appreciation to all of them.

Foremost, this interesting topic is come up by my industry supervisor **Dr.Henrik Gradin**, who plays an important role of conducting me with the research purpose, leading my direction when the obstacles comes and express the confidence in me.

Also, I am thankful for having professor **Görgy Dán** and professor **Viktoria Fodor** as my academic supervisor and examiner respectively. Professor Görgy gives the great guidance and supervision regarding this research and Viktoria imparts her knowledge and constant help with the report in this study.

My thanks to all relatives, friends and others who share their support through a long time of thesis project. My beloved parents, who is always by my side when I get frustrated and upset, the saying of yours to get together with myself touched me a lot.

Finally, I would express my deep gratitude to enthusiastic developers of Interledger. They patiently answered my questions with the concept and the implementation efficiently and detailedly.

Yuan Fan,
Stockholm,
September 10, 2019

Contents

Contents v

List of Tables vii

List of Figures viii

List of Acronyms ix

Listings x

1 Introduction 1
- 1.1 Motivation . . . 1
- 1.2 Research Questions . . . 2
- 1.3 Research Methodology . . . 2
- 1.4 Research Objectives . . . 3
- 1.5 Delimitations . . . 4
- 1.6 Thesis Organization . . . 4

2 Literature Study 5
- 2.1 Background . . . 5
  - 2.1.1 What is cross-ledger? . . . 5
  - 2.1.2 Evolution of Cross-chain . . . 5
- 2.2 Cross-chain manifestation . . . 6
- 2.3 Difficulties . . . 8
- 2.4 Summary . . . 9

3 Project Classification 11
- 3.1 Solutions . . . 11
  - 3.1.1 Ensure the atomic of transactions . . . 11
  - 3.1.2 Complete the transaction confirmation . . . 14
  - 3.1.3 Realize multiple chains interoperability . . . 18
- 3.2 Project study . . . 20
  - 3.2.1 Lightning Network . . . 20

3.2.2 Notary Scheme . . . 21
3.2.3 Relay Scheme . . . 28
3.2.4 Sidechain Scheme . . . 32
3.3 Summary . . . 36

4 Implementation and analysis **37**
4.1 Environment Setup . . . 37
4.1.1 Blockchains . . . 38
4.2 Smart contract . . . 39
4.3 HTLC Atomic-swaps implementation . . . 40
4.4 Interledger switch wallet . . . 44
4.4.1 STREAM payment . . . 44
4.4.2 Work flow . . . 45
4.5 Summary . . . 49

5 Conclusion and Future Work **51**
5.1 Conclusion . . . 51
5.2 Future Work . . . 52

Bibliography **53**

Appendix A **57**

Appendix B **65**
I. Atomic swaps based on HTLC . . . 65
II. Ethereum smart contracts . . . 67

List of Tables

4.1 Feature summary of chosen blockchains 39

1 Summary of 20 Cross-chain Projects 58

List of Figures

1.1 Research components of cross-chain intercommunication 3

2.1 Cross-chain development tree 6
2.2 Inter-chain exchange diagram 7
2.3 Inter-chain transfer diagram 8

3.1 Atomic swaps diagram 12
3.2 Hash Time-lock Contract diagram 14
3.3 Centralized Notary Scheme diagram 15
3.4 Multi-sig Notary Scheme diagram 16
3.5 Distributed signature Notary Scheme diagram 16
3.6 Direct interconnection network architecture diagram 18
3.7 Cross-chain platform network architecture diagram 19
3.8 ILP example process 22
3.9 Data transfer process from Ethereum to Wanchain[1] 23
3.10 Data transfer process from Wanchain to Ethereum[2] 24
3.11 The normal process of executing the exchange contract (with Jury endorsement) 26
3.12 The normal process of executing the exchange contract (with Mediator endorsement) 27
3.13 BTC-Relay cross-chain process diagram 28
3.14 Cosmos network architecture[1] 29
3.15 IBC sequence diagram 30
3.16 IBC sequence diagram, timeout 31
3.17 2-way pegged sidechain diagram[1] 33
3.18 Drivechain working diagram 34
3.19 OneLedger sidechain architecture 35

4.1 BTS->ETH swap diagram 41
4.2 CLI application outputs 43
4.3 Switch wallet interface 44
4.4 Exchange Sequence diagram 46
4.5 Sequence diagram of payment channel 48

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
ETF	Exchange Traded Fund
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
RPCA	Ripple Protocol consensus algorithm
FBA	Federated Byzantine Agreement
cli	command line interface
SHA-256	Secure Hash Algorithm 256

Listings

1 Bitshares deploy the HTLC 65
2 Ethereum refund after expiration 66
3 Create hash timelock contract example 67
4 Ethereum micro-payment channel structure – Machinomy 69

Chapter 1

Introduction

Chapter 1 gives a general introduction to the thesis contributions which including Section 1.1 describes Distributed Ledger Technology and points out the main issue this area is facing now. Section 1.2 listed out the main problem this thesis trying to address. After discussing the research methods, objectives and delimitations in Section 1.3, 1.4, and 1.5. The outline structure of thesis is presented in Section 1.6.

1.1 Motivation

Initially, the ledger means 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. The distributed ledger is a significant leap after the digitization of ledger-based technology. A distributed ledger is a database that is shared, replicated, and synchronized between network members [1]. 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.

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 challenging 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. Moreover, different from the Internet to achieve the circulation of information, cross-ledger could realize the distribution of value.

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 of finding out the differences between them.

1.2 Research Questions

There is one technical issue that affects the blockchain developers, that is the inter-communications. 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 independent value system. If we can break through the interoperability among different ledgers and let the value circulate on the broader 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 to achieve a real win-win situation.

For the existing blockchain system, it is very time-consuming and laborious to connect them individually. As the development of computer network, we need a set of across chain standards for blockchain interoperability, but this is not an easy thing to do. It should be an industry-driven standard for the widely applied. When the majority of projects follow a common, easy-to-use protocol, this standard is genuinely established. Having a universal standard of cross-chain communication pattern would represent an important milestone for the industry. The rapid flow of information will inevitably drive the improvement of efficiency and become the internal driving force for the development of the blockchain industry.

1.3 Research Methodology

This project mainly makes uses of a combination of qualitative and quantitative research strategy.

For the beginning of this research, a literature study and review was taken first to gain a deep understanding of cross-ledger history and realization of communications through published papers. Then the critical 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.

To gain a proper perspective of the practical usage and performance of purposed solutions, it is necessary to actually implement at least one of them. Two different test cases were introduced and analyzed during the implementation. Involving theoretical comparison between a framework design and integration with the HTLC atomic swaps, and one asset transfer test scenario working mechanism analysis.

1.4 Research Objectives

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.

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 fundamental problems the blockchains now facing, according to those difficulties, I will give several examples through the case study in the following chapters. The analysis and 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 elements of this study as shown in Figure 1.1

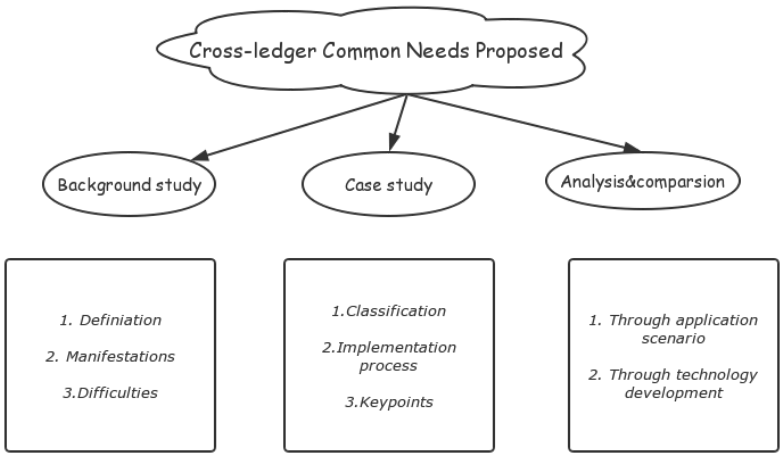


Figure 1.1: Research components of cross-chain intercommunication

1.5 Delimitations

Information provided in this thesis work is the result of multiple new kinds of research, some findings and theoretical background are based on publicly available resources of varying quality, involves the posts from the discussion forum. Addressing the cross-chain problem and located in one specific field could lead to an unbalanced view, since cross-chain may evolve into a financial aspect that I do not comprehend. This thesis conclusion does not provide any suggestion for investment.

There are tons of nuance, variations in cross-chain system design and assumptions. Due to the time limit, here we focus on the specific cross-chain transactions performance to analyze. Aside from the projects that are not landing for use, many projects outlined are proposing solutions that in requirement of multiple different blockchain systems, so we are not able to test them out accordingly.

1.6 Thesis Organization

This thesis is organized into five chapters, as follows:

- Chapter 1 concentrates on the research value and market meaning of this thesis, then briefly introduces the concept of cross-chain communication.
- Chapter 2 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 3 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 4 put efforts in analyzing the market demand situation, and discuss the applications that could be adopted, compare the technology development of some representative projects.
- Chapter 5 summarizes the findings during the research and suggests several ideas for related future work.
- Appendix A contains a comparison table that evaluating 20 cross-chain projects from several valuable aspects.
- Appendix B lists implementation code pieces with essential functions explained and the utilization of smart contracts, for further test use.

Chapter 2

Literature Study

This chapter gives a concept background towards cross-chain technology in Section 2.1, characterizes the specific manifestation of cross-chain projects in 2.2, and in Section 2.3 addresses the cross-chain realization difficulties into 5 parts.

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 causes 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 [2]. While *Vitalik Buterin* mentioned in **Chain Interoperability** [3] that interoperability in the blockchain area mainly describes the ability of assets exchange, information intercommunication, and atomic transactions between various blockchains. The blockchain interoperability can be achieved by introducing the third party without modifying the original chains.

2.1.2 Evolution of Cross-chain

As shown in Figure 2.1, it has not been a short period for Ripple to release the **Interledger Protocol**(ILP) [4] 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 [5] adopted a *Hashed Timelock* mechanism to implement a fast transaction channel off the Bitcoin main chain. In 2016, the BTC-Relay [6] 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** [3] by Vitalik Buterin made a comprehensive and in-depth analysis of blockchain interoperability issues. In 2017, Polkadot [7] and Cosmos [8] first proposed a solution for building a cross-chain network infrastructure platform. Now the Cosmos hub blockchain has launched in March 2019.

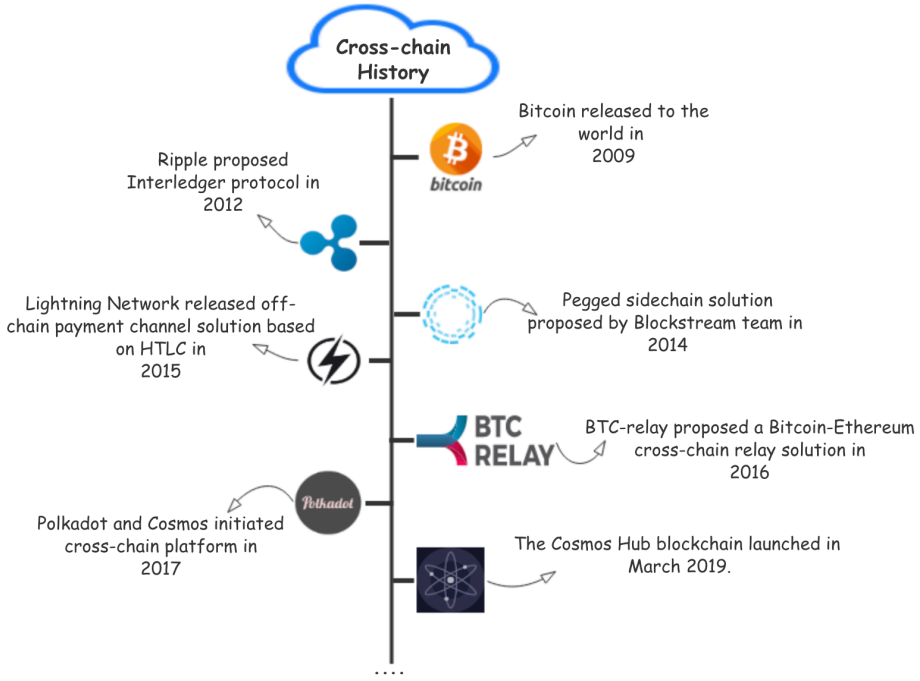


Figure 2.1: Cross-chain development tree

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 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.

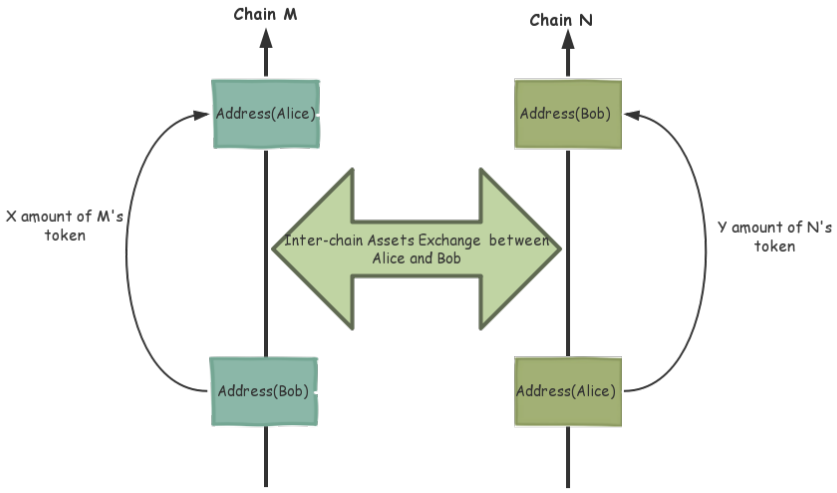


Figure 2.2: Inter-chain exchange diagram

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 assets number of 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 2.3, 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 for future uses. Thus, the transfer of this asset succeed.

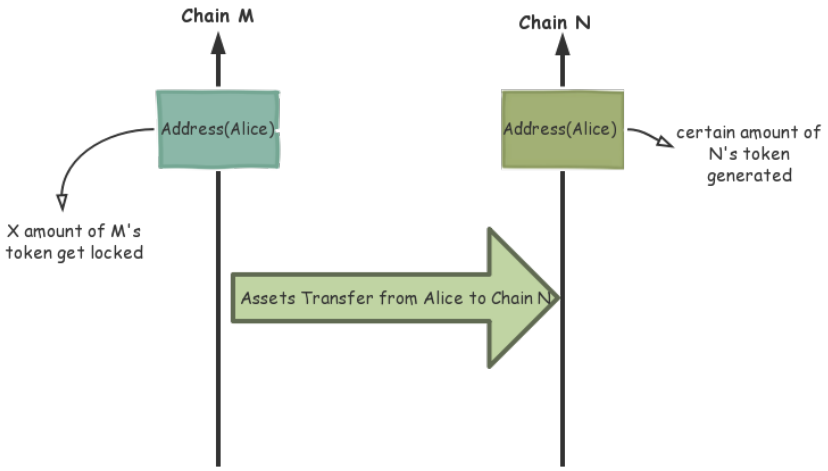


Figure 2.3: Inter-chain transfer diagram

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, and 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 most significant system vulnerabilities in cross-chain transactions, and the security

of both systems will be threatened. The consistency is the fundamental guarantee or 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 of the transaction contains two levels of problems. One is to confirm that the transaction has occurred, wound up and written into the right block. Second is to make sure the system has approved the transaction with enough blocks. In this way, the probability of invalidation of the transaction due to system reconfiguration will be very low. The blockchain system itself is relatively closed, lacking the mechanism to obtain external information actively. Therefore, it is not an easy task to confirm the transaction status of another chain. It can be said that it is one of the core difficulties of cross-chain transactions.

- **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, they will inevitably affect each other. How to ensure the security of their 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

3 project study. Chapter 2 also describes the critical difficulties that cross-chain projects are facing. Some of them will be discussed in the following chapter.

Chapter 3

Project Classification

Chapter 3 presents the corresponding solutions in Section 3.1 based on the difficulties mentioned in Chapter 2. Followed by the case study of eight well-known cross-chain projects in cross-chain industry in Section 3.2. By illustrating the working theory and new terms that purposed, we will be familiar with the scheme categories defined by Section 3.1.

3.1 Solutions

Based on what we discussed about the difficulties in the previous sections, some solutions came up with multiple cross-chain projects. The following paragraphs will illustrate them thoroughly.

3.1.1 Ensure the atomic of transactions

3.1.1.1 Atomic swaps

An atomic cross-chain swap [9] is the basic theoretical framework for multiple parties exchange assets across multiple blockchains. Atomic operations in computer science ensure 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 shown in Figure 3.1:

1. Assume Alice has 1 BTC on Bitcoin while Bob has 20 ETH on Ethereum. Alice wants to change Bob's 20 ETH with 1 BTC. Both Alice and Bob have wallet addresses on both Bitcoin and Ethereum.

2. To start this transaction, Alice needs to randomly generate a key K, which is known only to Alice. Then she initiates a 1 BTC on-chain transaction (transaction ①) to Bob. The transaction can only be finished when Alice obtains the signature of Bob and provides the key K.

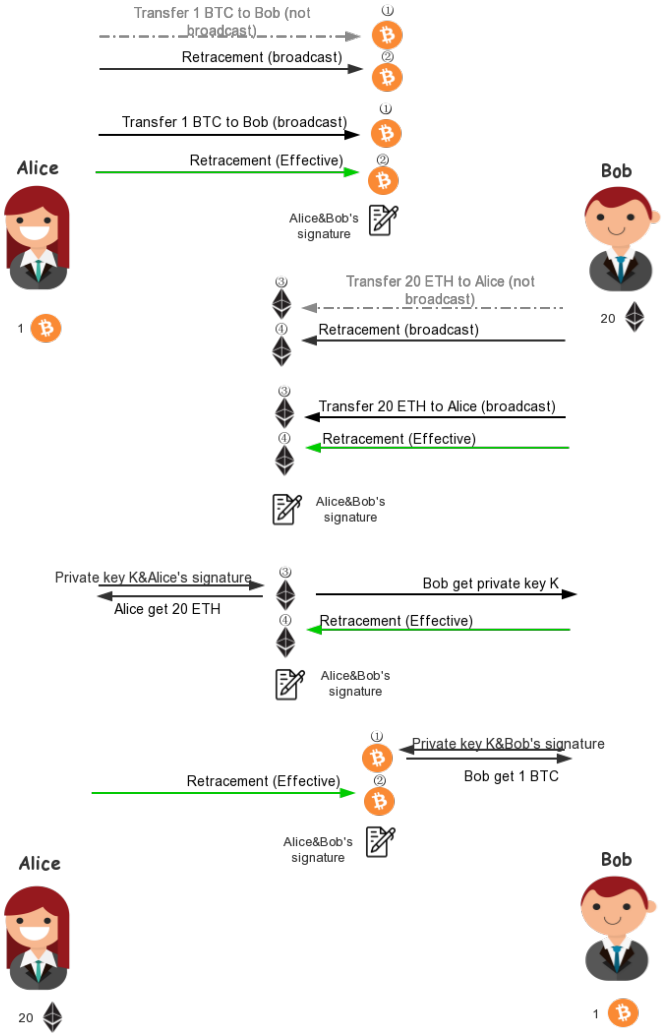


Figure 3.1: Atomic swaps diagram

3. Before broadcast transaction ①, Alice will first broadcast a retracement (transaction ②). If transaction ① does not receive the correct key or signature within 48 hours, the amount paid by that transaction will be returned to Alice. Transaction ② must be signed by Alice and Bob after 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 Ethereum 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 contract together before networks 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. 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 Bitcoin, 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 Bitcoin to Ethereum, but only the assets ownership of both chains. The total assets of Bitcoin and Ethereum 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 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 [10] 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.

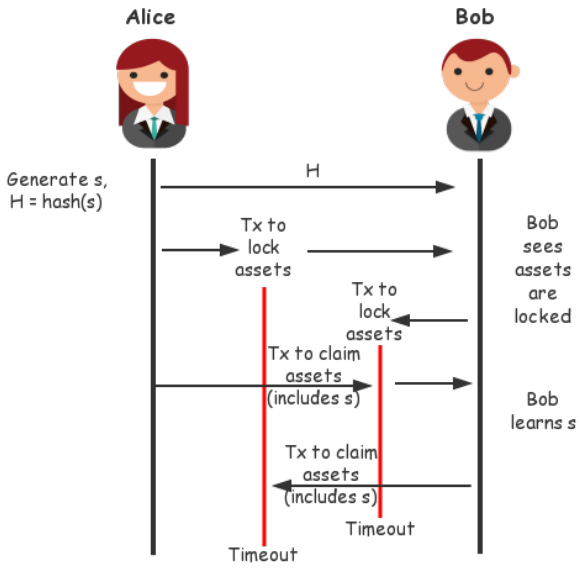


Figure 3.2: Hash Time-lock Contract diagram

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.

3.1.2 Complete the transaction confirmation

As we all know, blockchain systems are relatively independent and closed, there is 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" represents any entity that could interact with two chains, it may be one or a group, the centralized or distributed agency, 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 the requirement can only be one application software deployed on the same node to obtain the others' system data.

After the "middle-man" completes the data collection, how to confirm the transaction, which transaction 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** [3]:
  In the notary scheme, a trusted one or group is used to declare to the chain A that an event has occurred on the chain B, or that the statement is correct. These groups can both automatically or requested to listen and respond to events. There are three different sub-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 model. It purpose a scheme that instead of letting Alice and Bob to perform trade directly, the reliability is increased when dealing indirect transactions with third-party institutions with credit endorsements (such as Alipay) both parties trusts. 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.

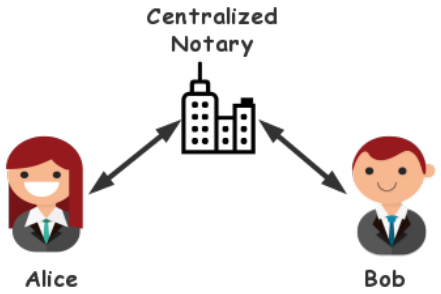


Figure 3.3: Centralized Notary Scheme diagram

To some extent, the use of centralized institutions has replaced technical credit guarantees, from professional 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 critical bottleneck for system stability.

  - *Multi-sig Notary schemes*
    The multi-signature notary mechanism is accomplished by multiple no-

taries 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 private key, and cross-chain transactions can only be confirmed when a certain number or proportions of notary signatures are reached.

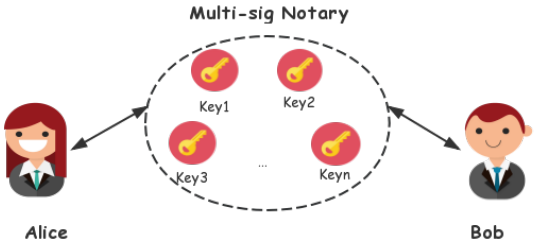


Figure 3.4: Multi-sig Notary Scheme diagram

This method is more secure than the single-signature mode, and a few notaries who are attacked or do evil will not affect the regular 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 a 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.

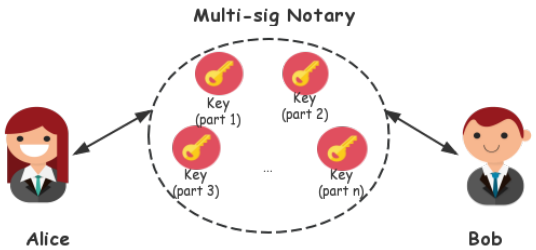


Figure 3.5: Distributed signature Notary Scheme diagram

As Figure 3.5 shows, distributed signature is based on cryptography, the key point is that for cross-chain transactions, the system generates one and only one key. 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 processed as cipher-texts, making it impossible to extract the complete key even all the participants put the pieces together. Thus, the security of the key is fully guaranteed.

- **Relay [3]**:
  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 data. Self-verification methods are depending on the system structure. For example, BTC-relay [6] based on *Simplified Payment Verification*(SPV), and Cosmos [8] rely on verify the number of nodes' signature.

  Vitalik mentioned relay in his Chain Interoperability paper [3], pointing out that chain A and chain B can use the other party's block data for information synchronization and cross-chain function calls. Currently, information synchronization can be done, but there is no mature technical solution for cross-chain function 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.

- **Sidechains**:
  The concept of a sidechain as defined in white paper [11] 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 [3]. "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 blockchain is regarded as the parent chain. In this way, this relationship is based on assets, not the blockchain itself. Pegged sidechain 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 blockchain 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 same systems, which allows the two sides to become a whole part without losing the relative independence of the ledgers.

3.1.3 Realize multiple chains interoperability

Unlike the evolution of computer network, it is still in the "single-machine" era for the emerging technology of blockchain, and the interactions demand between 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 and 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 existing different blockchain application systems in the upper layer, and then the underlying cross-chain mechanism is developed.
Usually these systems are heterogeneous 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

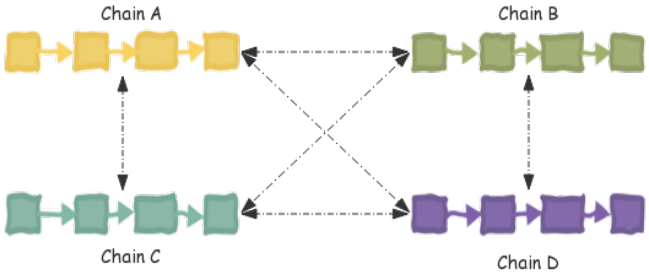


Figure 3.6: Direct interconnection network architecture diagram

This method is the most time-consuming and laborious without the support of the unified underlying protocol. It is necessary to establish six paths between the four chains to realize the interconnection between them as shown in Figure 3.6. Also, 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

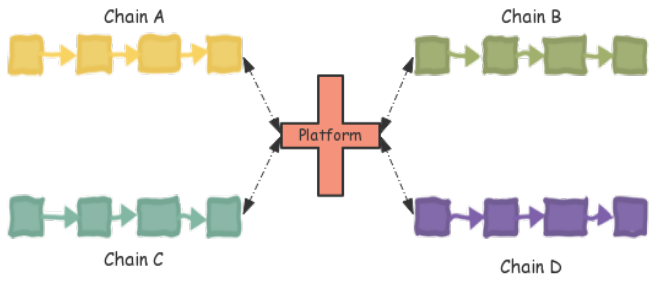


Figure 3.7: Cross-chain platform network architecture diagram

Figure 3.7 shows that once the cross-chain platform established, blockchains can indirectly interconnect with each other. Thus, only four 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. 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. If it is to interoperate with other chains outside the system, it is necessary to develop a separate middleware to communicate.

Besides, 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.

3.2 Project study

Given the solutions discussed above, we are ready to investigate the popular projects that adopts these solutions.

3.2.1 Lightning Network

In general, we could not say lighting network realize the cross-chain function, though it provided a typical application towards atomic swaps and HTLC. The design idea of the lightning network is straightforward. It put a large number of high-frequency small-value transactions off-chain to expand the transaction processing capability of the blockchain.
Lightning network [5] is a fast and scalable Bitcoin transaction project, and 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. This trading pool is a "micro-payment channel". When a transaction occurs between two parties, the proportion share 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 version is invalid. The entire process is completed 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 3.2.

   This diagram 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 of H. Before the arrival of time T, if Bob can learn an appropriate s (secret) where it's 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 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 with, through the intermediary can also achieve mutual transfer.

3.2.2 Notary Scheme

3.2.2.1 Ripple Interledger protocol

Earlier the discussion is focused on cross-chain, this project is more focused on cross-ledger, which means that the agreement supports not only decentralized blockchains but also supports various centralized ledgers. This project provides broader support for cross-chain applications. Interledger protocol has evolves many version right now, here we introduces the version explained by white paper [4].

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) [4] is compatible with any online ledger systems. Specifically, the ILP will establish a two-way 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 3.8Bob, and one selected market maker should have their 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.
- The Interledger module will pack this information 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.

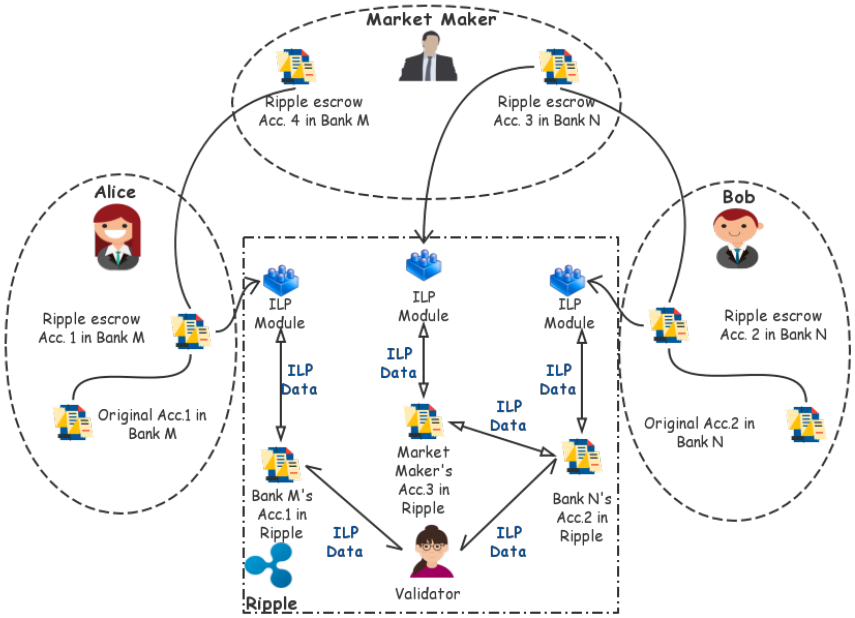


Figure 3.8: ILP example process

3.2.2.2 Wanchain

Wanchain [12] is a cross-chain platform project initiated in 2016. It is a heterogeneous cross-chain framework that implements cross-chain communications based primarily on distributed notary scheme. This model mainly uses cryptography "Secure Multi-Party Computation" and "Threshold Key Sharing Scheme" to achieve 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. Wanchain 3.0 now launches bridges from Bitcoin to the Ethereum network. The transaction reliability verification is completed by multiple **Storeman** nodes of Wanchain. The following figures 3.9 and 3.10 are shown the transfer process between Wanchain and Ethereum.

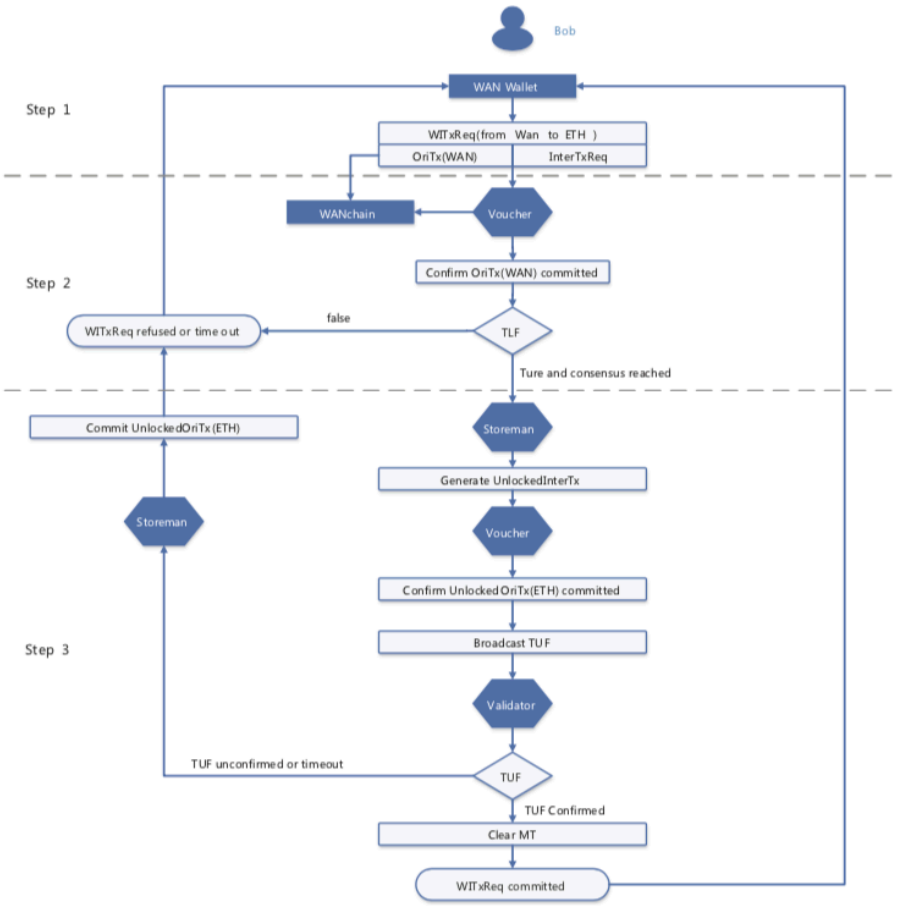


Figure 3.9: Data transfer process from Ethereum to Wanchain[1]

[1]Image courtesy of Wanchain white paper [12]

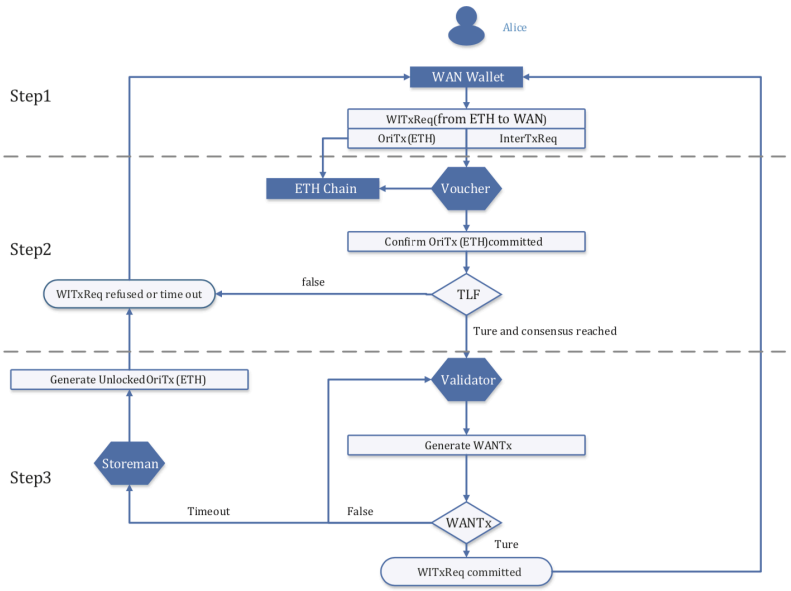


Figure 3.10: Data transfer process from Wanchain to Ethereum[2]

- The token of the user in the original chain will be sent to Wanchain in the locked account, and Hash Time Lock locks the transaction;
- 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 exchange account on Wanchain, and locked;
- After the user's wallet detects the transaction locked by the cross-chain contract, it releases the secret to the cross-chain contract;
- Storeman obtains the control of the original chain token through the secret, thus achieving confirmation of the original chain transaction.
- If the user does not release the secret within the scope of the hash time lock, the hash time lock expires
- The transaction of the contract is automatically invalidated, and the user regains control of the original token.

[2]Image courtesy of Wanchain white paper [12]

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. 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.

3.2.2.3 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 environment for decentralized applications via abstract interfaces. [13]

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, realizing 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. Top of those, the core problem that PalletOne solves is the problem of cross-chain interoperability. The most creative feature in PalletOne is the consensus mechanism, it is complicated and divide the network 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. Most of the work in PalletOne only need to be done by the Jury without calling Mediator, in case that Mediator is causing bottleneck performance.

The following Figures 3.11 and 3.12 provided by the official yellow paper are the vivid examples of the cross-chain process between BTC and ETH with different endorsement.

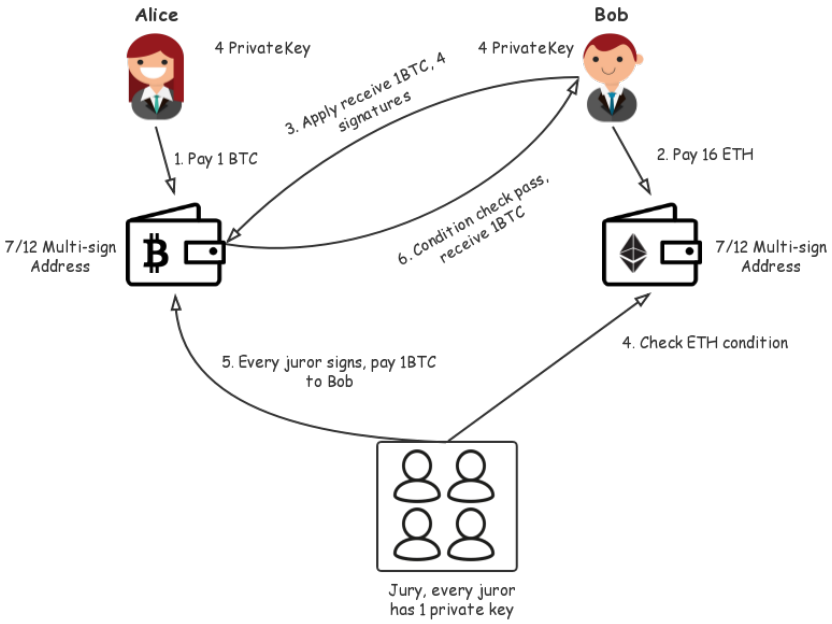


Figure 3.11: The normal process of executing the exchange contract (with Jury endorsement)

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

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.

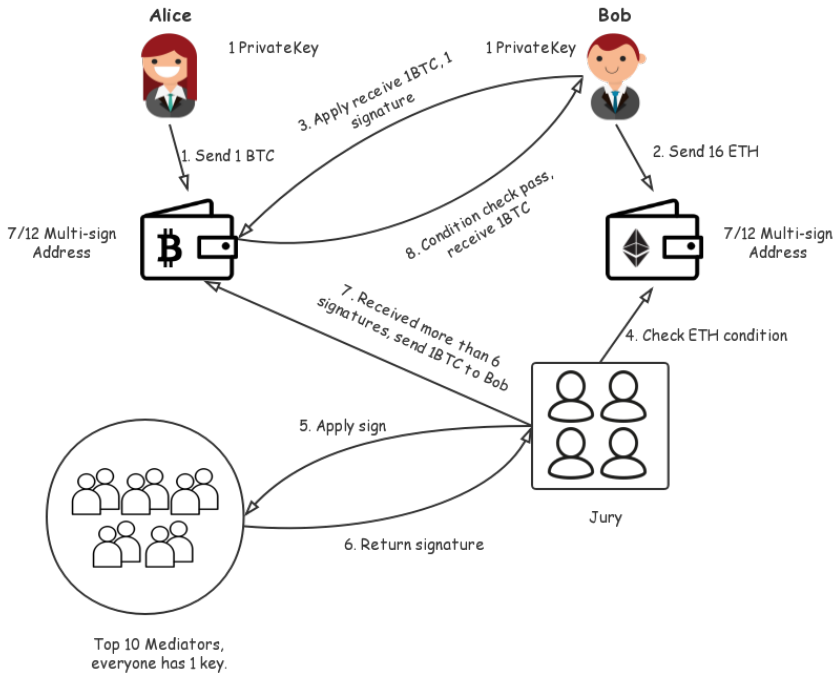


Figure 3.12: The normal process of executing the exchange contract (with Mediator endorsement)

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 ten nodes to form the 7/12 multi-signature address. Also, 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.
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.

3.2.3 Relay Scheme

3.2.3.1 BTC-Relay

In 2016, the BTC-Relay released by the Consensys team [6] is the most classic relay cross-chain solution, enabling cross-chain transactions between Ethereum and Bitcoin, as well as realizing Ethereum's Decentralized 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.

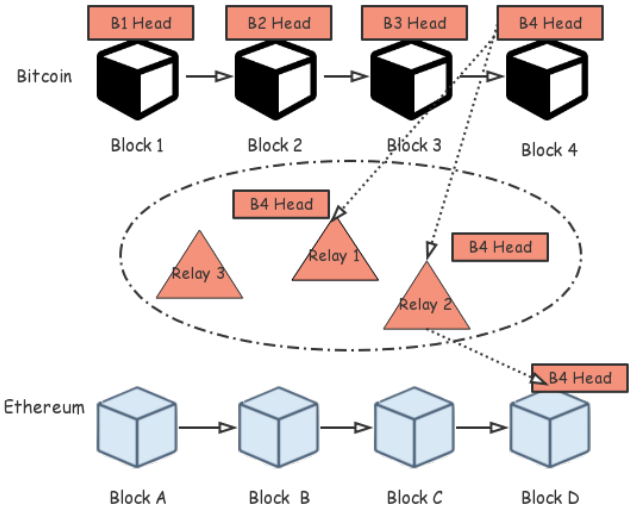


Figure 3.13: BTC-Relay cross-chain process diagram

Like Figure 3.13 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 [8] 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 and can more easily support the flow of assets across the chain. All zones share a set of network protocols, consensus mechanisms, and data storage methods, assembling the new one through the API interface.

The overall architecture of Cosmos network are shown in Figure3.14:

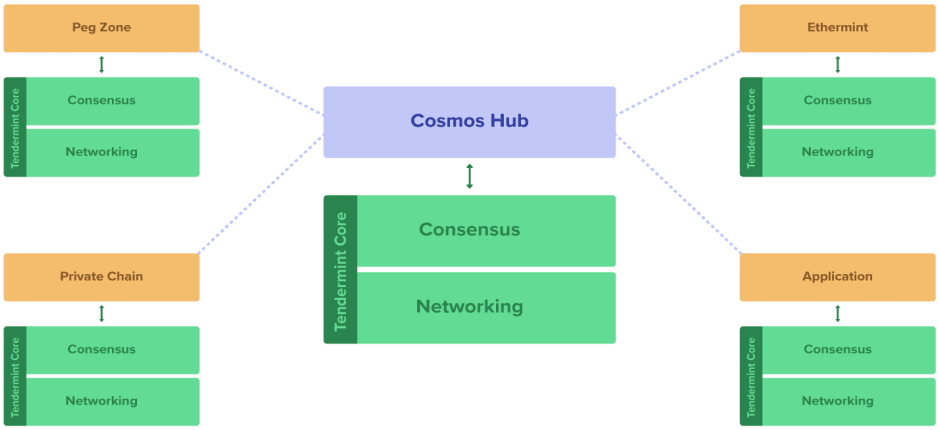


Figure 3.14: Cosmos network architecture[1]

There are many Zones connected to the Hub (both are chains). 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 state of the Hub, but the

[1]Figure taken from: https://golden.com/wiki/Cosmos_network

communication among Zones can only be done indirectly through the Hub. Each cross-chain asset transfer requires a standard successful confirmation from Zones and Hub.

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 with UDP or TCP protocol.

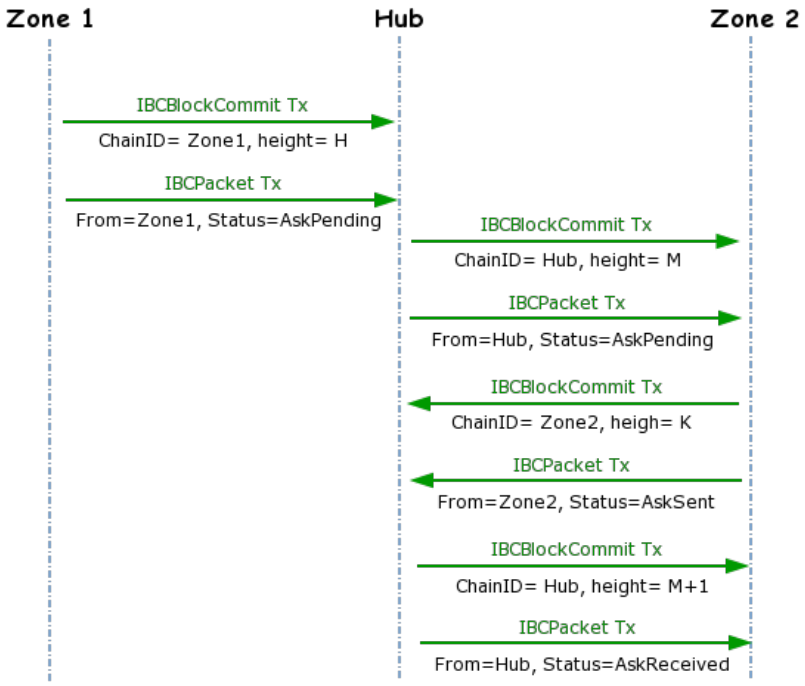


Figure 3.15: IBC sequence diagram

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 and verifies it validity;
3. Zone 1 sends the valid transaction into the message queue for Hub;

4. 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 produces Merkle proof from the original chain and assembles it into a packet, initiates the transaction, passing it to the receiving chain);
5. Hub verifies that Merkle proof is valid, send a message to Zone2 (The process of sending a message to Zone 2 by Hub is the same as previous steps);
6. Zone 2 verifies that Zone 1 has a valid transaction after receiving the Hub message. Send a message to Hub confirming that it can acquire assets from Zone 1;
7. Hub complete one asset transaction by messaging Zone 2 and transfer the assets. 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 figure3.16 below, after waiting for a period of time, the Hub sends a message telling Zone 1 that the current transaction is timeout and the transaction fails.

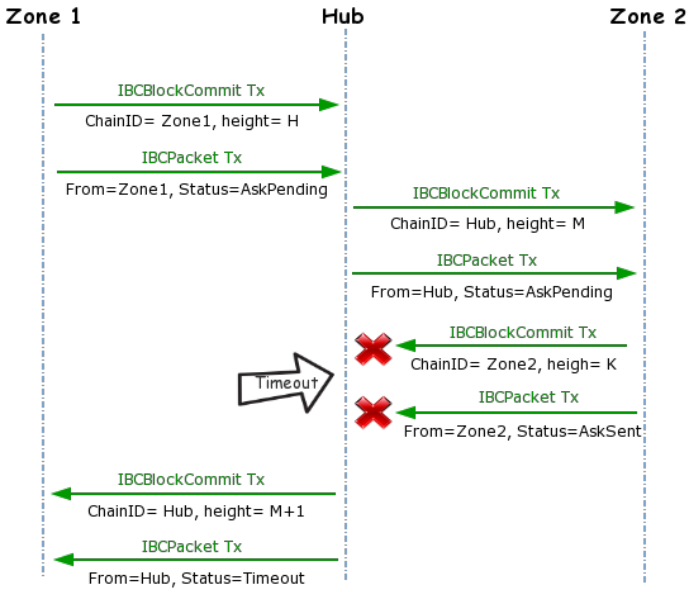


Figure 3.16: IBC sequence diagram, timeout

Cosmos based on **Tendermint** consensus [14], 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.

3.2.4 Sidechain Scheme

3.2.4.1 Ethereum 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 [15]. 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 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 is 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 3.17.

1. When chain A wants to make a transaction 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 $M1$, 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 $M2$ generated on chain 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 be used freely on chain B.
6. The process of a transaction from chain B to chain A is similar to previous steps.

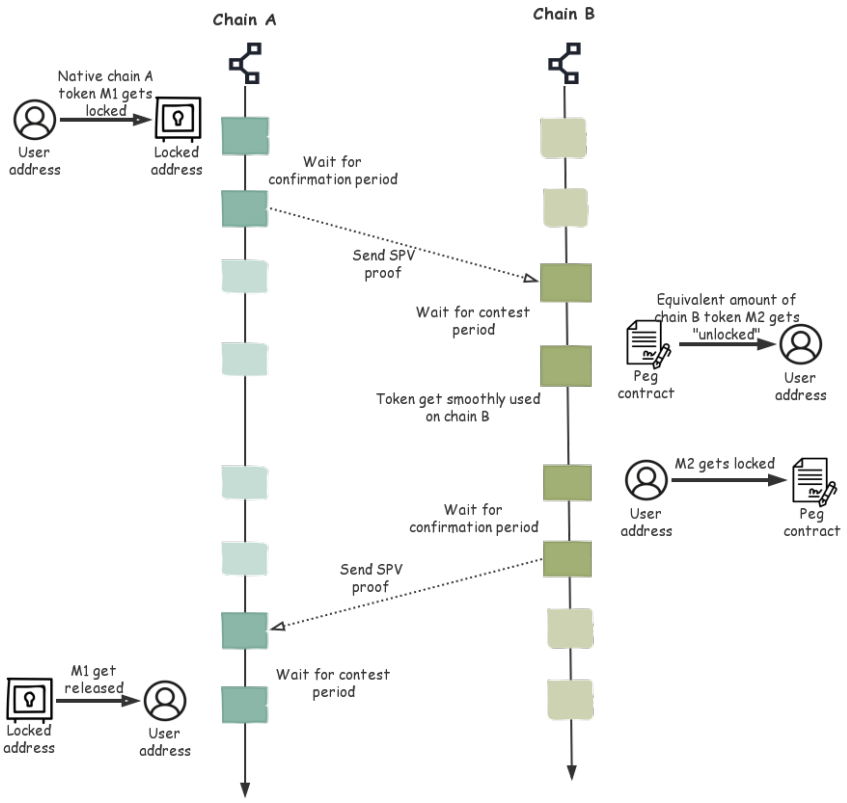


Figure 3.17: 2-way pegged sidechain diagram[1]

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

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 OneLedger

As one cross-chain consensus protocol, OneLedger [16] uses sharding and improved practical Byzantine fault-tolerant consensus. By creating a sidechain, it can easily realize cross-chain interactions between individuals or businesses 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 pattern.

A drivechain [17] 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 3.18 shows:

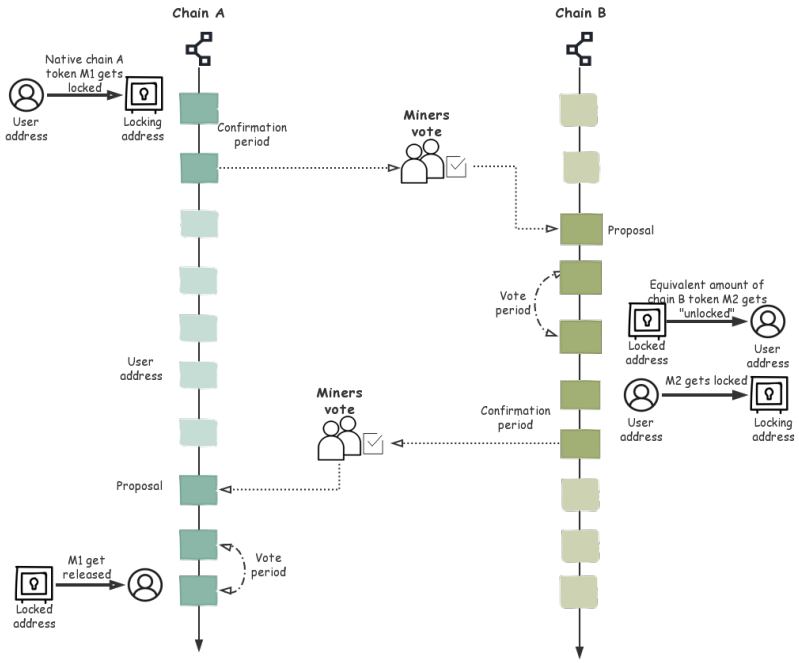


Figure 3.18: Drivechain working diagram

According to OneLedger's white paper, the core of OneLedger is a set of consensus protocols that enable OneLedger to integrate different blockchain products effectively. 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 three layers, I specifically studied the *Public Chain Consensus* which apply on the atomic transfer between blockchains through OneLedger Network on the base layer.

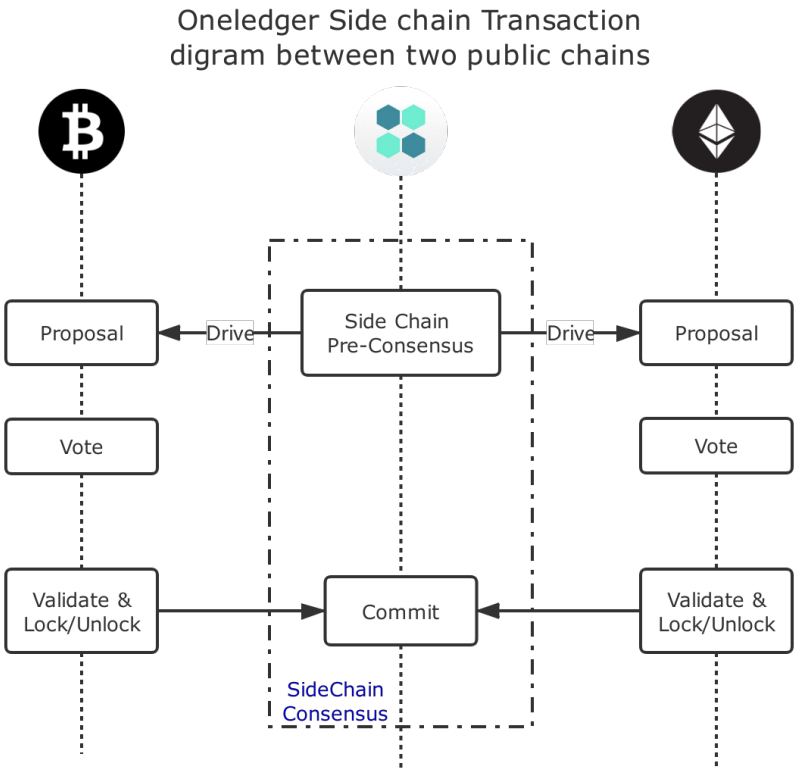


Figure 3.19: OneLedger sidechain architecture

There are two steps in the 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 both public chains accept the proposal, 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.

3.3 Summary

This Chapter has presented a thorough case study of 8 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 comments and thoughts towards them. The total work of comparison will be shown as a table in Appendix A.

Chapter 4

Implementation and analysis

In the previous chapter, we mainly identified several subsets of cross-ledger communication schemes and explained the related projects in details. Nevertheless, clear logic levels do not necessarily ensure a good user experience or functional applications. Thus, to evaluate the performance of different protocols or platforms, it is necessary to design and implement appropriate approaches.

Section 4.1 introduces the implementation background regarding target blockchain description, test-net setup, and related features. Once we get familiar with the environment, section 4.3 describes the implementation details and working diagram for cross-chain swaps, and some use cases are also mentioned. Then we choose Interledger protocol based application to evaluate the performance for cross-chain transfers, with the working details illustrated in section 4.4. Section 4.5 summarizes the results.

4.1 Environment Setup

Evaluating all the investigated solutions at this period is unrealistic, for the limited number of blockchains that supported by various platforms, incomplete blockchain plugins designed by protocols, even some of the projects have not implemented for test use so far. Accordingly, at this early stage of cross-chain techniques, the plan is first to implement a basic functional cross-chain swap program, then choose one developed application for source code study. Followed by an analysis of these two solutions from several chosen aspects, such as the complexity of realizing the transaction, safety issues, and future application scenarios.

4.1.1 Blockchains

In these experiments, we aimed at utilizing different cross-chain solutions on two trade parties, which requires the user account setup on those blockchain test-nets. Out of consideration of ensuring the integrity and feasibility of the implementation, we choose Ethereum as the main object blockchain to perform the test.

A. Ethereum

Ethereum [18] is an open-source well-known blockchain platform that allows anyone to build and use decentralized applications running through the blockchain technology. Similar to the great significance Bitcoin brought, Ethereum is regarded as the representative of the blockchain 2.0 era.

The blockchain 1.0 era is usually referred to the development stage of blockchain application represented by bitcoin between 2009 and 2014, where they mainly focus on solving the decentralization of currency and the form of payment. After 2014, blockchain developers are gradually shifting to address the deficiencies of technical and scalability related to bitcoin. In 2014, Vitalik Buterin released Ethereum white paper [19], which introduced smart contracts into blockchain and opened the application of blockchain outside the currency field, thus opening the era of blockchain 2.0.

Ethereum is a programmable blockchain using the Turing-complete scripting language, that enables developers to create a decentralized application that custom complex actions instead of giving users a set of predefined actions. Those applications can be run on Ethereum Virtual Machine(EVM), the core of Ethereum. Advanced and programming friendly is the reason why we choose Ethereum to perform cross-chain solutions.

B. Bitshares

Bitshares [20] is the first delegated proof of stake blockchain created by BM team in 2014. It is an open-source high-performance distributed trading system that supports valuable objects such as cryptocurrencies, legal tender, and precious metals [21]. Bitshares blockchain produces the corresponding token BTS for usage in bitAssets system and other finical services featured on the distributed ledger. Compared to Ethereum, Bitshares has much lower transaction confirmation time, and intended to be the commercial exchange platform in the future.

Bitshares also have a convenient idea of mapping account names to addresses, so you can transfer money directly using the name of the account you are applying for, which is a meaningful detail compared to an elaborate list of code addresses. The developed integration library using javascript and HTLC support facilitate my

implementation of atomic swaps.

C. Ripple XRP ledger

Ripple is the world's first open payment network, a peer-to-peer global payment network based on block connectivity. XRP ledger [22] presented by Ripple serves as an open-source distributed ledger for real-time financial transactions. As the home of base currency XRP in the Ripple network, it has introduced a unique consensus mechanism RPCA. RPCA [23] combines the problems of Byzantine generals to get rid of the restriction of consensus through mining and validates and confirm transactions in a short time by voting special nodes.

As another product created by Ripple, XRP ledger is much easier to integrate with Interledger protocol, together will they enable money network interoperability.

Table 4.1: Feature summary of chosen blockchains

Blockchain	Ethereum	Bitshares	Ripple
Focus	Smart Contract	DEX	Payment Network
Consensus Mechanism	PoW	DPoS	RPCA & FBA
Token	ETH	BTS	XRP
Transaction Throughput (tps)	15-25	3300	1500

Table 4.1 above summarized several important features those three blockchain holds, the data of Ripple and Ethereum throughputs are calculated from the highest transaction number provided official website transaction charts[1]. And the performance of bitshares is measured by user from test-net[2].

4.2 Smart contract

Smart contract [24] is an essential concept and widely used in Ethereum. It is described as a computer protocol to facilitate, verify, or enforce the behavior of contracts digitally. A smart contract allows trusted transactions confirmed without a third-party, and these transactions are untraceable and irreversible.

In Ethereum, smart contracts are written in Solidity language. Solidity [25]is one contract-oriented, Turing-complete programming language that mainly consist of four elements: *Contract, Event, Variable, and Function*. **Event** is one particular

[1]Ripple:https://xrpcharts.ripple.com/#/metrics Ethereum:https://etherscan.io/chart
[2]Bitshares: https://bitsharestalk.org/index.php?topic=23829.0

part compared to other programming languages. It substituted the Pub/Sub services in Ethereum.

You can set up a smart contract account with functional services by compiling the contract codes and deploying it on the Ethereum network. Once the contract is deployed successfully, the user would get a contract address. Users could create a transaction containing a payment of ETH to the contract along with other information it may need for the interaction. Similarly, functions and variables could be called afterward. The transaction block would be confirmed to the whole network once the mining computers validate and sign.

By utilizing smart contracts in this implementation, we could simulate cross-chain transactions with Ethereum accounts. Specifically, building the HTLC contract to realize atomic swaps. In the Interledger application, the contract works as the payment channel. Some example codes explanation could be seen in Appendix B II. Ethereum smart contracts.

4.3 HTLC Atomic-swaps implementation

As we discussed in section 3.1, the most fundamental answer of cross-chain realization is the guarantee of atomicity in transactions. This implementation intended to solve the atomic problem by creating a cross-chain transaction between two parties based on HTLC. The sequence diagram of the cross-chain transaction process is shown in Figure 4.1.

This implementation is performed under the Bitshares private test-net and Ethereum Ropsten test-net environment. BTS holder can only initiate the cross-chain trade in this scenario. This incompleteness is traced back to the HTLC mechanism Bitshares provided, the preimage and hash value are both required when creating the contract. Thus we had to generate unidirectionally.

To ensure the atomicity of the transaction, we recommended and limited the preimage length to be no shorter than 32. In fact, due to the `Bytes32` format is using in solidity contract creation, specifically, the `abi.encodePacked(...)` algorithm[1]. This packed algorithm would not treat string shorter than 32 bytes with either zero-padded or sign extended. On the other hand, this limitation brings out stronger hashlock security.

Some important functions are explained in Appendix B and the whole project can be seen here[2].

[1] see here: https://solidity.readthedocs.io/en/v0.4.24/abi-spec.html#abi-packed-mode
[2] github repo:https://github.com/Fy45/BTS-ETH-atomic_swaps

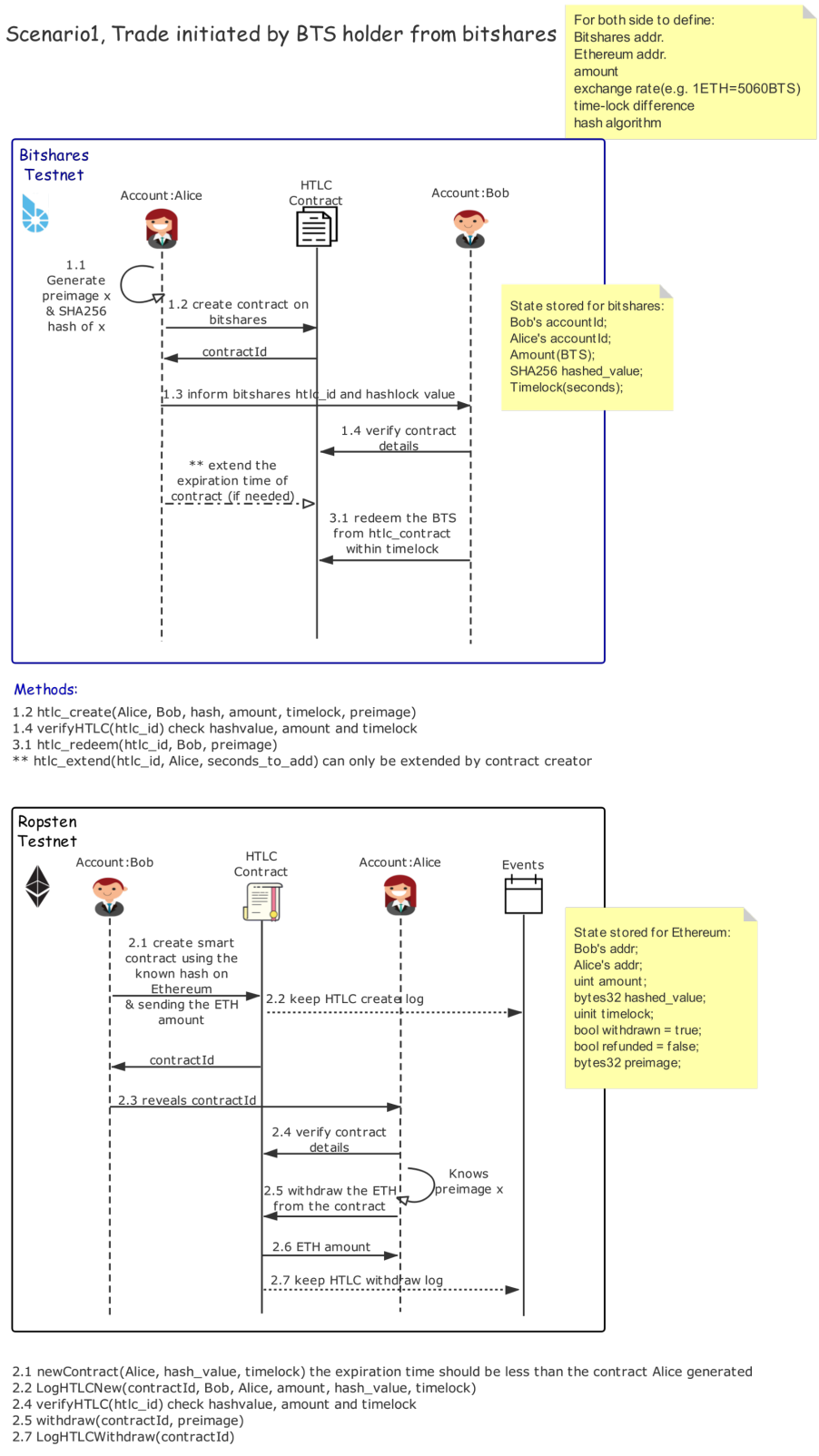


Figure 4.1: BTS->ETH swap diagram

The function of create HTLC, verify the contract details, withdraw the money and refund are realized in this cli application. The following figures 4.2a and 4.2b show how a complete cross-chain transaction interacts.

Before the action has been taken. Both parties should agree on the exchange rate and the expiration time offline. As shown in the figure 4.1.

- When user Alice chooses to send BTS for ETH, she needs to generate a safety password with fixed length herself first. Along with the counterparty account name, amount and timelock to deploy the contract. The secret should be only known to Alice. SHA-256 [26] algorithm is use for calculating Hash value. The transaction is automatically canceled after some period T. Then Alice should inform John from Ethereum blockchain with the hashlock value and the BTS HTLC id.

- John would initiate the contract conditionally on the same hash value he got from Alice, but lock the contract for $T/2$. Thus, it could effectively prevent Alice from doing evil by revealing the secret too late to redeem the BTS. Also, John should notify Alice with the Ethereum contract id.

- If the contract is executed successfully, both party should verify the timelock, hash value, token amount in the contract by calling `verifyHTLC(htlc_id)`. When contract details check out, Alice would reveal the secret to John, which gives her the right to John's ETH and gives John the right to Alice's BTS.

- If the contract failed for some reason. Such as information is unmatched with the pre-set value, Alice would not reveal the preimage to redeem the ETH, and John would not give the contract id to Alice. Alternatively, Alice does not reveal the preimage before the time expires, John could call `refundHTLC(Bob, htlc_id)` to get ETH back shown in figure 4.2c

Due to different working mechanism in Bitshares and Ethereum, the refund performance differs. Unlike HTLC in Bitshares, which automatically returns the unclaimed amount to the contract creator after time expires, Ethereum handles refund by calling the specific function inside the contract. Moreover, the contract in Bitshares can be extended for a longer time by the sender.

```
$ npm test

> BTS-ETH-atomic_swaps@1.0.0 test /Users/fanyuan/KTH-ICT-TCOMM/Thesis/Analysis/BTS-ETH-atomic_swaps
> node src/start

Unknown chain id (this may be a testnet) 8e0c5b1e2e37487ce5b5d40b5acc57d8387992ef2ea82562701871319674dc52
Successfully connected to BTS local test network.
synced and subscribed, chainstore ready
Enter 1 to send BTS get ETH
      2 to receive BTS from ETH
      3 to extend BTS contract time
      4 to resolve ETH HTLC
> 2
Enter your sender account id of ETH wallet(e.g. firstAcc is 0): 0
 Ropsten ETH wallet address  0xe2B1B5d92b64846D815cd34c674973f88DcefF4d
Enter ETH address to receive funds: 0x209f4b189e246Ae171da5a6F1815c91C70CAA23A
Enter the ETH you want to send: 0.1
Enter the hash_value you got from BTS side: 0x0c72c0fb858b5f0e2b86d599ea82cd75b6fd15cd9c79fcd01cc0b94767c34480
 In order to protect your money, please lock the contract less time than BTS does...
Enter the time you want to lock in contract (seconds): 300
 Deploying...
 ETH HashTimelockContract was successfully created!
 Please inform your counterparty with the ETH HTLC id 0xb284bea87a0b349722245bee9695da7bc81aed73ae3d48f28e268ad571cf71d2
Enter your BTS account name: Bob
Enter the BTS HTLC id: 1.16.172

 BTS HTLC:
 From Account id         | 1.2.17
 To Account id           | 1.2.18
 Transaction amount      | 1000 BTS
 Hash value              | 0x0c72c0fb858b5f0e2b86d599ea82cd75b6fd15cd9c79fcd01cc0b94767c34480
 Expiration time         | 2019-08-06T11:48:05

 If details are correct then input yes to redeem your BTS
 Or else please enter exit and talk with your counter party:
> yes
 Waiting for ETH contract to be resolved...
 Resolving BTS HTLC contract...
 BTS HashTimelockContract was successfully redeemed at : Tue Aug 06 2019 13:41:10 GMT+0200 (GMT+02:00)
  Account Balance is 86065.9435 BTS
```

(a) Complete transaction(ETH->BTS)

```
$ npm test

> BTS-ETH-atomic_swaps@1.0.0 test /Users/fanyuan/KTH-ICT-TCOMM/Thesis/Analysis/BTS-ETH-atomic_swaps
> node src/start

Unknown chain id (this may be a testnet) 8e0c5b1e2e37487ce5b5d40b5acc57d8387992ef2ea82562701871319674dc52
Successfully connected to BTS local test network.
synced and subscribed, chainstore ready
Enter 1 to send BTS get ETH
      2 to receive BTS from ETH
      3 to extend BTS contract time
      4 to resolve ETH HTLC
> 1
Enter BTS account name of sender: Alice
Enter BTS account name of recipient: Bob
Enter BTS amount to send: 1000
 To log contract uniformly,
 we highly recommended generating secret should be no shorter than 32!
Enter the preimage value you generate: O^0!&=J?KY5K?qMP^FPFKlcL37o2PbRt
Enter the time you want to lock in contract (seconds): 600
 Deploying...
 BTS HashTimelockContract was successfully created!
 Please inform your counterparty with the hash value: 0x0c72c0fb858b5f0e2b86d599ea82cd75b6fd15cd9c79fcd01cc0b94767c34480
 and BTS HTLC id 1.16.172
Enter your recipient account id of ETH wallet: 1
 Ropsten ETH wallet address: 0x209f4b189e246Ae171da5a6F1815c91C70CAA23A
Enter the ETH HTLC id: 0xb284bea87a0b349722245bee9695da7bc81aed73ae3d48f28e268ad571cf71d2

 ETH HTLC:
 Sender          | 0xe2B1B5d92b64846D815cd34c674973f88DcefF4d
 Reciever        | 0x209f4b189e246Ae171da5a6F1815c91C70CAA23A
 Transfer amount | 100000000000000000 Wei
 Hash value      | 0x0c72c0fb858b5f0e2b86d599ea82cd75b6fd15cd9c79fcd01cc0b94767c34480
 Unlock time     | Tue Aug 06 2019 13:43:40 GMT+0200 (GMT+02:00) (~ 3 mins)
 Enter yes if you want to redeem the agreed amount of ETH from contract:
 0xb284bea87a0b349722245bee9695da7bc81aed73ae3d48f28e268ad571cf71d2
 Or enter exit if you want to quit:
> yes
 Resolving ETH HTLC...
 ETH HashTimelockContract was successfully redeemed!
 Account: 0x209f4b189e246Ae171da5a6F1815c91C70CAA23A has balance of 3.461803228989493912 ETH
```

(b) Complete transaction(BTS->ETH)

```
 BTS HTLC:
 From Account id         | 1.2.17
 To Account id           | 1.2.18
 Transaction amount      | 10 BTS
 Hash value              | 0x1bd88cbf65b2da50c4ae122523ab2af2c5032c7e728ea3c2afd22b1965d9594f
 Expiration time         | 2019-08-05T12:24:10

 If details are correct then input yes to redeem your BTS

 Or else please enter exit and talk with your counter party:
 > yes
 Waiting for ETH contract to be resolved...
 ETH HTLC timed out
 Refunding ETH...
 ETH HashTimelockContract was successfully refunded!
 Account: 0xe2B1B5d92b64846D815cd34c674973f88DcefF4d has balance of 10.745026920831579951 ETH
```

(c) Refund output of ETH

Figure 4.2: CLI application outputs

4.4 Interledger switch wallet

Switch is an Interledger wallet shown in Figure 4.3. It is designed by Kava-labs [27] which can be used on the exchange of four types of cryptocurrencies so far. Switch achieved non-custodial cross-ledger transactions by leveraging the stream payment protocol based on ILPv4 [28]. In this version, ledger-enforced hashlock was removed and change into packetized payment model. The introduction and reasons are illustrated in STREAM payment.

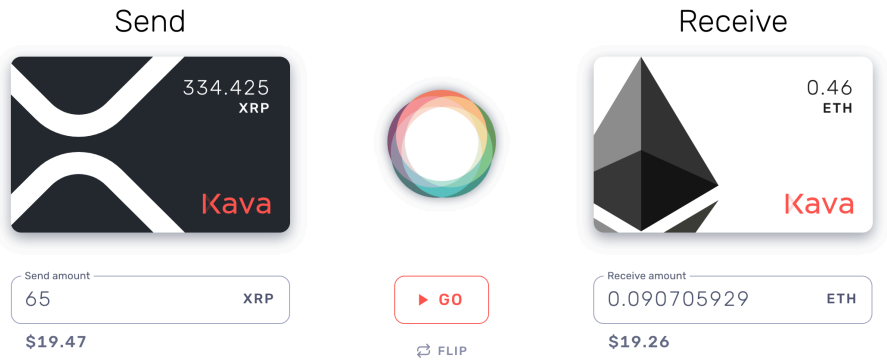


Figure 4.3: Switch wallet interface

4.4.1 STREAM payment

STREAM is an Interledger transport protocol, short for Streaming Transport for the Real-time Exchange of Assets and Messages, is presented by Interledger developer team in 2018 [29]. This protocol ensures the money and data are sending over ILP packets in a reliable way, often regarded as the TCP in the ILP.

Similar to the Internet network handling the different size of files like streaming media, STREAM protocol adopts the idea of sending the data or money in lots of tiny packets during a short of time. It is efficient since the transaction speed in the blockchain network is dependent on the payment bandwidth, STREAM would automatically adjust the money/data size in each ILP packets to avoid network congestion. Moreover, during one connection, new streams can be open and closed anytime to ensure multiple messages exchanged.

Endpoint in the application based on STREAM protocol could open stream payments, send a specific amount of money through them. The stream abstraction will provide a mechanism to fragment this payment into multiple tiny packets. These packets are authenticated and encrypted by a shared secret. One could close the

stream whenever the transaction is complete.

As we discussed in Ripple Interledger protocol, ILP handling the cross-chain payments through a connector/escrow, and the exchange liquidity is also determined by it. So using atomic swaps may cause the problem of floating option [30]. If the exchange rate changes in their favor, they can execute the trade; if the exchange rate goes against them, they can let the timeout expire at no cost, which somehow introduces the counterparty risk. STREAM, on the other hand, set the minimum acceptable amount in each packet, preventing the money claim when the exchange rate is undesired than expected. Build payment channel is also an enhancement to cross-chain interaction since users do not need to wait or pay for the expensive on-ledger transfer every time they had an exchange. Instead, they could call the latest claim to the ledger and get the share amount of money in the channel when they are finished.

4.4.2 Work flow

Switch is one wallet application that implements the function of assets swaps by making a payment to yourself through a connector. As indicated in RFC [31], a user could minimize the counterparty risk by using the payment channel and meager in-flight amounts.
The simple workflow is shown in Figure 4.4. To initiate the asset exchange, user needs to make the deposit to the Interledger wallet from each decentralized ledger first, and start streaming payment, the maximum of M and N is controlled by STREAM protocol, m and n are calculated by the exchange rate connector provided. At the same time, two payment channels are created on each ledger between the kava-lab connector and user account. It should be noticed that the total deposit amount should limit the total exchange amount.

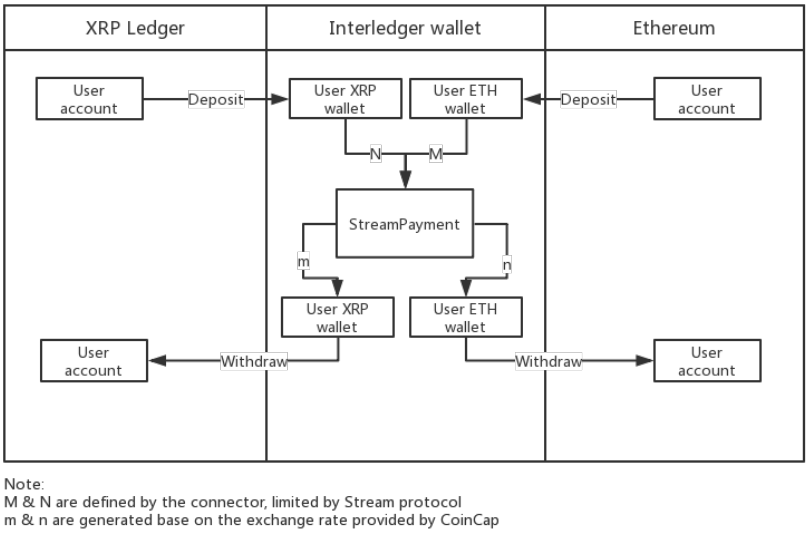


Figure 4.4: Exchange Sequence diagram

Take Ethereum for example, Figure 4.5 illustrates the sequence diagram of how payment channel works.

1. After create the payment, the Ethereum account will trigger a micropayment channel through the Machinomy contract[1] with a payment of X ETH for transaction from user account to online-peer, the online-peer defined by connector will also create a channel with Z ETH to user account, the amount Z is a fixed number also defined by connector. The contract will notify both sides with the channel id for interaction.

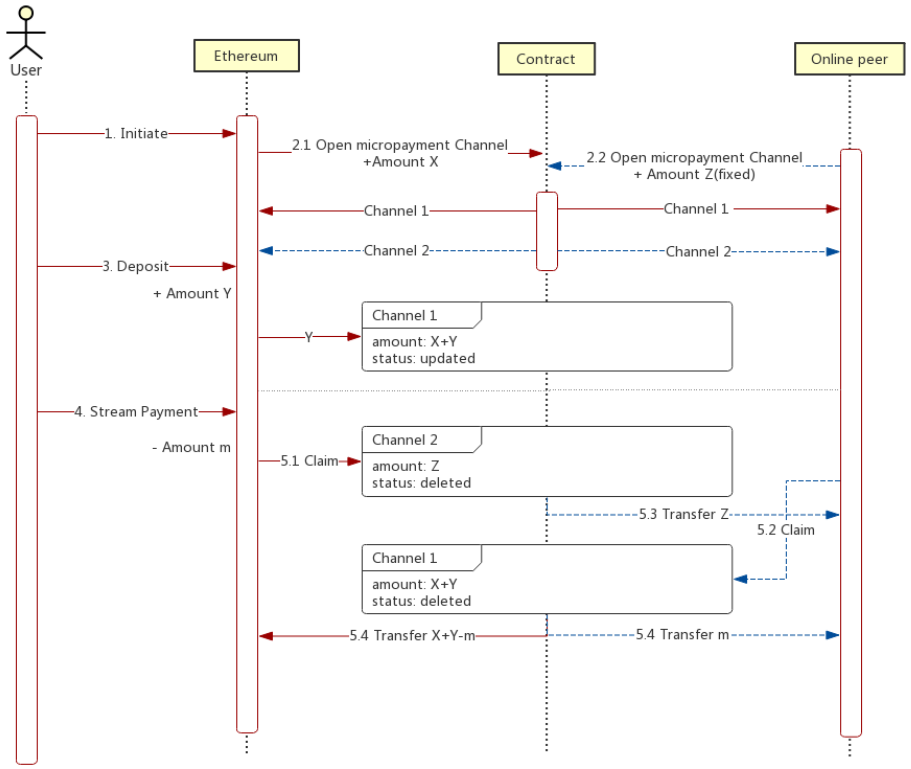
2. The user could perform deposit action with Y ETH to channel 1, changing the channel status to "update" and channel value to $X + Y$

3. a) If the swap is occurred from ETH to XRP, after one stream payment, the contract will keep the record of the amount of m ETH payment from user account to online-peer in channel 1.

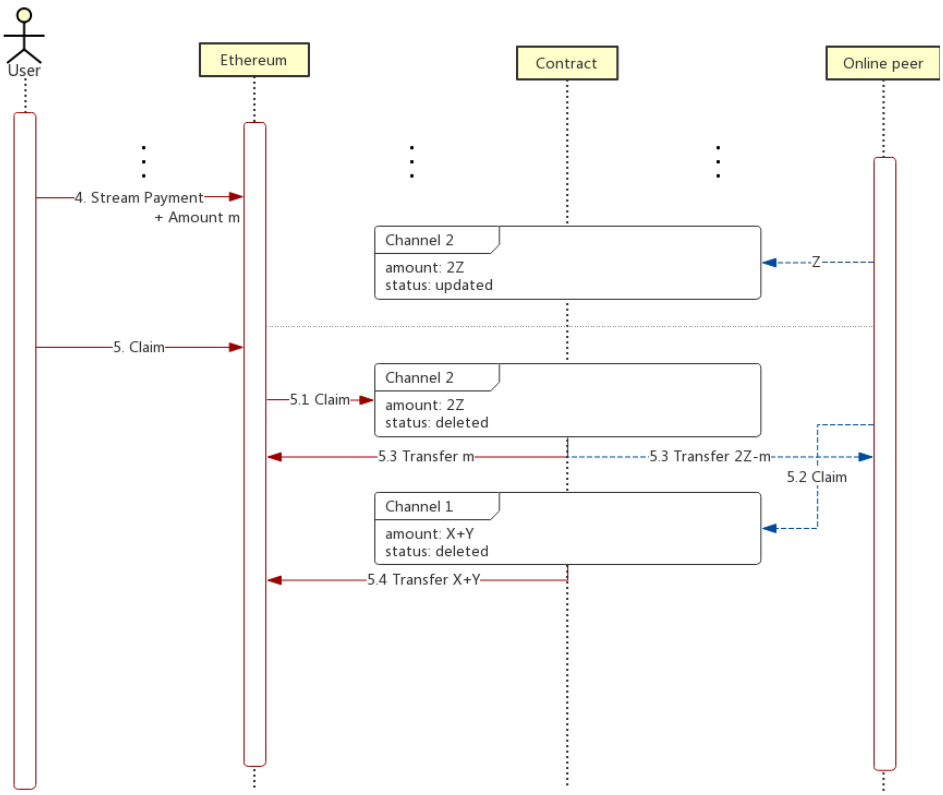
   b) If the swap occurs another way, each stream payment will arouse one deposit of Z ETH to channel 2 to avoid invalid payment due to balance error, and the contract will keep payment record of n ETH from online-peer to user in channel 2.

4. To finish the swap, the user will call withdraw from the wallet. Hence the user could claim the correct balance in payment channel 2, update the channel status to “delete”, and vice versa.

[1] https://github.com/machinomy/machinomy/blob/97ad47d07a7989b1f38466b1d4b8414e3a978b24/doc/Contract.md



(a) ETH->XRP



(b) XRP->ETH

Figure 4.5: Sequence diagram of payment channel

4.5 Summary

This chapter mainly introduces two implementations of different methods to accomplish the cross-chain transaction. First, we give a brief introduction of the applied ledger features and test environment settings. Then, we implement a simple HTLC atomic swaps framework from Bitshares to Ethereum, present one trade case result. Finally, we studied and tested one developed application integrated with interledger, dig into the working mechanism, and shows the result.

Chapter 5

Conclusion and Future Work

This chapter concludes important findings based on the entire project and proposed several promising future works.

5.1 Conclusion

This degree project identified the cross-chain problems and investigated 20 cross-chain solutions. Based on the scheme they are using, characterized, and sort them out into 3 categories, finished by a comprehensive comparison table shown in Appendix A.

Then to test the performance and get one universal standard solution aiming to cross-chain question, one implementation of HTLC atomic swap between two parties is realized, and a study of an ILP integration application working mechanism is carried out. During the implementation, I have explained how the process is kept as atomic and reliable as possible. However, compared to Interledger application, this will introduce American call options failure if bad actors performed the contract. Interledger application, on the other hand, introduces one specific transport protocol based on ILP. This protocol can stream money in small chunks, thus achieve the fast cross-chain payment. Similarly, streaming payment has the limitation of each payment amount, which will lead to the problem with a larger payment that requires multiple transactions happened continuously, not in once. In a way, it creates some inconveniences and decreases efficiency.

In conclusion, both cross-chain atomic swaps and cross-chain streaming can be regarded as the way to reach equilibrium for exchange. In cross-chain swaps, the money and data were securely locked without including any trusted party, but the counterparty risks could not be eliminated. In streaming payments, users will set a maximum loss equal to the amount of time and fees they are willing to pay, and the settlement can stop at anytime the exchange rate is worse than expected. While

minimizing the potential loss, it also increases the time to transact.

Based on what I have concluded, Interledger with stream payment could be the universal answer to realize the cross-ledger intercommunication so far, for the following reasons:

- Few ledgers support HTLC, specifically most traditional ledgers. By having connectors instead of ledger-enforced hashlock, the simple value transfer between connectors is the only requirement to support ILP. Thus, Interledger protocol has broader application scenarios than atomic swap does.
- Interledger is one strong functional protocol that analogous to the IP protocol over the Internet. It provides a solid under-layer protocol background for upper-layer developments such as the transport layer and application layer, which can be utilized in future services.

5.2 Future Work

The following aspects are being actively adopted as a part of future work:

- Since many different cross-chain solutions and projects have sprung up over the years. There is an increasing number of projects that aiming at handling the issues with these two chosen solutions are facing today. When some of the projects are developed enough to perform a test case, involving them into test and analysis are considered as the future work.
- The conclusion driven from Implementation and analysis is based on two different test application that does not apply in the same use scenario, although the result is valid because we evaluate the working mechanism. We could be benefit from getting more information through one common test environment in the future.
- At this stage, we can only provide theoretical analysis between cross-chain intercommunication between two parties. So pursuing a practical performance analysis towards larger cross-chain system with multiple users could leads to a more comprehensive conclusion.
- Technologies so far are mainly focused on solving cross-chain communication problems, but there is still a lack of research on the cross-chain property of ease of use, scalability and security, which are prerequisites for large-scale application of cross-chain.

Bibliography

[1] S. Brakeville and B. Perepa, "Blockchain basics: Introduction to distributed ledgers," *IBM, May*, 2016.

[2] A. Osello, A. Acquaviva, D. Dalmasso, D. Erba, M. Del Giudice, E. Macii, and E. Patti, "Bim and interoperability for cultural heritage through ict," in *Handbook of Research on Emerging Digital Tools for Architectural Surveying, Modeling, and Representation*. IGI Global, 2015, pp. 274–291.

[3] V. Buterin, "Chain interoperability," *R3 Research Paper*, 2016.

[4] S. Thomas and E. Schwartz, "A protocol for interledger payments," *URL https://interledger. org/interledger. pdf*, 2015.

[5] J. Poon and T. Dryja, "The bitcoin lightning network: Scalable off-chain instant payments," 2016.

[6] Consensys, "Btc-relay documentation," https://media.readthedocs.org/pdf/btc-relay/latest/btc-relay.pdf, september 10, 2016.

[7] D. Wood, "Polkadot light paper," https://polkadot.network/Polkadot-lightpaper.pdf, 20-09-2017.

[8] irisnet, "Cosmos white paper," https://github.com/irisnet/irisnet/blob/master/WHITEPAPER.md, 21st Feb, 2019.

[9] M. Herlihy, "Atomic cross-chain swaps," in *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing*. ACM, 2018, pp. 245–254.

[10] R. Russell, "Lightning networks part ii: Hashed timelock contracts (htlcs)," *See https://rusty. ozlabs. org*, 2015.

[11] A. Back, M. Corallo, L. Dashjr, M. Friedenbach, G. Maxwell, A. Miller, A. Poelstra, J. Timón, and P. Wuille, "Enabling blockchain innovations with pegged sidechains (2014)," *URL: tinyurl. com/mj656p7*, 2014.

[12] J. Lu, "Building super financial markets for the new digital economy," https://wanchain.org/files/Wanchain-Whitepaper-EN-version.pdf, 20th Jan, 2019.

[13] P. team, "Palletone yellow paper v1.0 beta," https://pallet.one/doc/PalletOne_yellowpaper_en.pdf, May, 2018.

[14] E. Buchman, "Tendermint: Byzantine fault tolerance in the age of blockchains," Ph.D. dissertation, 2016.

[15] J. Poon and V. Buterin, "Plasma: Scalable autonomous smart contracts," *White paper*, pp. 1–47, 2017.

[16] Oneledger.io, "Oneledger white paper," https://oneledger.io/whitepaper/oneledger-whitepaper.en.pdf, v2.0.

[17] S. D. Lerner, "Drivechains, sidechains and hybrid 2-way peg designs," 2016.

[18] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014.

[19] V. Buterin *et al.*, "A next-generation smart contract and decentralized application platform," *white paper*, vol. 3, p. 37, 2014.

[20] T. bitshares Team, "The history of bitshares, bitshares documentation," https://docs.bitshares.org/en/master/technology/history_bitshares.html, 29th Aug, 2018.

[21] D. Larimer, C. Hoskinson, and S. Larimer, "Bitshares: A peer-to-peer polymorphic digital asset exchange," *Accessed: Jan*, vol. 15, p. 2017, 2013.

[22] Ripple, "Xrp ledger overview," https://xrpl.org/xrp-ledger-overview.html, 23rd Jul, 2019.

[23] B. Chase and E. MacBrough, "Analysis of the xrp ledger consensus protocol," *arXiv preprint arXiv:1802.07242*, 2018.

[24] G. Governatori, F. Idelberger, Z. Milosevic, R. Riveret, G. Sartor, and X. Xu, "On legal contracts, imperative and declarative smart contracts, and blockchain systems," *Artificial Intelligence and Law*, vol. 26, no. 4, pp. 377–409, 2018.

[25] C. Dannen, *Introducing Ethereum and Solidity*. Springer, 2017.

[26] P. Gallagher and A. Director, "Secure hash standard (shs)," *FIPS PUB*, vol. 180, p. 183, 1995.

[27] Kava-labs, "Switch, swap btc, eth, dai and xrp in seconds. keep your private keys private," https://github.com/Kava-Labs/switch, 11st Jun, 2019.

[28] rfcs in Interledger, "Differences from previous versions of ilp," https://github.com/interledger/rfcs/blob/master/0027-interledger-protocol-4/0027-interledger-protocol-4.md#differences-from-previous-versions-of-ilp.

[29] I. team, "Stream: A multiplexed money and data transport for ilp," https://interledger.org/rfcs/0029-stream/, 27th Aug 2019.

[30] R. Han, H. Lin, and J. Yu, "On the optionality and fairness of atomic swaps," Cryptology ePrint Archive, Report 2019/896, 2019, https://eprint.iacr.org/2019/896.

[31] I. team, "Interledger protocol v4- unconditional-payment-channels," https://github.com/interledger/rfcs/blob/master/0027-interledger-protocol-4/0027-interledger-protocol-4.md#why-unconditional-payment-channels, 21th Aug 2019.

[32] G. Verdian, P. Tasca, C. Paterson, and G. Mondelli, "Quant overledger whitepaper," 2018.

Appendix A

Crosswise Comparison Table

The comparison table below summarizes twenty state-of-art cross-chain solutions, varying from protocols to platforms. The level of blockchain interoperability[1] is defined by Overledger develop team, 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 a connector;
- N-N: Many connected blockchains per time without connectors.

[1] This concept is adopted from Quant Overledger White paper [32]

Table 1: Summary of 20 Cross-chain Projects

Project	**Description**	**Scheme**	**Consensus**	**Type**	**Interoperability**	**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.	Notary	ILP	Platform	1-c-1	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	Blockchain	1-c-1	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	Blockchain	N-c-N	Decentralized exchange	No

Continued on Next Page

Continued from Previous Page

Project	Description	Scheme	Consensus	Type	Interoperability	Application	Compatibility with traditional ledger
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	Blockchain	N-c-N	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	Layer 2	1-1	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	Platform	N-c-N	Information exchange	No

Continued on Next Page

Continued from Previous Page

Project	Description	Scheme	Consensus	Type	Interoperability	Application	Compatibility with traditional ledger
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	Blockchain	N-c-N	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	Platform	N-c-N	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	Platform	N-c-N	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	Protcol	N-N	Most situation	Yes

Continued on Next Page

Continued from Previous Page

Project	Description	Scheme	Consensus	Type	Interoperability	Application	Compatibility with traditional ledger
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	Platform	N-c-N	DApps	No
PalletOne	It is a high-performance distributed cross-chain protocol, 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	Protcol	N-N	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	Protocol based	Platform	N-N	DApps	Yes

Continued on Next Page

Continued from Previous Page

Project	Description	Scheme	Consensus	Type	Interoperability	Application	Compatibility with traditional ledger
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	Layer 2	1-1	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	Platform	1-N	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	Layer 2	1-1	Bitcoin-related	No

Continued on Next Page

Continued from Previous Page

Project	Description	Scheme	Consensus	Type	Interoperability	Application	Compatibility with traditional ledger
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	Protcol	1-1	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	Platform	1-c-N	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	Layer 2	1-1	Sidechain expansion	No

Continued on Next Page

Continued from Previous Page

Project	Description	Scheme	Consensus	Type	Interoperability	Application	Compatibility with traditional ledger
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	Platform	N-c-N	Transactional	Yes

Appendix B

Code Pieces

I. Atomic swaps based on HTLC

Listing 1: Bitshares deploy the HTLC

```
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;

    let tr = new TransactionBuilder();

    let preimageValue = secret;
    let preimage_hash_calculated = hash;

    let operationJSON = {
      from: fromAccount.get("id"),
      to: toAccount.get("id"),
      fee: {
        amount: 0,
        asset_id: "1.3.0"
      },
      amount: {
        amount: amount,
        asset_id: "1.3.0"
      },
      preimage_hash: [2, preimage_hash_calculated],
```

```
      preimage_size: preimageValue.length,
      claim_period_seconds: time_lock
    };

    tr.add_type_operation("htlc_create", operationJSON);

    return tr.set_required_fees().then(() => {

      tr.add_signer(spKey, spKey.toPublicKey().toPublicKeyString());

      return tr

        .broadcast()
        .then(result => {
          console.log(
            "BTS HashTimelockContract was successfully created!");
          let htlcResponse = result[0].trx.operation_results[0];
          let htlc_id = htlcResponse[1]
          return htlc_id
        })
        .catch(error => {
          console.error(error);
        });
    });
  });
};
```

Listing 2: Ethereum refund after expiration

```
async function waitForHTLC(contractId) {
  const htlc = new web3.eth.Contract(HTLC_abi, HTLC_contract_address)


  return new Promise((resolve, reject) => {
    const poll = setInterval(async function() {
        const contractArr = await htlc.methods.getContract(contractId)
    .call()
        const contract = htlcArrayToObj(contractArr)
        const unlockTime = Number(contract.timelock)
        const secret = contract.preimage
        const block = await web3.eth.getBlock('latest')
        if (secret !== '0
    x0000000000000000000000000000000000000000000000000000000000000000'
    ) {
        clearInterval(poll)
        btsSecret = web3.utils.hexToAscii(secret)
        resolve(btsSecret)
      } else if (block.timestamp > unlockTime) {
        clearInterval(poll)
        reject('ETH HTLC timed out')
      }
    }, 5e3)
  })
}

```

```
24
25
26
27 async function refundHTLC(sender, contractId) {
28   const htlc = new web3.eth.Contract(HTLC_abi, HTLC_contract_address)
29
30   let contractArr = await htlc.methods.getContract(contractId).call()
31   let contract = htlcArrayToObj(contractArr)
32   const amount = web3.utils.toBN(contract.amount)
33   const timelock = Number(contract.timelock)
34   if(nowSeconds() < timelock){
35     throw 'expected failure due to timelock'
36   }
37
38
39   const senderBalanceBefore = await getBalance(sender)
40   const refundTx = await htlc.methods.refund(contractId).send({
41     from: sender,
42     gas: 3000000
43   })
44   const tx = await web3.eth.getTransaction(refundTx.transactionHash)
45   const expectedBalance = senderBalanceBefore
46                                   .add(amount)
47                                   .sub(txGas(refundTx, tx.gasPrice))
48           assertEqualBN(
49             await getBalance(sender),
50             expectedBalance,
51             "sender balance doesn't match"
52           )
53           contractArr = await htlc.methods.getContract(contractId).
     call()
54           contract = htlcArrayToObj(contractArr)
55           assert.isFalse(contract.withdrawn) // withdrawn still false
56           assert.isTrue(contract.refunded) // refunded set
57
58   let actualBalance = await getBalance(sender)
59   actualBalance = web3.utils.fromWei(actualBalance,'ether')
60   console.log("ETH HashTimelockContract was successfully refunded!");
61   console.log(`Account: ${sender} has balance of ${actualBalance} ETH
       `);
62 }
```

II. Ethereum smart contracts

Listing 3: Create hash timelock contract example

```
1 pragma solidity ^0.5.0;
2
3 contract HashTimelock{
4     event LogHTLCNew(
5         bytes32 indexed contractId,
6         address indexed sender,
```

```
7         address indexed receiver,
8         uint amount,
9         bytes32 hashlock,
10        uint timelock
11    );
12  ...
13
14  modifier fundsSent() {
15        require(msg.value > 0, "msg.value must be > 0");
16        _;
17    }
18    modifier futureTimelock(uint _time) {
19        require(_time > now, "timelock time must be in the future");
20        _;
21    }
22    ...
23
24    function newContract(address payable _receiver, bytes32 _hashlock,
       uint _timelock)
25        external
26        payable
27        fundsSent
28        futureTimelock(_timelock)
29        returns (bytes32 contractId)
30    {
31        contractId = sha256(
32            abi.encodePacked(
33                msg.sender,
34                _receiver,
35                msg.value,
36                _hashlock,
37                _timelock
38            )
39        );
```

This contract provides a way to create and keep HTLCs for ETH.[1] 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.

[1]The complete codes can be found at https://github.com/Fy45/BTS-ETH-atomic_swaps/blob/master/contract/HashedTimelock.sol

Listing 4: Ethereum micro-payment channel structure – Machinomy

```
1   struct PaymentChannel {
2       address sender;
3       address receiver;
4       uint256 value; // Total amount of money deposited to the
    channel.
5
6       uint256 settlingPeriod; // How many blocks to wait for the
    receiver to claim her funds, after sender starts settling.
7       uint256 settlingUntil; // Starting with this block number,
    anyone can settle the channel.
8   }
9
10  mapping (bytes32 => PaymentChannel) public channels;
11
12  event DidOpen(bytes32 indexed channelId, address indexed sender,
    address indexed receiver, uint256 value);
13  event DidDeposit(bytes32 indexed channelId, uint256 deposit);
14  event DidClaim(bytes32 indexed channelId);
15  event DidStartSettling(bytes32 indexed channelId);
16  event DidSettle(bytes32 indexed channelId);
17
18  /*** ACTIONS AND CONSTRAINTS ***/
19
20  /// @notice Open a new channel between `msg.sender` and `receiver
    `, and do an initial deposit to the channel.
21  /// @param channelId Unique identifier of the channel to be
    created.
22  /// @param receiver Receiver of the funds, counter-party of `msg.
    sender`.
23  /// @param settlingPeriod Number of blocks to wait for receiver to
     `claim` her funds after the sender starts settling period (see `
    startSettling`).
24  /// After that period is over anyone could call `settle`, and move
     all the channel funds to the sender.
25  function open(bytes32 channelId, address receiver, uint256
    settlingPeriod) public payable {
26      require(isAbsent(channelId));
27
28      channels[channelId] = PaymentChannel({
29          sender: msg.sender,
30          receiver: receiver,
31          value: msg.value,
32          settlingPeriod: settlingPeriod,
33          settlingUntil: 0
34      });
35
36      DidOpen(channelId, msg.sender, receiver, msg.value);
37  }
38
39  /// @notice Ensure `origin` address can deposit money into the
    channel identified by `channelId`.
40  /// @dev Constraint `deposit` call.
41  /// @param channelId Identifier of the channel.
```

```
42      /// @param origin Caller of `deposit` function.
43      function canDeposit(bytes32 channelId, address origin) public view
         returns(bool) {
44          PaymentChannel memory channel = channels[channelId];
45          bool isSender = channel.sender == origin;
46          return isOpen(channelId) && isSender;
47      }
48
49      /// @notice Add more money to the contract.
50      /// @param channelId Identifier of the channel.
51      function deposit(bytes32 channelId) public payable {
52          require(canDeposit(channelId, msg.sender));
53
54          channels[channelId].value += msg.value;
55
56          DidDeposit(channelId, msg.value);
57      }
```