

Blockchain for Security of A Cloud-Based Online Auction System

Yun Shu

A thesis submitted to the Auckland University of Technology
in partial fulfillment of the requirements for the degree of
Master of Computer and Information Sciences (MCIS)

2017

School of Engineering, Computer and Mathematical Sciences

Abstract

In recent decades, Internet auctions have already been grown up as the most significant e-commerce business model in worldwide. Meanwhile, with the rapid rising of cloud computing over the past few years, the legacy online auction platform is gradually replaced by service-oriented cloud computing in real time. This thesis describes design and implementation of a high-performance online auction system over the cloud. We propose the methodology to provide persistent state records during the auction process so that we can ensure the reliability of submitted bid price, fairness and guarantee the security of price message in the delivery process. We employ the actor-based applications to achieve stateful, parallel and distributed architecture. Moreover, utilising distributed databases to provide secure and efficient data storage. Our preliminary result provides the guidelines for implementation of high-performance and real-time bidding online auction.

Auction fraud has become the highest threat and hazard to the future of this business model (Grazioli & Jarvenpaa, 2000). In this thesis, we are going to demonstrate the details of blockchain which will provide a new perspective to resolve this problem. It is able to be used for current financial services, certificates, remittances and online payments. Furthermore, it also provides several crucial services such as smart contract, smart property, trust system and security services (Gareth William Peters, Panayi, & Chapelle, 2015). This thesis will discuss how to apply a private blockchain to a cloud-based online auction and the principle of operation. The purpose is to fundamentally solve the problem of online fraud caused by information asymmetry of electronic transactions. To the best of our knowledge, this is the first time that the blockchain is applied to authentication of online auction. Our preliminary result is for preventing auction fraud from the aspects of smart properties and smart contract.

Keywords: auction fraud, blockchain, elliptic curve digital signature algorithm, real-time online auction, actor model, concurrent, parallel and distributed system, real-time service-oriented cloud computing.

Table of Contents

Abstract	I
Table of Contents	II
List of Figures	V
List of Tables.....	VII
Attestation of Authorship.....	VIII
Acknowledgment	IX
Chapter 1 Introduction	1
1.1 Background and Motivation.....	2
1.2 Research Question.....	3
1.3 Contribution	4
1.4 Objective of This Thesis.....	4
1.5 Structure of This Thesis	5
Chapter 2 Literature Review	7
2.1 Introduction	8
2.2 MVVM and Serverless Framework	10
2.2.1 MVVM.....	10
2.2.2 Serverless Framework.....	12
2.3 Actor Framework	14
2.3.1 Erlang.....	14
2.3.2 Akka.....	14
2.3.3 Orleans	15

2.4	Traffic Reliability in Online Auction	16
2.5	Auction Model.....	17
2.6	Blockchain.....	20
2.6.1	Block.....	24
2.6.2	Decentralized Framework.....	27
2.6.3	Smart Contract and Smart Property	28
2.6.4	Properties Exchange on Blockchain	32
2.6.5	Tradenet	35
2.6.6	Challenges of Blockchain	36
2.6.7	Hawk.....	37
2.7	Online Fraud.....	40
Chapter 3 Methodology		45
3.1	Introduction	46
3.2	Design.....	46
3.2.1	Immutable Message	46
3.2.2	Fault Tolerance of The Actor	48
3.2.3	Location Transparency.....	50
3.2.4	State Persistence in Actor Model	50
3.3	Distributed Database	51
3.3.1	High Performance Read/Write.....	51
3.3.2	Fault Tolerance of The Database.....	52
3.4	Elliptic Curve Cryptography	52
3.4.1	Elliptic Curve Digital Signature Algorithm	52
3.4.2	Implementation	54
3.5	Blockchain Network Architecture	56

3.5.1	Mining Nodes.....	58
3.5.2	Transaction Nodes.....	61
3.5.4	Actor Framework Works with Blockchain	62
Chapter 4 Results		64
4.1	Actor Framework	65
4.2	ECDSA.....	67
4.3	Transaction Verification in Blockchain	71
4.4	Implementation of Tokens on Private Blockchain	72
Chapter 5 Analysis and Discussions		78
5.1	Analysis.....	79
5.1.1	NoSQL Database.....	79
5.1.2	Blockchain	80
5.2	Discussions.....	83
Chapter 6 Conclusion and Future Work.....		85
6.1	Conclusion.....	86
6.2	Future Work.....	88
References.....		89

List of Figures

Figure 2.1 Architecture of legacy web applications.....	9
Figure 2.2 The model-view-viewmodel framework.	11
Figure 2.3. Serverless framework	13
Figure 2.4 Understanding of an actor	16
Figure 2.5 A supervisor and its children in a graph	16
Figure 2.6 Overview of online auction process	18
Figure 2.7 The workflow of real-time bidding	19
Figure 2.8 The diagram of blockchains	22
Figure 2.9 Bitcoin network (resource from bitcoin)	22
Figure 2.10 An example of mining process in the blockchain.....	23
Figure 2.11 The details of a branch of blockchain.....	24
Figure 2.12 A block in blockchain that is represented by JSON	25
Figure 2.13 Digital signature in blockchain technology	26
Figure 2.14 Decentralized network.....	27
Figure 2.15 Hawk overview.....	38
Figure 3.1 Overview of Online Auction System.....	46
Figure 3.2 The communications between actor systems	47
Figure 3.3 A bidder actor and its behaviours	48
Figure 3.4 A diagram of the comparison actor.....	49
Figure 3.5 Fault tolerance of actor model.....	49
Figure 3.6 State persistence for bidding	51
Figure 3.7 The addition of two distinct elliptic curve points	53
Figure 3.8 Doubling a point on an elliptic curve	53


Figure 3.9 Blockchain network architecture	56
Figure 3.10 Summary of deployment of our private blockchain	57
Figure 3.11 Details of our private blockchain deployment.....	58
Figure 3.12 Resource allocation of miners	59
Figure 3.13 Coinbase account.....	60
Figure 3.14 Actor system, mongo database and blockchain network.....	62
Figure 3.15 Actor communicate with blockchain	63
Figure 4.1Blockchain hashcode example	69
Figure 4.2 Private key and public key in hexadecimal	70
Figure 4.3 The transaction file needs to be signed by private key	70
Figure 4.4 House token	73
Figure 4.5 House token in wallet	74
Figure 4.6 Smart contract of coffee cup.....	74
Figure 4.7 Set a new project	75
Figure 4.8 Bidding the price	76
Figure 4.9 Bidding List.....	76
Figure 4.10 Settle the price	77
Figure 4.11 The final transaction on Blockchain	77
Figure 5.1 Insert ServerTimeSync status into database	79
Figure 5.2 Query state from comparison actor and bidder actor	80
Figure 5.3 Difficulty incensement in recent two years	81
Figure 5.4 Hash rate incensement in recent two years.....	81
Figure 5.5 Incensement of cost per transaction in recent two years	82
Figure 5.6 Block size incensement in recent two years	83

List of Tables

Table 2.1 An example of blockchain applications in version 2.0	28
Table 2.2 A record of Alice's account.....	32
Table 2.3 A changed record of Alice's account.....	33
Table 2.4 A record of Bob's account.....	33
Table 3.1 Mnemonic code and root key	61
Table 4.1 A feature comparison of the actor framework and the client-server architecture	65
Table 4.2 TimeAsync Actor State in MongoDB	66

Attestation of Authorship

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person (except where explicitly defined in the acknowledgments), nor material which to a substantial extent has been submitted for the award of any other degree or diploma of a university or other institution of higher learning.

Signature:  Date: 01 Nov 2017

Acknowledgment

This research work was completed as the part of the Master of Computer and Information Sciences (MCIS) course at the School of Engineering, Computer and Mathematical Sciences (SCMS) in the Faculty of Design and Creative Technologies (DCT) at the Auckland University of Technology (AUT) in New Zealand. I would like to sincerely thank my parents for the financial support they provided during my entire time of master's study in Auckland.

My deepest thanks are to my primary supervisor Dr Wei Qi Yan who has provided me with much appreciated technological guidance and support. I believe that I would not have been able to achieve my Master Degree without his invaluable help and supervision. Also, I would like to appreciate my secondary supervisor Dr Jian Yu and school administrators of our school for their support and guidance through the MCIS in the past years.

Yun Shu

Auckland, New Zealand

01 Nov 2017

Chapter 1

Introduction

The first chapter of this thesis consists of five sections. In the first section, background and motivation of this thesis are introduced, an actor framework is utilised to establish a high-speed, high-scalability and low-latency network for an online auction system. Meanwhile, blockchain technology also has been introduced to resolve the online fraud issue. Objectives will be discussed in the fourth section. This chapter also covers the details of research questions after the in-depth, comprehensive understanding of the relevant literature and research background. Finally, we will present an overview of the structure of this thesis in Section 5.

1.1 Background and Motivation

With the ubiquitous Internet connections in the world, the online auction platform is reshaping the market by providing professional services, like the third part guarantee payment, or digital signatures and certificates. These professional services are no longer limited by the typical location of buyers and sellers (Ackerberg, Hirano, & Shahriar, 2006). The research outcome (Krasnokutskaya, Terwiesch, & Tiererova, 2016) also indicates that eighty percent of online transactions are carried out by participants from foreign countries and regions. However, the e-commerce industry, especially online auction, is up against several difficulties. For example, traditional online auction system is robust to handle large-scale data transmission from different regions at the high-speed and parallel network. Additionally, the traditional architecture must face significant issues that come from traffic reliability and massive data processing within a short period. Furthermore, the online fraud in this industry is also a severe shortcoming.

Thus, we are eager to find a new way to solve the issues above. Five key features from existed actor frameworks are demonstrated: safe message delivery, mobility, state persistence, location transparency and fair scheduling (Karmani, Shali, & Agha, 2009). The mobility, location transparency and fair scheduling are able to provide the data consistency and integrity on the distributed cloud services; the state persistence is able to improve the packet loss and traffic rates for cloud online auction model; the mechanism of immutable message delivery guarantees that messages are not able to be tampered. We believe that the implementation of the actor framework in an online auction system will help to provide a high-speed, low-latency and high-stability of the network environment. Moreover, the actor framework supports scalability. It is easy for us to apply this framework to a distributed architecture.

However, actor framework still is not able to solve the online fraud issue. One key goal of our thesis is to facilitate online transactions between participants within the untrusted environment. The search outcome illustrated that the incredible trading environment is caused by complicated reasons like geographical separation, or uncertainty, inconvenience or corruption of existing legal systems (Wood, 2014a).

Several types of research about the Bitcoins and Ethereum have enlightened us about leveraging blockchain for online auction system, especially financial fairness. There are several prior work explored about how to implement the blockchain technique to achieve the financial fairness. Researchers demonstrate how Bitcoin is utilised to guarantee the fairness of a secure multi-party computing protocol that also performs various types of offline security calculations (Andrychowicz, Dziembowski, Malinowski, & Mazurek, 2014; Kiayias, Zhou, & Zikas, 2016; Kumaresan & Bentov, 2014; Zyskind, Nathan, & Pentland, 2015). Furthermore, the Ethereum blockchain has introduced the smart contract which is an executable code (Delmolino, Arnett, Kosba, Miller, & Shi, 2016). We are able to automatically move digital assets, or transfer the ownership according to any pre-specified rules in the smart contract (Buterin, 2014).

1.2 Research Question

As we mentioned, this thesis aims at the data consistency and integrity during the high-performance network environment for online auction system, and implementations of blockchain database to provide the confidential transaction in the untrusted environment. Analyzing the methods and techniques helps us to know the necessary procedures of implementing actor framework and the Elliptic Curve Digital Signature Algorithm. Therefore, the research questions of this thesis are:

- (1) Could we use cloud computing to improve the performance of online auction system?
- (2) Is there any cloud-based technology which could provide a new idea to solve the issues of online fraud?

We attempt to find answers to the following question which is developed from the primary question:

“What are the security algorithms for online auction system? Which algorithm is the best one for our research? Which cloud-based framework is the best solution to provide the immutable message delivery that ensures our bid price cannot be tampered during the data passing from end to end?”

The core idea of this thesis is stated actor framework and blockchain. Thus, the techniques that we adopted in this research project need to be evaluated, and proper techniques need to be chosen so as to implement the best result of blockchain and state actor model.

1.3 Contribution

The main contributions of this thesis include: (1) We use the actor framework to improve the packet loss and traffic rates for our cloud online auction model. The message is wrapped in the web stock, so we cannot guarantee that any messages can be received or dispatched each time successfully. Thus, the additional implementation must be taken to actor mode like at-least-once in Orleans. It requires retry when transport losses. (2) We implement the mechanism of immutable message delivery and ensure that our bid price cannot be tampered during the data passing from end to end. (3) We record all actors' state persistently in document database so that we are able to approve the data persistence, which is a highly-challenging task in the high-latency database like SQL database.

Also, this is also the first time that the blockchain is applied to authentication of online auction. Our preliminary result is for preventing auction fraud from the aspects of smart properties and smart contract.

1.4 Objective of This Thesis

Firstly, this thesis introduces actor framework relates to offer the data consistency and integrity during the high-performance network environment. Moreover, the features and principles of the actor framework will be illustrated and evaluated in this thesis. Furthermore, Ethereum blockchain technology is about implementation of smart property and smart contract for providing reliable online transactions underlying the untrusted trading environment.

Secondly, in order to implement the actor model and blockchain system on the cloud service, a conceptual framework is also presented for our cloud-based online auction system. Therefore, the overall objective of this thesis is divided into three different

parts which include:

- design of actor framework for English auction model;
- implementation of elliptic curve digital signature algorithm for elaborating the working mechanism of blockchain;
- Development and implementation of the private blockchain and actor framework on Azure;

Finally, the comparison of performances and algorithms for the actor model and blockchain will be presented in this thesis.

1.5 Structure of This Thesis

The thesis is structured as follows:

In Chapter 2, literature will be reviewed, such as the previous studies in message-driven development, and online fraud solutions in an online auction for the area of state actor framework and blockchain. Actor framework and blockchain are studied as a high-level processing in semantic. Thus, Chapter 2 will introduce the fundamentals of the Elliptic Curve Digital Signature Algorithm, the data consistency and integrity during the high-performance network environment. So that we are able to improve the packet loss and traffic rates for our cloud online auction model. Meanwhile, we are able to provide a secure transaction mechanism between clients underlying an untrusted trading environment.

In Chapter 3, the explanation of research methodology of this thesis will be stated. Moreover, potential solutions and answers will also be presented. Moreover, the experimental layout and design as well as datasets and implementations with the evaluation methods, will be presented.

In Chapter 4, the methodologies and algorithms we presented will be implemented. Moreover, experimental results and outcomes will be detailed with supports of tables and figures. The limitations of this project will also be addressed as well.

In Chapter 5, analysis and discussions are depicted based on experimental results and outcomes we acquired in Chapter 4. Finally, conclusion and future work will be presented in Chapter 6.

Chapter 2

Literature Review

With in-depth analysis of the research question and rationale reviews of the previous studies, the focus of this thesis is on actor framework and blockchain. For instance, the actor-based framework is able to achieve stateful, parallel and distributed architecture. We utilise distributed databases to provide secure and efficient data storage. Our preliminary result provides the guidelines for implementation of high-performance and real-time bidding online auction. The-state-of-art of state actor model and message driver methods will be summarized in this chapter.

2.1 Introduction

Cloud computing is becoming increasingly attractive to firms and developers today. Compared with the traditional Internet service provider, the legacy service provider is divided into infrastructure vendors and service providers. The infrastructure providers (Platform as a Service), such as Microsoft Azure, Amazon Web Application Service, and Google App Engine. The services provide impressive, reliable, and cost-efficient cloud-based platforms and lease these platforms to the service providers who rent resources from infrastructure providers to serve the end clients (Armbrust et al., 2010). Moreover, those developers who worked for the service providers have significant opportunities to transform their innovative ideas into highly scalable Internet services. We also call them as Software as a Service (SaaS). These services can be easily expanded to large scales so as to handle the expeditious increase in service demands (Armbrust et al., 2009).

Because of these features of SaaS, we, therefore, have the opportunity to deploy actor-based applications in a parallel and distributed system on the cloud to construct the bottlenecks of traditional client / server framework of the online auction. The actor model (Hewitt, Bishop, & Steiger, 1973) adopts an abstract concept to describe concurrency of the program that is centred on the actor unit which performs distributed computations and communicates through asynchronous information exchange. Thus, the actor is suitable for developing large-scale parallel programs. However, the concurrency of actors is limited by hardware resources and capability of logical computations (Agha, 1985). As the rapid progression and innovation of cloud technology in recent years, hardware constraints gradually weakened. The actor model is increasingly being applied to highly concurrent applications on SaaS, such as Microsoft Orleans and JVM Akka.

The main technical bottleneck of traditional online auction system is that it is hard to handle significant amounts of data from different regions in a highly concurrent and parallel environment. More specifically, when an online auction system receives quotes from all over the world, the traditional tree-tired architecture shown in Figure 2.1 will place cache area between middle tire and physical storage for improving I/O

performance (Power & Li, 2010). Unfortunately, usage of the cache will directly lose the concurrency. The cache manager or application must implement concurrency control policy to avoid deviations resulting from concurrent updates to a cached object (Miller, M. S., Tribble, & Shapiro, 2005). Hence, the traditional architecture has to face major issues from traffic reliability and massive data processing within a short period. With or without the cache, this pattern cannot fulfil the requirement of conformity on a cache with rapid reaction for interactive access (P. A. Bernstein, Bykov, Geller, Kliot, & Thelin, 2014). Eventually, these problems will result in excessive use of CPU resources and physical memory depletion. On the other hand, a traditional relational database that is commonly utilised in a concurrently read or write scenario cannot provide reliable data concurrency and consistency.

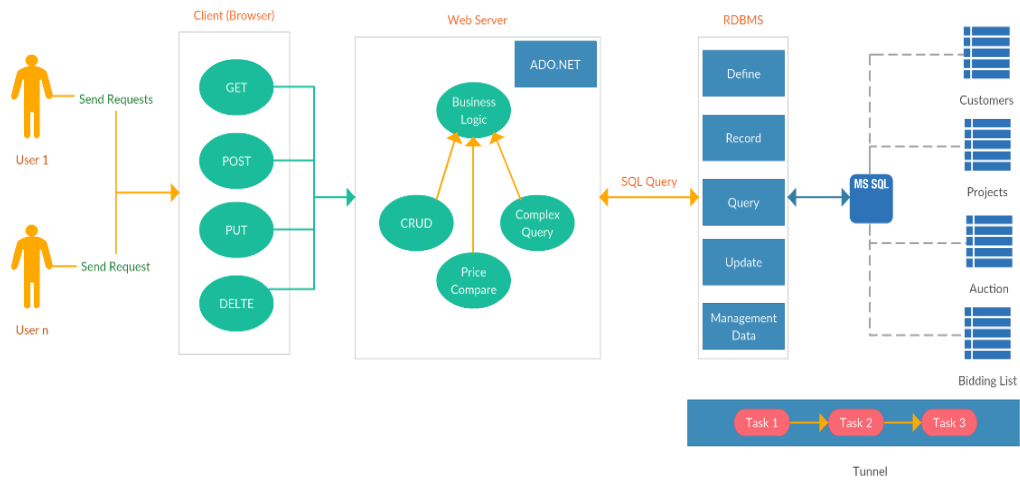


Figure 2.1 Architecture of legacy web applications

In this thesis, we discuss the methodology that can be implemented to convert the structure of traditional English online auction into actor architecture, so that we can guarantee data consistency and integrity in the high-performed environment of the online auction. We utilise the mechanism of safe message delivery which ensures the security of bid data from end to end. Moreover, the internal state of the actor offers the reliability of persistent state records during the auction process. Finally, fair scheduling, location transparency and mobility of state will assure data integrity and consistency to the actor model (Miller, M. S., Tribble, & Shapiro, 2015) resulting in offering high-

concurrency and low-latency online auction service. To the best of our knowledge, this is the first time that the actor framework is applied to the online auction.

2.2 MVVM and Serverless Framework

2.2.1 MVVM

The reason we use MVVM pattern is that the flexibility of this pattern provides a convenience for a decentralised distributed architecture. This pattern was used early in Windows Presentation Foundation(Smith, 2009). With the progress of the technology, MVVM pattern has now been widely used in a variety of e-commerce platform. This architecture has a quite good compatibility for client-side (browser, mobile, or desktop application). MVVM pattern model is able to separate business logic and UI easily. So, we can increase the reusability of our code. Meanwhile, this makes the code easily to be developed, test and maintain (L. Liu, 2012).

Model tier is responsible for handling the real data and information. It also includes all the business objects and the relations of objects such as one to one, one to many, and many to many. We usually called this tier as the secondary package of business information. According to the rules of object-oriented, our business information will be packaged into the business objects. Moreover, we transfer this information into the view model. For example, our auction project objects, bidder objects, time sync objects, immutable message objects, etc. In our whole project, the model tier is the core. Because of this, we are not able to expose our core business objects directly to users. If the business information needs to be changed, we only need to update our model tier.

View tier is mainly responsible for communicating with the customer. We also called this tier as UI. It displays information to our client. This is similar to MVC architecture; the users are able to obtain useful information and enjoyable user experience by using UI in the view tier. In MVVM design model, our view tier is not limited to the desktop application, which can be a web page or mobile client. Most of the view tier today is compatible with the different size of the screen from 4K display to mobile size. The most significant thing is that there should not include any of

business logic in this tier. Therefore, view tier is able to be coded by using front-end languages such as HTML and JavaScript. Another benefit of MVVM framework is that we are able to develop the front end (View) and the back end at the same time. This dramatically increases the development efficiency. To be more specific, when we develop the view tier, we only need to mock up the fake data for the view model. As the view tier, we do never care about the business logic. Also, the view tier is most frequently asked to make changes to clients. This design pattern reduces the impact of UI changes (Gao, Zhang, & Yao, 2012).

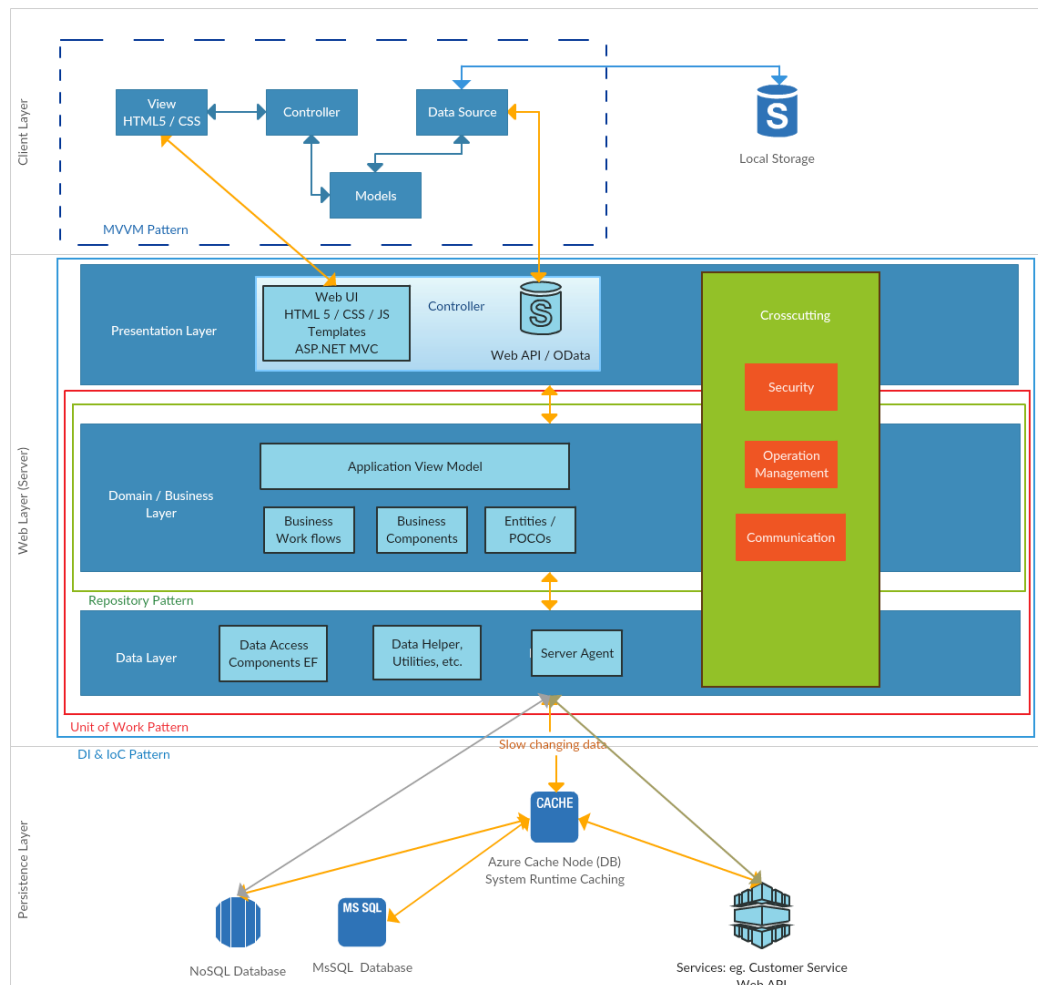


Figure 2.2 The model-view-view model

View model tier is holding all display logic for view tier. It will not quote the view controllers directly instead of binding command and data of view controllers in view tier to view model layer (X. Li et al., 2015). The role of the view model layer is the

coordination of the objects in the view tier and model tier. So, the view model layer is the bridge between the view layer and the model layer. As we have already indicated that we should never expose our model tier directly to the client side. View model plays a pivotal role for data exchanging and data validation. When the data comes from the client side, view model needs to validate all data to make sure that there is no dirty data will be passed to model tier. On the other hand, when data comes from the model layer, we need to expose the necessary information to view model tier. So, we say that the view model layer connects the entire system, separating the view layer and the model layer. The design model achieves high cohesion and loosely coupled targets. The view model layer is the link layer between the view layer and the model layer. In this layer, the client-side operation commands of the view layer are transferred to the business functions identified in the model layer and the business information returned from the model layer is displayed for the client through the view layer. Therefore, the view model layer is the core one of the overall system.

2.2.2 Serverless Framework

With the progression and innovation of cloud computing frameworks, increasingly efficient, lightweight, abstracted and virtualised framework (like docker container) have indicated that the public cloud market has evolved rapidly (D. Bernstein, 2014). Yu et al. (2010) pointed out that because the current virtual machine-based technology has already reached very stable level within the cloud computing market, the main trend begins to focus on container-based virtualization, such as Docker or RaaS Cloud (Li, Tang, & Chou, 2015; Sáez, Andrikopoulos, Sánchez, Leymann, & Wettinger, 2015).

Enterprises are beginning to accept cloud systems, it is not a viable solution, but also a significant strategy (McGrath, Short, Ennis, Judson, & Brenner, 2016). This has led to the emergence of many competing or parallel patterns for building efficient and effective cloud solutions. Cloud technology, as a cloud system, is used in conjunction with a business process framework, such as Roboconf (Pham et al., 2015), which creates a powerful automated computing system that greatly enhances enterprise productivity. Ad Hoc Cloud Computing is another example, which is utilized to improve productivity and efficiency (McGilvary, Barker, & Atkinson, 2015). The use

of cloud events in conjunction with the Internet of Things (IoT) is probably the most influential. As more and more data sources grow, it is increasingly evident that we are behind our ability to handle these data. Software-defined ecosystems designed to handle data provide a space for many diverse and efficient applications of cloud events (Parashar, Abdelbaky, Zou, Zamani, & Diaz-Montes, 2015).

For realizing the distribution from end to end, actor-model employs web socket protocol to maintain the data persistent with the front end and central database. The decentralized structure requires the serverless framework. To be more specific, we are not going to build a web API server to expose from the endpoints to the front end. This kind of traditional server framework is difficult to scale out. Meanwhile, it obviously does not have enough capacity to support distributed database. Most of the cloud server platform like Azure or AWS has already provided the server fewer components such as Lambda on AWS or Fabric Server on Azure. For instance, we are able to build an actor by using Fabric Server called Time Sync actor which can be used to sync time with clients. We can copy and rebuild the actor as many times as we want to. Even if one of them is failed, the others would keep working. We can build bidder and compare actors in the same way. The actors are communicating between each other by the immutable message.

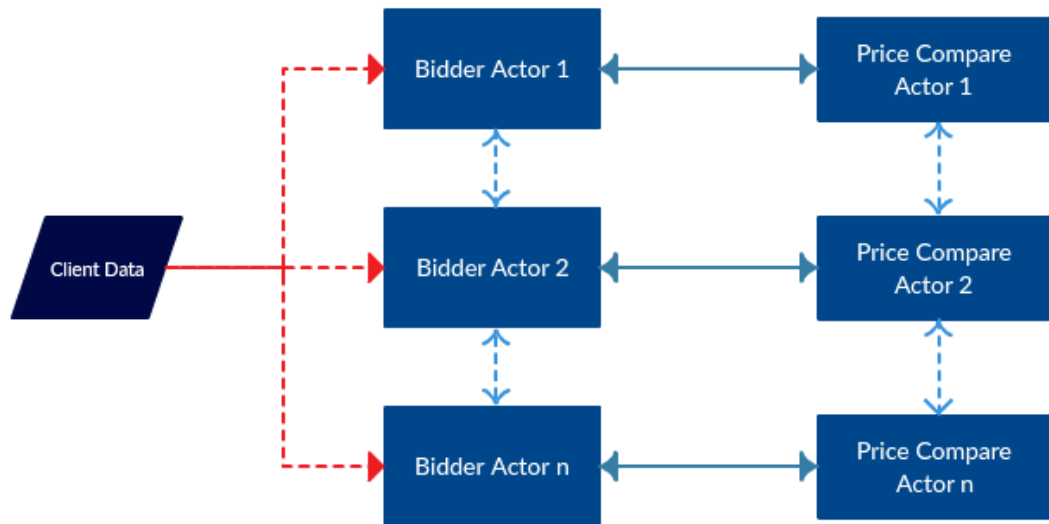


Figure 2.3. Serverless framework

2.3 Actor Framework

Five key features are demonstrated (Karmani et al., 2009) for comparing those existed actor frameworks: safe message delivery, mobility, state persistence, location transparency and fair scheduling. There are now three main actor frameworks: Erlang, Akka, and Orleans.

2.3.1 Erlang

The Erlang is a functional programming language (Armstrong, Viriding, Wikström, & Williams, 1993). The actor in Erlang is called process. Compared with Orleans, actors in Erlang are created explicitly (Orleans uses virtual actors). So once the actor is created, its location cannot be changed anymore (Armstrong, 2007). Meanwhile, Erlang uses the link operation to handle the erratic propagation. The problem is that if the processes are not linked together, the process will die silently. Moreover, Erlang utilises the Open Telecom Platform (OTP) to expand performance and capabilities from distribution, fault tolerance, and concurrency. OTP provides supervisor tree for unexpected error handling (Vinoski, 2007), but it will kill all its children and recreates them once one of them dead. Otherwise, the developer needs to control the process of lifecycle manually. By contrast, Orleans and Akka create and garbage collect their actors on runtime automatically.

2.3.2 Akka

Akka is an actor-based framework, which is available for Java, Scala, and C#. The main features are almost as same as Orleans. For instance, each actor is single threaded; the private state is able to access through the reference. The difference is that the actor in Akka is named by the path that illustrates the hierarchy structure from the supervisor to the children.

2.3.3 Orleans

Orleans is the latest actor framework from Microsoft which blends several technologies from previous actor frameworks, such as Erlang and Akka. For example, it fully supports immutable message delivery, state persistence, efficient and fair scheduling for actors of sharing CPU, location transparency. Meanwhile, Orleans also supports weak mobility - the actor can be moved from one machine to another in the idle state (P. A. Bernstein et al., 2014).

The main features of an actor, namely, an event-based asynchronous thread, protected internal state's mailbox mechanism, transparent location, make it scale out from clusters and scale up to multi-core processors (De Koster, Marr, D'Hondt, & Van Cutsem, 2013). An actor is regarded as a container for behaviour, internal state, children, and supervisor strategy as shown in Figure 2.3.

The mailbox is utilised to store and deal with immutable messages that are sent from other actors who are in or out of the actor system (Hewitt, 2010). The behaviour of the actor defines the actions that are responsible for the matched messages. States in actor reflect the possible statuses that could be business logic, a set of listeners, the situation of HTTP requests, and so on. These state data is significant for actors. Thus, actors encapsulate the internal state's data and only share them with other actors through immutable messages (Haller, 2012). Each actor will split tasks into sub-tasks and delivery to its generated children shown in Fig. 4. Meanwhile, it will automatically handle failures (also called supervisor strategy) from these children for the sake of providing an environment of highly fault-tolerant systems shown in Fig. 5.

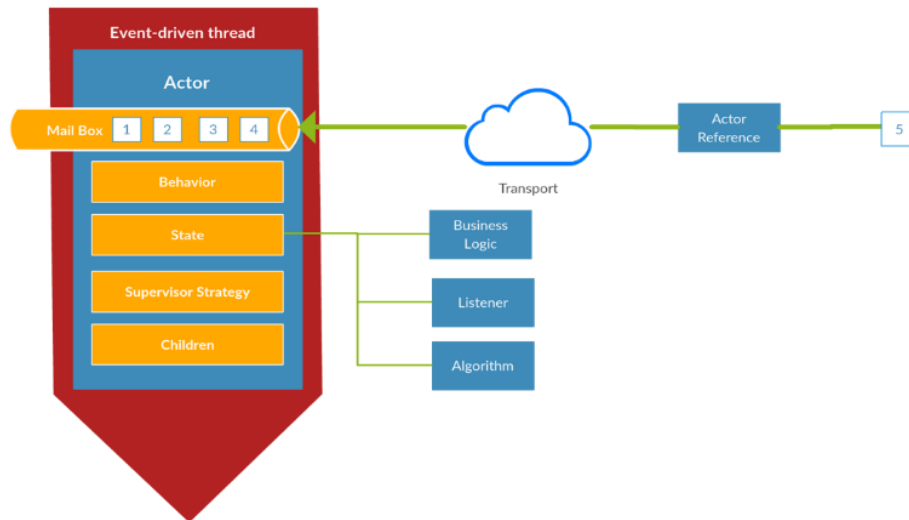


Figure 2.4 Understanding of an actor

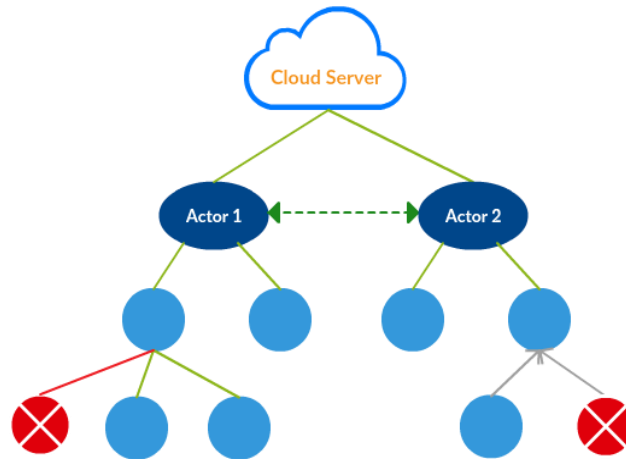


Figure 2.5 A supervisor and its children in a graph

2.4 Traffic Reliability in Online Auction

Traditional client/server architecture faces a severe problem of traffic reliability in an online auction. The server is responsible for processing all clients' bid data regarding each round in a live online auction system. Because of network delay, the clients' request could be late to arrive the server side. Thus, the data from clients' side will be dropped that will result in packet loss. This is one of the main reasons why online auction systems cannot be trusted. Bidders never know whether their bid has arrived safely on the server side or not.

A proposed solution to solve this problem is active protocol filters (H. Liu, Wang, & Fei, 2003). This protocol has a capability of filtering out low-bid data in the process of price delivery before they reach the server side. Then the active filters will reject the appropriate clients. Meanwhile, sending rejection notices to clients is for the sake of improving loading balance of the server side. Other online auctioneers use semi-enclosed auctions, which can lengthen the auction cycle. For example, eBay or Amazon provides non-real-time online auction method so that there is no need to control network delays of the online auction system. Obviously, performance and stability of the communications between online auction systems, such as cost, fairness, response time, traffic reliability and packet loss, have become the major concern for all parties involved in the online auctions (McAdam, 2001). Another mode (H. Liu et al., 2003) is for multicast-based online auctions. Although the multicast model can enhance the capability of packet delays and traffic rates, the issue of packet loss still cannot be resolved.

2.5 Auction Model

In general, most e-commercial auction services launched are based on disclosed method, like English auction or reverse auction. The eBay in the USA, the Alibaba from China, or the Trademe in New Zealand is the best example based on this point. These auction methods were developed and used in B2C, C2C and B2B domains. C2C-based English auction is the dominant mechanism in e-commerce marketplace (Chan, Ho, & Lee, 2001).

With the globalisation of Internet-based cloud services, the online auction platform is reshaping the market by providing a professional service. A key aspect of this shift is the provision of many services, especially professional services, which are no longer limited by the common location of buyers and sellers (Ackerberg et al., 2006). The research outcome Krasnokutskaya et al. (2016) indicates that 80% of online transactions are carried out by participants (both buyers and sellers) from foreign countries and regions.

Under these conditions, it is difficult for us to track the status of all bidders in the clustering of trade to ensure the fairness and reliability of online auctions. English

auction has set an excellent example which is the most widely used methodology on the Internet. We can divide the whole process of online auction into four steps (as shown in Figure 2.4). They are bidder invitation, online auction, financial clearing and product delivery. Meanwhile, this type of transaction has a very high demand for concurrency and responsiveness of a system. In a very short period, the system needs to complete the price comparisons, price updates, and real-time notifications of all bidders shown in Figure 2.5.

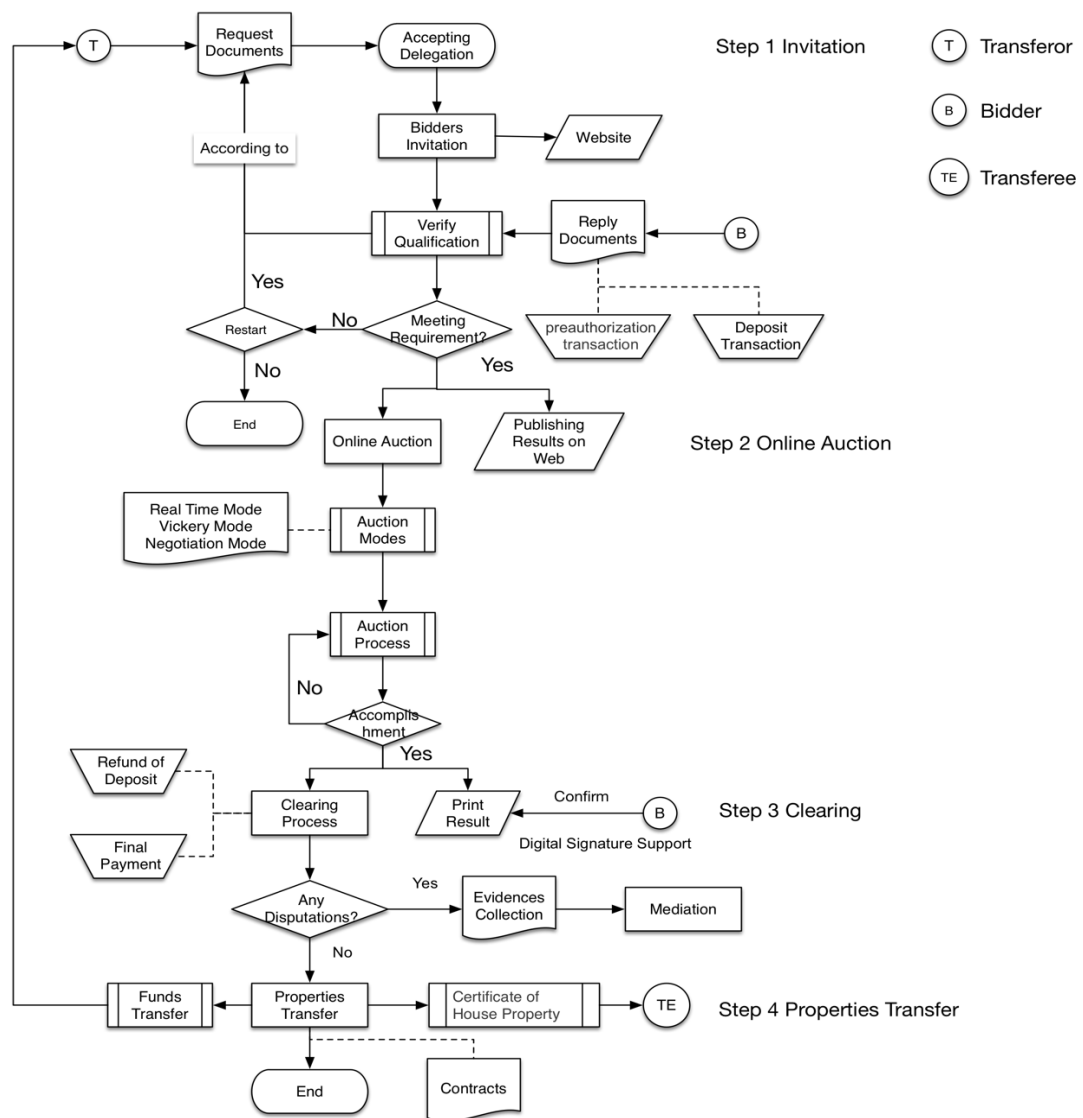


Figure 2.6 Overview of online auction process

Two important principles should be emphasised on cloud-based online auction: price priority and time priority. Bidding keeps increasing before the close time, and finally, the bidder holding the highest offer will win absolutely. Also, an English auction provides a given period to all interested bidders in each round. This ensures that buyers have enough time to think over and give their final responses shown in Fig. 5. However, we must take into account that the network delay can quite easily lead to several buyers in a few milliseconds of time to place the same price. For traditional online auction system, all quotes must be stochastically posted back to the main server for price and time comparison. Moreover, the system achieves the highest offer to the database in each round. If several buyers place the same highest offer, the system will compare the timestamps to choose the earliest bidder.

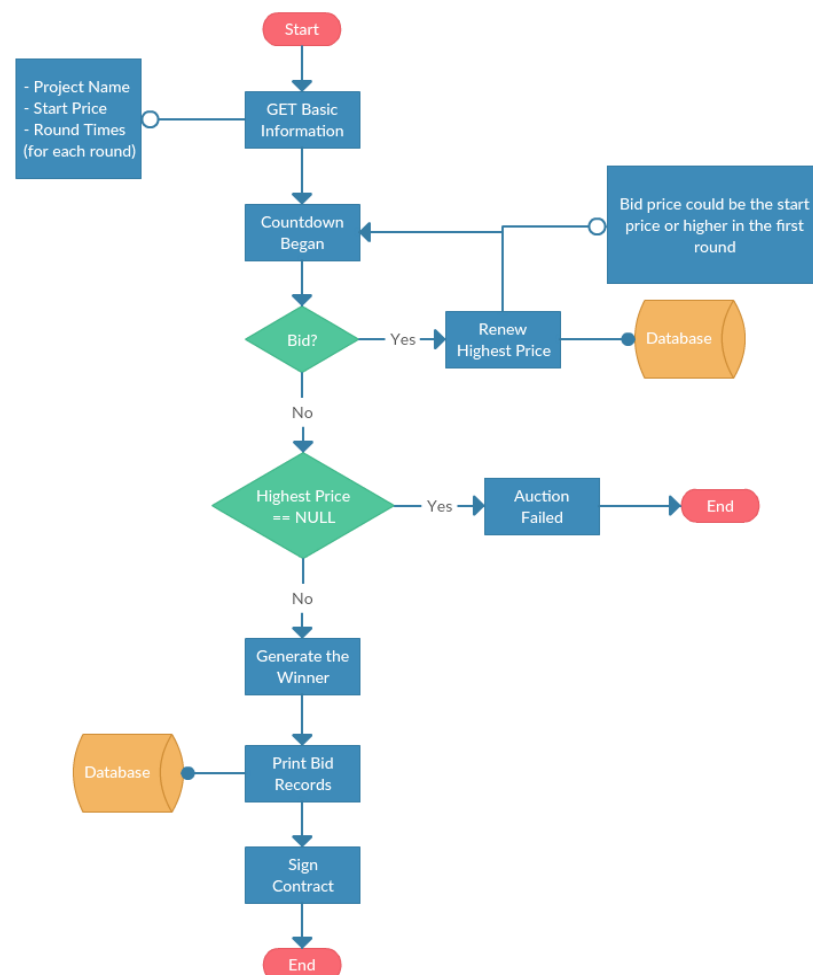


Figure 2.7 The workflow of real-time bidding

English auction allows sellers to set a start price, a reserve price, and bid increment. Reserve price (Kekre & Bharadi, 2011b) provides an insurance mechanism against low closing prices which means that if the final bidding cannot reach the reserve price, all bidders will be failed. Moreover, bid increment also plays a pivotal role in English auction outcomes. Although high bidding increment pushes bidding price up quickly (Karmani et al., 2009), it will result in reducing the participation.

Most of the online English auctions offer proxy bid service. It allows the buyers to place a maximum bid price via the service. Moreover, bidding will keep increasing until it reaches the proxy bid price (Chan et al., 2001).

The biggest challenge for applying this trading model to an online trading platform is whether we distribute the comparisons of the highest price and the timestamp to the other clusters. Also, it is tough to guarantee that the security of information exchange between clusters. Because of features of the distribution of clustering, information exchange on the Internet will be riskier than we expected. The existing signature verification technology has been quite effective for online message protection (Philip & Bharadi, 2016). For example, modified digital difference analyser algorithm is the best case in this point. It is used to capture dynamic characteristics of a signature in discrete values so that we are able to identify whether the variables in messages have been modified (Kekre & Bharadi, 2011a). Most of these solutions of security for a “live” online auction is too heavily. The encryption or decryption process will undoubtedly affect the efficiency of online auction systems. We are still looking for a more lightweight solution to solve the issue of information exchange, like actor model. This is also the main reason why we introduce the actor model to the online auction platform. The technology is also widely used in big data deployment and data mining to guarantee the efficiency of data processing (Estrada & Ruiz, 2016).

2.6 Blockchain

As the most significant electronic market, the online auction transforms the collectables business and e-commercial business to a global billion-dollar market (Corcoran, 1999). With the idea of frictionless economy, an online auction is considered to be a rapid and efficient platform to eliminate geographic boundaries and

establish the accurate price based on supply and demand (Ba, Whinston, & Zhang, 2003). Alibaba, Amazon and E-Bay are the exemplary cases in this area. These platforms offer low cost, excellent technical support, and massive data analysis so that traditional sales industry begins to transition to e-business model increasingly. The usage of online auction sites ranges from individuals to firms; however, the market fraud is rising. According to the Internet fraud operated by the National Consumers Union, online auctions were still the primary source of Internet fraud (Alanezi, 2016).

For this reason, a growing number of institutions, such as the Alibaba, Samsung, Louis Vuitton, Internet Fraud Watch (www.fraud.org) and the Internet Fraud Complaint Center (www.ifccfbi.gov), are jointly developing a new mechanism to combat this challenges (C. E. H. Chua & Wareham, 2004). Simultaneously, Alibaba and a myriad of financial institution proposed a new solution - giant distributed database blockchain. The blockchain is secured by Byzantine fault tolerance (BFT). This database is used to maintain a continuous growth of blocks. As shown in Figure 2.5, the design of blockchain is to prevent data modification. Once the data were recorded, it could not be altered reversely anymore.

In blockchain, each block consists of block hash link, timestamp, and valuable data as shown in Figure 2.6. The blocks can be transferred from one blockchain to another. This technology has approved that it can eliminate the double-spend problem (Pilkington, 2015).

Thus, it can be utilised to various scenarios, such as financial services and medical record (Gareth W Peters & Panayi, 2016). The blockchain might become the most critical online promising technology for Internet interaction (X., Li, Jiang, Chen, Luo, & Wen, 2017). This database is more magic than what we expected. It could be the final solution to the online auction fraud.

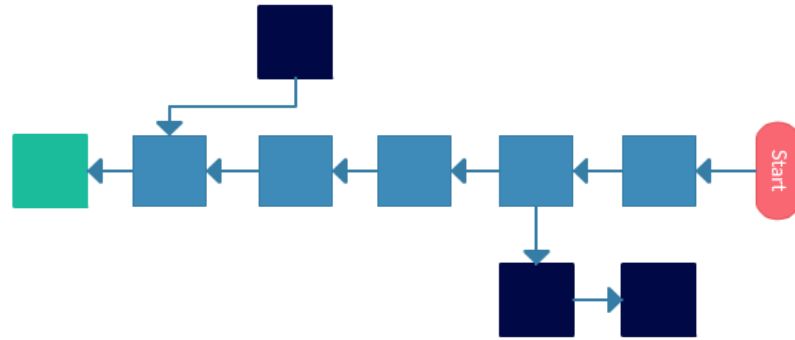


Figure 2.8 The diagram of blockchains

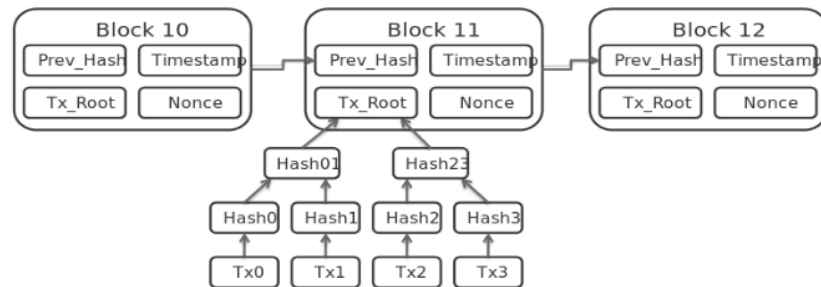


Figure 2.9 Bitcoin network (resource from bitcoin)

Based on the distributed blockchain technology, we create a broad range of distributed applications. The revolutionary methodology in this area is the Ethereum platform, which includes a complete programmable framework (English, Auer, & Domingue, 2016). This framework is utilised to implement the smart contract. Moreover, the blockchain infrastructure facilitates virtual currency, such as Bitcoin, trust and contract applications. The most significant feature is that the linked data are decentralised so that we do not need to be dependent on the central server anymore.

In this thesis, we discuss the methodology that can prevent the online fraud actively. The implementation of a blockchain can secure the currency transition of online action. We utilise the mechanism of smart contract and linked data which ensures the security of online payment from end to end. Moreover, final transactions are permanently

recorded in the database so that we are able to provide permanent records for all clients (Black, Hashimzade, & Myles, 2013). Finally, the Elliptic Curve Digital Signature Algorithm (ECDSA) is implemented in blockchains for offering trusted handshakes between blocks (Johnson, Menezes, & Vanstone, 2001).

Blockchains are considered as one of the most promising technologies in the next generation of Internet transaction systems. It can be used not only for current financial services such as digital asset management, deposit certificates, remittances and online payment systems but also for smart contracts, Internet of Things (IoT), reputation systems and security services (Gareth William Peters et al., 2015), etc. In order to provide a secure environment for virtual currency transactions, Bitcoins (BTC) utilise blockchain to reinforce its cryptocurrency.

Specifically, the blockchain facilitates the elastic and distributed ledger for storing a significant amount of transactions, attributing them to a block in the network and ordering these blocks in real time (English et al., 2016). As the number of blocks proliferates, generating new blocks requires significant resources. So many private organisations join the competing with each other for utilising dedicated high-power computers to run ASICs for the real-time process of Bitcoin networks (Kroll, Davey, & Felten, 2013). This process is also called mining as shown in Figure 2.7. Once the miner successfully finds a new valid block on the longest branch of the branching tree, a new transaction will be added to the log and its nonce is selected. Meanwhile, invalid blocks will be ignored. This mechanism is called proof-of-work (POW), which is considered difficult to be performed, but the result is easy to be verified (Becker et al., 2013).

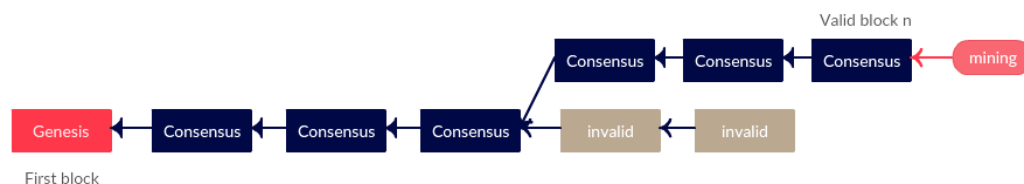


Figure 2.10 An example of mining process in the blockchain

Blockchains work in a decentralised and untrusted environment through the integration of encryption hash, consensus mechanism, and digital signatures (based on asymmetric encryption). Figure 2.11 shows the details of a branch of the blockchain. In the blockchain, each client fully participates in all operations, such as initiating new transactions, receiving transactions, validating transactions, and the generation of new blocks.

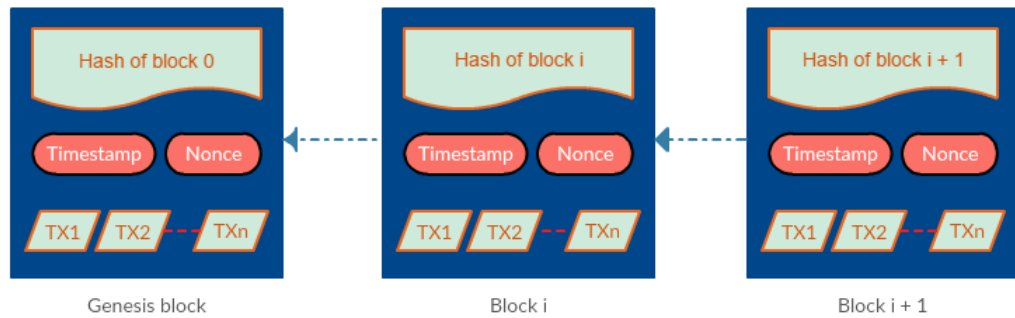


Figure 2.11 The details of a branch of blockchain

2.6.1 Block

To prevent online fraud, such as double spending, all clients must participate in a peer-to-peer protocol that implements a distributed timestamp service, and provide a fully-serialized log which contains the details of all transactions. The transactions in the log are formulated into blocks that contain block version, serial numbers, timestamps, encrypted hashes, a nonce, metadata, and a set of valid records of the transaction. A single block structure is shown in Figure 2.9, which is presented by JavaScript Object Notation.

Specifically, a header of the blockchain illustrates which block validation rule should be followed. The previous block is a 256-bit encrypted hash number. The difficulty is utilised to indicate the coefficient of difficulty for mining the new block. The timestamp is a big integer that shows the current timestamp for the world UTC since 1970. A nonce number is a 4-byte field. This field usually begins from zero, and each time the block is hashed, this field is incremented. The maximum amount of transactions, which are stored in the single block, entirely depend on the size of the blocks and the size of each record.

The block body contains a number of transaction records and the set of transaction logs. A credible record of the transaction consists of two main components: output and input. The identity section of records of transaction inputs is a significant reference to the unspent transaction outputs (UTXO) of the sender while the sender must have to designate the destination address and numbers in outputs. Also, clients or miners are able to validate the digital signature so as to ensure the transaction is entirely validated. Asymmetric cryptography mechanism (Shown in Figure 2.10) is utilised in blockchain to verify the authentication of each transaction for the sake of ensuring the authentication, data integrity, and repudiation. The trusted third party provides the digital signature based on asymmetric cryptography is widely utilised in an online market (Christidis & Devetsikiotis, 2016).

```
{
  "hash": "893a09b5a19838027b60f8007876bcf4333d4904",
  "prev_block": "90f8b1f196c97b60c760f2d18601bc29bc5fe23a",
  "time": 1491973298603,
  "difficulty": 3343090408,
  "nonce": 2504324516,
  "tx": [{ "hash": "YXNkZmFzMjM0",
    "in": [{
      "prev_out": {
        "hash": "000..."
      }
    }],
    "out": [{
      "value": "74.76710074",
      "scriptPubKey": "36a1..."
    }]
  },
  ...
}
```

Figure 2.12 A block in blockchain that is represented by JSON

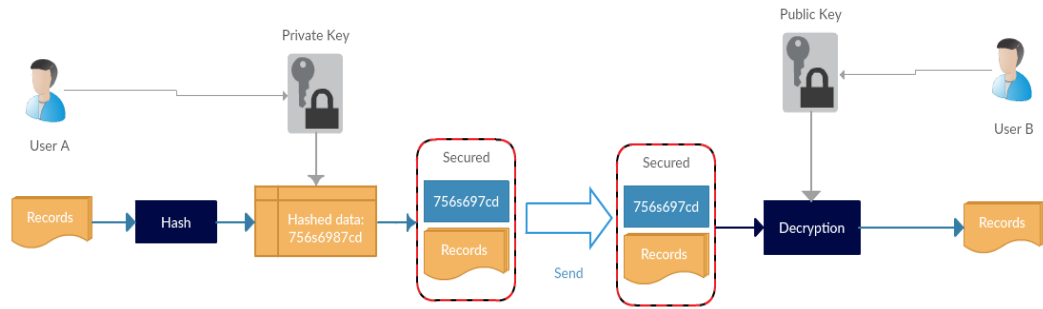


Figure 2.13 Digital signature in blockchain technology

In online auctions, massive individuals participate in transactions. This situation makes it tough to bind an identity to a participant. Naturally, blockchain facilitates to solve this problem quickly. The absence of interpersonal interactions and communications-based authentication in electronic markets is another issue that leads to online frauds. In a traditional business, the buyers and sales have got used to the face-to-face business at the same place. Customers and vendors establish their basic trust via conversation, handshake, facial expression, eye contacts, etc. (Ba et al., 2003). The in-kind transactions provide opportunities to know the quality of products by viewing and touching. Thus, due to the existing problems in an online auction, we will demonstrate how the blockchain solves these problems by using decentralise framework, reputation system and smart contract in the next sections.

2.6.2 Decentralized Framework

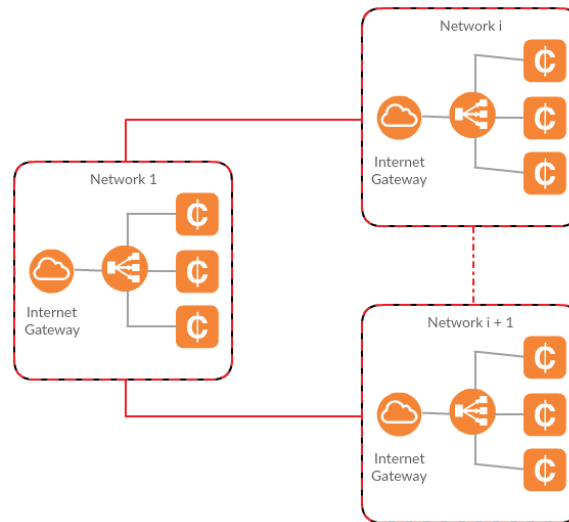


Figure 2.14 Decentralized network

Once the records of transaction or block are generated, the validation mechanism will be triggered immediately. The new node needs to be verified by the network. However, as shown in Figure 2.11, there may be a profoundly distrusted network of the blockchain, because each branch may have a view which is different from the entire network state. Thus, a decentralised network is needed to diminish the branches in the blockchain.

In a distributed blockchain database, one of the most challenging issues is how to reach consensus block on a transaction among the invalid blocks. Each miner communicates with the blockchain database through a dedicated node in which a client is implemented. Many blocks across the network form a decentralised network. Once a block receives transaction record from another customer, it is going to verify the authentication of the block first. Moreover, the result of validation will be broadcasted to every valid block connected to it. Thus, the valid block will be rapidly spread throughout the whole network. The advantage of the decentralised systems is the independence, which is able to save the cost of the enterprise. However, the decentralised network also increases the difficulty in verifying the transaction record in the untrusted environment.

2.6.3 Smart Contract and Smart Property

Programmable electronic smart contract conceptual concept can be traced back to nearly two decades (Szabo, 1997). From the very beginning, the smart contract and smart property in Blockchain 2.0 are mainly utilised to solve the critical issue from generic identification and authorisation, whereas the Blockchain 1.0 focuses on virtual money transaction, the Blockchain 2.0 is for decentralising the market generally, and the implementation of a blockchain beyond the currency (Kosba, Miller, Shi, Wen, & Papamanthou, 2016). To be more specific, the technologies from Blockchain 2.0, such as smart contract, smart property, decentralized applications (Dapps) and decentralized autonomous organizations (DAOs) are implemented in the application layer that provides to the software developer a tightly integrated end-to-end platform, such as Ethereum (Altcoin project), for building blockchain-based software (Wood, 2014a).

The essential functionality of decentralised transaction ledger in Blockchain 2.0 is utilised to register, approve, and execute all types of contracts and properties. Table 2.1 illustrated the properties; the contracts could be transferred with the blockchain.

Table 2.1 An example of blockchain applications in version 2.0

#	Category	Properties
1	Public records	Land and properties title, death certification, business registration data, and vehicle registration data.
2	Private records	Personal contracts, digital signature, individual loans and marriage certifications.
3	Identification	Passport, ID cards, driver license.
4	Documents	Notarized documents, property ownership Certifications, and proof of insurance documents
5	Financial records	Deposit, bank statement, private equities, stock records, bonds, pensions.

#	Category	Properties
6	Intangible assets	Copyrights, intellectual property, domain names, and reservations.

Any forms of assets can be registered in blockchain database, once these properties are encoded into the blockchain, they become smart properties, such as public records, private records, identification, digital documents, financial reports, and intangible assets, like stock shares, copyrights of music and books. The fundamental idea of the smart property is controlled by an owner who has the private key (Swan, 2015). By using the smart contract, the ownership of smart property is able to be transferred automatically to another owner after all payments are made. The execution of this smart contract is entirely automatic; there is no need to be interacted by human operations when the authentication and verification are successfully processed.

Delmolino et al. (2016) proposed that a smart contract is executable code. For example, we can write a smart contract on Ethereum by using Solidity language. These codes run on the Ethereum's blockchain. A smart contract mainly includes code, storage files, gas, and account balance. Any user is able to create and publish their contract on the blockchain. Once the contract is published, it cannot be changed anymore. Also, another type of the smart crypto contract also has been discussed. Compared with the original smart contract, only the participants of the contract are able to access the sensitive messages like transactions records in the contract. Several programming frameworks have been implemented within high-level programs as specifications and encryption, which include the secure multiparty computation of contract compiler (Kreuter, Shelat, Mood, & Butler, 2013) (C. Liu, Wang, Nayak, Huang, & Shi, 2015; Rastogi, Hammer, & Hicks, 2014), and zero-knowledge proofs (Fournet, Kohlweiss, Danezis, & Luo, 2013) (Fournet et al., 2013). L. Zheng, Chong, Myers, and Zdancewic (2003) explains how to build secure distributed protocol, such as a sealed-bid auction, battleship games and banking applications.

A smart contract's storage file will be published to the blockchain with the logic code together. When another user executes the contract, a consensus is reached on the

outcomes of the implementation. Meanwhile, the blockchain is updated accordingly. For example, when a bidder tries to join an online auction on our system, he/she will trigger the smart contract automatically. The contract will check the bidder's account balance and transfer a deposit directly into our system. If the bidder does not have enough balance, the application will be rejected. While, the deposit will be automatically refunded to the bidder if he/she does not win on the system.

Also, the contract's code is able to be executed either by a user or by another contract. For instance, if a bidder wins the auction, he/she will trigger a smart contract and transfer the money directly to the seller. At the same time, the smart contract will notify another smart contract to transfer ownership to the bidder. The information exchange between smart contracts includes message data and contract address. While executing its code, the smart contract will read significant information from its storage file, like contract address that can be used to collect money or write the transaction records into its storage file. Because the contract has its address, so it is able to receive money into its account balance as well and transfer the funds to another contracts or users.

The smart contract allows us to perform general calculations on the blockchain. However, they are good at managing data-driven interactions between entities on the network (Industries, 2016). Let's explain this one with an example. Consider Alice, John and Tony in a blockchain network, as well as X and Y-type digital assets are trading. John has deployed a contract on the blockchain that defines:

- (a) The deposit function allows him to deposit X units into the contract;
- (b) The trading function sends back an X unit from the contract itself for every five Y units received;
- (c) The withdraw function allows John to withdraw all the assets of the contract.

Note that the deposit and withdrawal functions are only executed by John (through his private key) that can be called because it is John's decision; once John releases the smart contract on the blockchain, any user on the network can successfully call functions on this smart contract.

John sends the transaction to the address of the smart contract, called the deposit function, and transfers the three units of X into the smart contract. If the transaction reaches a consensus between transaction nodes, the transaction will be stored permanently on the blockchain. Alice owns 12 units of Y, and then sends a transaction that moves ten units of Y to the trading function of the contract and returns two units of X. Before Alice trigger the trading function, the blockchain will check the balance of X unit on the smart contract. This transaction is recorded on the blockchain as well. Then, John sends the signed transaction to the withdrawal function of the contract. The contract checks the signature and ensures that the withdrawal is initiated by the contract owner and transfers all of its deposits (1 unit of X, ten units of Y) back to John.

According to the example above, we can make few concepts clearer:

- (1) The smart contract has its public address and state, which are able to take custody of digital assets on the blockchain (Clack, Bakshi, & Braine, 2016). Meanwhile, the smart contract supports the account-based model. In the instance above, it is able to hold the X and Y digital assets.
- (2) We are able to contain the business logic inside to the smart contract. For example, when a user buys 1 unit of X, they need to pay five units of Y.
- (3) The smart contract is triggered by the transaction messages that send to its public address.
- (4) John hopes that the relationship with the counterparties is a data-driven model. A transaction is signed by the valid key-pairs, which demonstrates a transfer of value(Antonopoulos, 2014). John implements a smart contract on the blockchain, which defined the trading rule like if one of the users transfers five units of Y asset into the smart contract address, it will return 1 unit of X to user's wallet.

Conceptually, we can think of a contract as a particular “trusted third party” – however, this party is trusted only for correctness and availability but not for privacy. In particular, a contract's entire state is visible to the public. Whenever a message is received, the protocol code will be called. A contract can define multiple execution entry points in Ethereum's Solidity language; each entry point is defined as a

function. The message content will specify the entry point where the contract code will be invoked. Therefore, the message is like a function call in a standard programming language. After the contract completes processing the received message, it returns the return value to the sender (Christidis & Devetsikiotis, 2016).

2.6.4 Properties Exchange on Blockchain

To explain how the smart properties are transferred on the blockchain, the best way to imagine a decentralised banking network that tracks the aggregate balances and transaction records for each client. We are primarily having a table style record for each user. The table has three columns: “asset type”, “owner/counterparty(Gendal Brown, 2015)” and “amount”. Once the property is registered into the blockchain, a permanent record will be marked. For example, a row of the table will indicate that “NZD”, “Alice”, and “10000” identify Alice’s NZD balance has ten thousands New Zealand dollars. Bob has an account with a hundred NZD. If Alice tries to transfer like \$10 to Bob’s account, we can imagine that the “amount” of the Alice row will get an update to \$9990, and the balance of Bob’s account will increase to \$110. In this case, the NZD is able to be the digital asset.

We can use a blockchain network with an Ethereum currency transaction model to easily transfer digitally tagged assets in an encrypted and verifiable way. Please consider again the model of a decentralised database which is shared in an entirely trustless environment. Each row carries the same properties fields, the only difference that the “owner” field now holds the public key. Thus, everyone who knows the public key is able to transfer the money into this account. Alternatively, it can be seen as Alice owns ten thousands of units of an asset of something. For example:

Table 2.2 A record of Alice’s account

Asset type	Owner (Alice)	Amount
X	0x05e93e78eaf49be440f05db15f470a17a1c47257	10

As shown in Table 2.2 , a row in the database, carries a kind of asset X, it is public

key is “0x05e93e78eaf49be440f05db15f470a17a1c47257” and the total amount is 10. We assume that Alice knows Bob’s public key. So, she can transfer asset into Bob’s account. To be more specific, she signs a transaction to transfer two units of X from her account to Bob’s account. We see what is happening below is that the blockchain will generate a Tx records. The Alice’s account will increase a new record which indicates that two units of X have been removed from her account. Meanwhile, the units of X asset in Bob’s account have updated from 0 to 2. This is done in order to control concurrency and prevent conflicts between concurrent write operations in the system; rows are not modified but are increased, and new rows are created with updated values (Greenspan, 2015)

Table 2.3 A changed record of Alice’s account

Asset type	Owner (Alice)	Amount
X	0x05e93e78eaf49be440f05db15f470a17a1c47257	10
X	0x05e93e78eaf49be440f05db15f470a17a1c47257	8

Table 2.4 A record of Bob’s account

Asset type	Owner (Bob)	Amount
X	0x05e93e78eaf49be440f05db15f470a17a1c34527	0
X	0x05e93e78eaf49be440f05db15f470a17a1c34527	2

Now, we are able to see Bob’s new balance above. There are two new assets “X” shown in the database record. During the transaction process, there is another milestone – validation, must be discussed.

We will encode the verification checks in the nodes of the block-chain network set

for this asset transfer will be to:

- Check the transaction address does exist.
- The transaction is properly signed by using the private key.
- Check the transaction is not a duplicated operation (preventing an asset to be spent twice).
- Check the transaction amount to the new row is correct. Because the blockchain database keeps all transaction records, so it is very easy to retrieve the transaction history and check the transaction amount. For example, if the Alice transfers two units of X to Bob, but the balance of Alice reads 9 units of X, the transaction will be failed.

We need to mention that a transaction can resolve several existing rows instead of just one; that is, to transfer the assets to the database, as long as it is properly signed to access them. These existing un-inserted rows are called unspent transaction outputs (UTXO) in Ethereum; they are created by earlier transactions in the Ethereum. Transactions consumed by UTXO are called transaction input; transaction creation UTXO is called transaction output (Antonopoulos, 2014). Then, the transaction basically removes a set of rows (UTXO) and creates a new row (UTXO) in the database.

One of the outstanding questions in the above description is how we generate assets and introduce them into the blockchain. Before we arrive at 10 X units of Alice, the 10 X units must come from somewhere. The answer is that it depends on the organization or starters who create the private chain or public chain. Typically, properly authorized nodes use special types of transactions to bring assets (or new units of assets) into the network.

For example, assume a private blockchain network, which is set up by Tony. He sets this up using the Ethereum system. His role is the system administrator of the private blockchain. Tony has the privilege to issue assets on the network by using his private key. He invites Bob and Alice join in; both of them have reached an agreement

that allows the Tony to issue assets on the blockchain. In order to ensure the system secure, they have already set three networks in different places to save and sync their data on the blockchain. Meanwhile, all of them have a pair of private and public keys. Once all clients are invited into the blockchain. Tony creates a signed transaction to generates 10 units of X. This transaction result will spread to all networks. All these nodes on the network will consider that this transaction record is valid since Tony's key pairs are permission. In next step, Tony transfers all these units of X to Alice which will result in Alice's account will be topped up 10 units of X. In the case of Ethereum, new coins are generated into the network with every mined block. The coin-based transaction will be included in the mining node in the block of transactions. The mining result will be broadcasted on all networks (Franco, 2014). The coin-based transaction has not inputted and the mining node is rewarded with the number of Ethereum (through the network).

There are several things we should bear in mind: if we have a group of users who want to trade digital tokens and have agreed on how these tokens are generated, then the blockchain network is the ideal tool for exchanging these tokens and tracking the ownership of these tokens. There is no need for intermediaries to facilitate tokens exchange, results from each node on the network will carry out the necessary checks and agree on the results of the acceptance. Asset tracking is out of the box because each node can access a set of identifiable encrypted transactions on the block.

2.6.5 Tradenet

With the rapid development of Bitcoin, we gradually realise that the practical value of blockchain is far more than the electronic money system, especially in the auction market. All deals are able to be automatically transacted based on specific permissions and contract(Swan, 2015). The real-time bidding (RTB) networks are the best case in existing examples of automatic markets.

With the rise of online auctions in future, the blockchain is applied to restrict the order and plan transactions for resource allocation of our real world. An advanced concept is that the self-operation system (integrated with blockchain) will be implemented for the management of those self-owned assets like house, private stock,

and self-owning car. Once these assets are registered into the blockchain, they will become self-directed assets (smart properties). These properties are able to employ themselves for automatically trading continuously connected to the Internet so that they can query significant data and search for the potential transferee (Pagliery, 2014). If the conditions of potential customers meet the requirement of smart properties, they are going to execute the smart contracts to finish the transaction. This is a significant step of the distributed online auction.

2.6.6 Challenges of Blockchain

Blockchain has a great capacity for establishing the future Internet interaction systems, but it has to face technical challenges. To be more specific, there are further needs to be considered in the current blockchain for the sake of meeting the requirement of real-time processing of billions of transactions. Additionally, there is a need to propose a new mechanism to avoid selfish miners in blockchain (Eyal & Sirer, 2014). Meanwhile, before the blockchain is widely applied to various Internet interaction, other challenges also need to be addressed, such as the lack of privacy and current consensus algorithms (Kosba et al., 2016). For example, the entire sequence of events that will be executed in the smart contract will broadcast to the whole network for validation and mining. However, their public key is visible for anyone result in all transaction records and clients' balance are exposed. Ron and Shamir (2013) and Meiklejohn et al. (2013) pointed out that hackers are using this public information to analyse the transactional graph structures of cryptocurrencies, and then, implement the deanonymization attacks.

We emphasize that the lack of privacy is a major obstacle to the widespread adoption of decentralized smart contracts because financial transactions (such as insurance contracts or stock deals) are perceived by many individuals and organizations as highly confidential. Despite the progress made in designing privacy to protect encrypted currencies such as Zerocash (Sasson et al., 2014) and others (Danezis, Fournet, Kohlweiss, & Parno, 2013), (Sasson et al., 2014), (Bonneau, J., Miller,

Clark, Narayanan, Kroll, & Felten, 2015). These systems give up programmability. Meanwhile, we do not know how to implement programmability in advance without

exposing the transaction records and data to miners in cleartext. The last and the most important is to ensure the authenticity of privacy rather than the nearest decentralized, encrypted currency, most of the smart contract implementations rely on the security of trusted servers (M. S. Miller, Morningstar, & Frantz, 2000).

2.6.7 Hawk

Kosba et al. (2016) proposed a framework called Hawk for establishing the privacy protections of the smart contract. With Hawk, non-professional programmers can easily write a Hawk program without having to perform any encryption. The Hawk compiler is responsible for compiling the program into a blockchain and an encryption protocol between users. As shown in the Figure 2.15, Hawk program consists of two parts:

- (1) A private part is represented as a party with input data from parties and currency units (such as bids in auctions). Private part performs a calculation to determine the payment distribution between parties. For example, in the auction, the winner's bid is transferred to the seller, and the other bidder's deposit will be refunded. The private Hawk program is aimed at protecting participants' data and money exchange.
- (2) The public part is expressed as a non-contact access to private information or money. The compiler of Hawk will translate the program into the following pieces, which together define the password protocol between the user, the manager, and the blockchain:
 - An executable programme for all consensus nodes on the blockchain;
 - An executable programme for users;
 - An executable programme for managers;

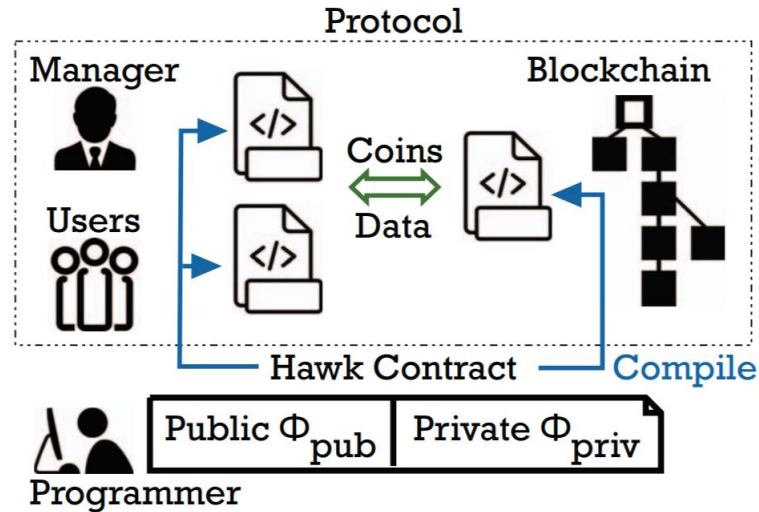


Figure 2.15 Hawk overview

The security guarantees: The Hawk's security guarantees consist of two aspects:

(1) On-chain privacy

On-chain privacy regulations provide trading privacy to the public (i.e., breach of any party to the contract) - unless the contractor itself voluntarily discloses the information. Although in the Hawk protocol, the user exchanges data with the database and relies on it to ensure the fairness of the suspension, the flow and amount of traffic in the private Hawk program is invisible from the public regarding encryption. Informally, this is achieved by sending encrypted information to the blockchain. Relying on zero knowledge proves the implementation of contract execution and financial protection of the correctness.

(2) Contractual security

While the on-chain privacy protects the privacy of the parties and prevents the security information to be released to the public (i.e., against anyone who does not participate in the parties of the financial contract), the contract protects the parties to the same contractual agreement from both sides of the contract. In the real business environment, we believe that the contractors always act selfishly to maximize their economic interests. Especially, they are able to forthwith deviate from the prescribed agreement, or even prematurely. Therefore, the contract

security is a multi-faceted concept, not only contains the confidentiality and authenticity of the encryption concept but also includes cheating and suspension of the existence of financial fairness.

(3) Minimally trusted manager

The implementation of the Hawke contract is facilitated by a manager (who deploys the contract within the blockchain). The manager can see the user's input and maybe not go to expose the user's private data to the third part. However, the manager is not entirely equal to the trusted third party - even if the manager is able to sidestep the agreement or connive with the other parties arbitrarily, the manager is not able to affect the correct execution of the smart contract. Kosba et al. (2016) believed that the manager will be financially punished if the manager breaks off the agreement. The users will receive compensation. Also, he also pointed out that the manager should not be trusted to preserve the security or privacy policies effect on the underlying currency.

Moreover, if multiple contract instances need to execute concurrently, each contract must be assigned and executed randomly by the different manger. So, the manager does not have the privilege to retrieve the smart contract from blockchain network. Technically, if they do not know the address of the smart contract, they are not able to assign the specific smart contract from the blockchain network.

Before we continue to explain hawk's smart contract, we need to clarify the concepts to avoid confusion. In Ethereum (Wood, 2014b), the protocol is called the Ethereum contract. However, in the paper of Hawk, the completed protocol is considered as the Hawk program which defines the contract; the programme of blockchain is a component of the more important protocol. Just in case that the managers break off the protocol, they will be financially punished so that the users will receive compensation.

Cryptographic protocols of Hawk: The Hawk's cryptographic protocols are able to be broken into two pieces:

- (1) The private cash. In this part, the Hawk smart contract focused on the implementation of direct money transfer between participants. The construction of this smart contract utilises the Zerocash-like protocol (Garman, Green, & Miers, 2016) that is used for the implementation of private cash and currency transfers. The spender needs to prove that the generated coins are constructed correctly by using the zero-knowledge proof:
 - a. The spent coins are part of existed private pool coins.
 - b. No double-spending issue.
 - c. The total value of the input coins must be equal to the withdrawn coins during the process of money conservation.
- (2) Binding the transactional privacy with smart contract logic. Once the currency is transferred into a private pool of coins, the blockchain is responsible for maintenance of the exchanges of private coins between users.
 - a. Using the frozen operation to freeze the direct money transaction between clients instead of committing the amount of money and an accompanying private into the smart contract.
 - b. Minimally trusted manager performs the computation of a proof of correctness and the pay-out distribution.
 - c. Hawk programme verifies the proof of the correctness and redistributes the frozen coins. If the verification failed, the money would be refunded to clients

2.7 Online Fraud

The seriousness of auction fraud is far beyond our imagination. Victims do not have to participate in the Internet auction but will suffer the consequences of fraud. The triangulation can be used to implement an offline fraud via a merchant sells the product from the online auction (Chua & Wareham, 2004). For example, when a thief uses stolen money to buy valuable products and put them online for auction, anyone could

win the auction. However, once the fraudulent purchase is restricted and confined, the buyers will lose the money paid for these stolen products. Meanwhile, all stolen products will be detained from the winners.

Information asymmetry has been recognized as the major issues that result in cheating quickly over the Internet (Choi, Stahl, & Whinston, 1997). This results from that two parties do not share the same data in business timely (Ba et al., 2003). For instance, trading partners use anonymous identities or buyers cannot acquire the real data on the quality of products. The main reason is that online market lacks interpersonal interactions and communications. By contrast, in the traditional business environment, both sides of a business establish their initial trust via physical contacts like hugging, eye contact, and a handshake. Meanwhile, buyers get to know the quality of products by touching, looking, or even tasting. However, these cases do not happen in the online auction based on e-commerce.

In order to reduce the fraudulent transactions in the online auction, the intermediaries have offered various services, like reputation system, feedback system, insurance or guarantee, and certification authorities (CAs). Besides the Class 4 of CAs can provide the maximum level of trust and assurance by thoroughly investigating companies and individuals, most of the anti-fraud mechanisms are passive defence (Froomkin, 1996). Literatures have shown that trading partners heavily rely on their reputations in the traditional business. Specifically, individual's reputation can act as a "hostage". The disrepute always spreads quickly in the business community. The social pressure or lost trusts caused by bad reputations might be more effective than legislation in this online community (Kekre & Bharadi, 2011b). Because the online auction sites rarely provide the strong authentication at present. Thus, those disrepute traders may renew the identity by re-registering a new user ID (Ba et al., 2003) to get a new reputation.

Table 2.5 provided several frauds of online auctions; the triangulation is just one of them (C. Chua & Wareham, 2002). Moreover, there are variations of online frauds. For instance, escrow services fraud is a variant of failure to ship. The escrow services are provided by the trusted third party, like Alipay, which is responsible for holding the

transaction funds from the buyers before the deal is successfully accomplished. However, the cheaters are able to set up a fake escrow service after receiving the funds. In order to prevent this type of fraud, auction houses establish a reputation system to mark each trader, like Alibaba, Amazon, and E-bay.

Bidders not only can view the reputation score itself, but also can find the number of positive, neutral or dissatisfied transactions as well as the comments (Ba & Pavlou, 2002). However, the reputation score system is rarely implemented based on the high-priced consumable market like artwork auction even if the seller really hard tells the fakes from originals.

With the globalisation of Internet-based cloud services, the online auction platform is reshaping the market by providing a professional service platform. A key aspect of this shift is the provision of multiple services, especially professional services, which are no longer limited by the location of buyers and sellers (Ackerberg et al., 2006). The research outcome (Krasnokutskaya et al., 2016) indicates that 80% of online transactions were joined by participants (both buyers and sellers) from different countries and regions.

Table 2.5 Types of online fraud

#	Fraud Types	Descriptions
1	Failure to ship	Never ship the product after payment.
2	Failure to pay	Buyers do not send the money to the seller.
3	Misrepresentation	Seller describes items incorrectly that do not match real items
4	Loss or damage claims	Buyer claim the loss or damage services to retrieve money back.
5	Shilling	Seller uses another account to bids on own stuff to push up the prices.
6	Triangulation Fraud	Sell stolen items online.
8	Buy and switch	Buyer switches the original sound with inferior one and returns it to the seller.
9	Shell auction	Seller set up a fake auction to store the credit card information from the buyer.
10	Bid shielding	Two or three bidders collude on an auction.
11	Fee stacking	Seller asks the buyer to pay extra fees after auction ends.

Under these conditions, information asymmetry illustrates a specific situation that two parties do not share the same information; it has been regarded as the majority of issues in the electronic markets (Akerlof, 1970). There are two most important aspects related to online frauds: one is the anonymous trading; the other is the unwarranted products or uncertain quality of goods (Choi et al., 1997).

Obviously, the blockchain and smart contract provide a new perspective for us to address the online fraud issues. The working mechanism of blockchain and smart contract is built on a untrust environment. All the smart properties registered on the blockchain need more than two thirds of the nodes to reach a consensus so as to ensure the authenticity of the information. In addition, all smart properties' transaction records and ownership changes are traceable. It would be very difficult to put stolen goods on blockchain because it is impossible for the seller to provide the original material to prove its ownership. If these goods have a special serial number, the stolen is not able to register the goods on the blockchain.

Chapter 3

Methodology

The main content of this chapter is to clearly articulate research methods, which satisfy the objectives of this thesis. The chapter mainly covers the details of research methodology for actor framework, elliptic curve cryptography and blockchain architecture. The actor framework is employed to provide the data consistency and integrity during the high-performance network environment. Meanwhile, ECDSA in the blockchain is able to provide the mechanism of zero-knowledge proof and a distributed ledger. Finally, the blockchain architecture illustrates the methodology of implementation of our private blockchain.

3.1 Introduction

In this chapter, we will articulate the design of architecture of our online auction system. As shown in Figure 3.1, when a bidder bids for the price (signed price data by their private key), the bidder actors will fetch the timestamp from TimeAsync actor, and then send the price data to comparison actors. Once the price is settled, the winner's data will be written into the blockchain via JsonRpc provider.

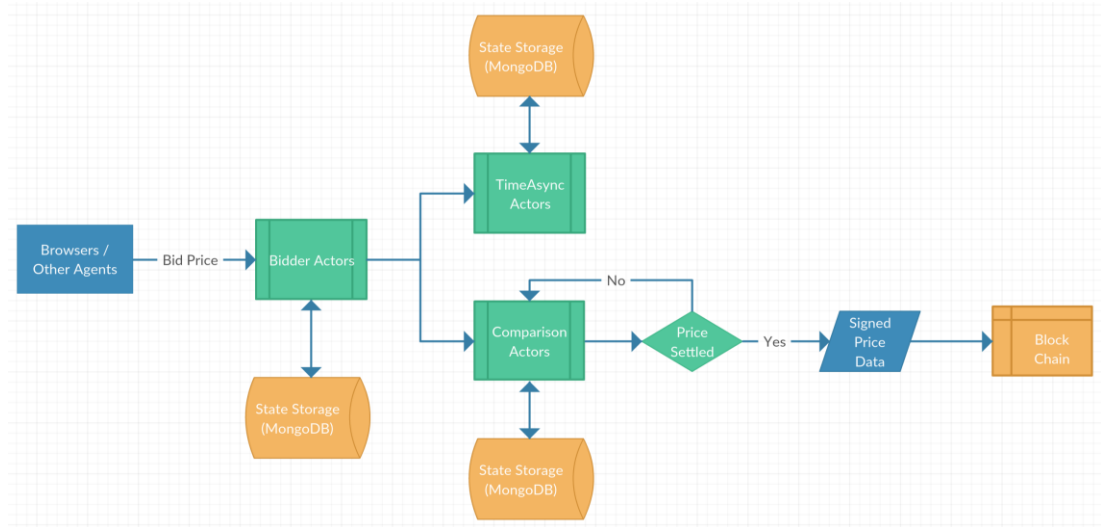


Figure 3.1 Overview of Online Auction System

To be more specific, we will explain how to design the module of bidder actors, TimeAsync actors and comparison actors from Chapter 3.2 to Chapter 3.3. The principle of signature of price data will be demonstrated in Chapter 3.4. Finally, we illustrate the design of the architecture of our private blockchain nodes.

3.2 Design

3.2.1 Immutable Message

When we instantiate actor pattern in code, actors become essential building blocks of an application. They are also a unit of isolation and distribution. Each actor has its unique identity consisting of its type and primary key (a 128-bit GUID). An actor will encapsulate its behaviours and internal state. Meanwhile, the state can be held by using

the built-in persistence facility which means the core of actor is isolated, that is, they do not share state and memory. Thus, the two actors or actor systems can only interact asynchronously by sending an immutable message (Gupta, 2012).

With regard to features, the bidder actor encapsulates its quotation and timestamp in the message and pass it to comparison actor in the actor system. After the comparison actor evaluates all messages from bidder actors, it will deliver the highest price from its actor system to another system for price and timestamp estimation shown in Fig. 3.1. Once the comparison actor achieves the final price, it will notice all bidder actors update the highest price in the actor system.

In this way, figuring out the highest offer will be hierarchical. First, the filter of the highest quotation starts from each actor system, and then gradually spreads the message to another actor system. After the comparison actors receive the final price, they will push it to bidder actors in the actor system instead of broadcasting the result to all clients. Due to the hierarchical feature, this pattern improves the network work balance efficiently.

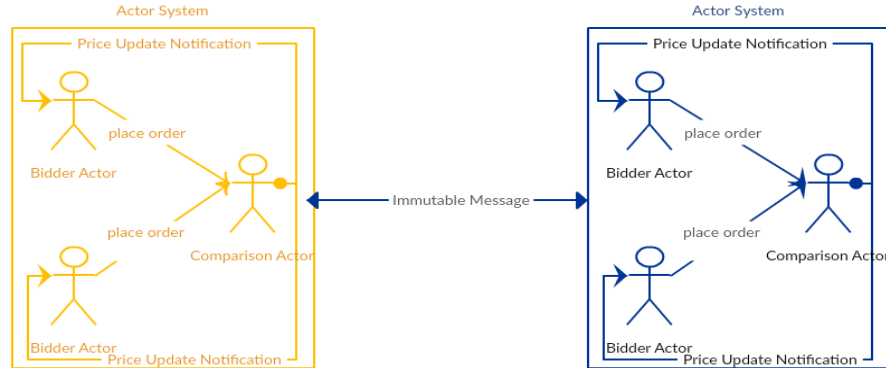


Figure 3.2 The communications between actor systems

Shown in Figure 3.2, when bidders place their offers, they must require the timestamp for server side first. The bidder actor will send a request to ServerTimeSync Actor via timestamp message. There are three parts of information in one message: project ID, bidder ID, and timestamp. When a message arrived the ServerTimeSync Actor, this actor will generate a new timestamp and update its state, which will be

stored in the document database. This information will be an intense connection between actors.

A Comparison Actor also communicates with the bidder actor via BidderMessage shown in Figure 3.3. The difference is that the comparison actor will set up an observer pattern to keep tracking the messages from bidder actor in real time. The comparison actor has the capability to subscript several actors' messages at the same time so that the messages from various actors in different regions are able to be handled immediately.

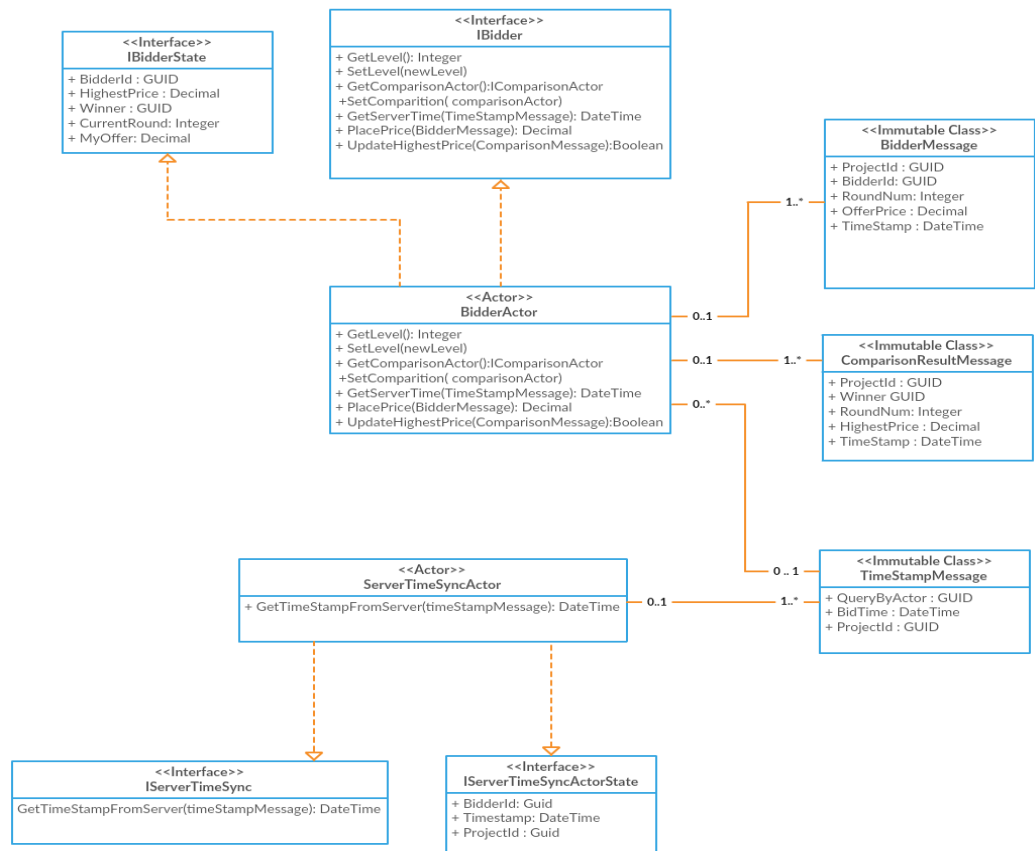


Figure 3.3 A bidder actor and its behaviours

3.2.2 Fault Tolerance of The Actor

The fault-tolerant mechanism notoriously is hard to implement correctly the distributed systems that include actor model applications and distributed databases (Stutsman, Lee, & Ousterhout, 2015). The reason is that the nodes may crash before

finishing its computing or database server may crash and result in losing all data and replicas shown in Figure 3.4.

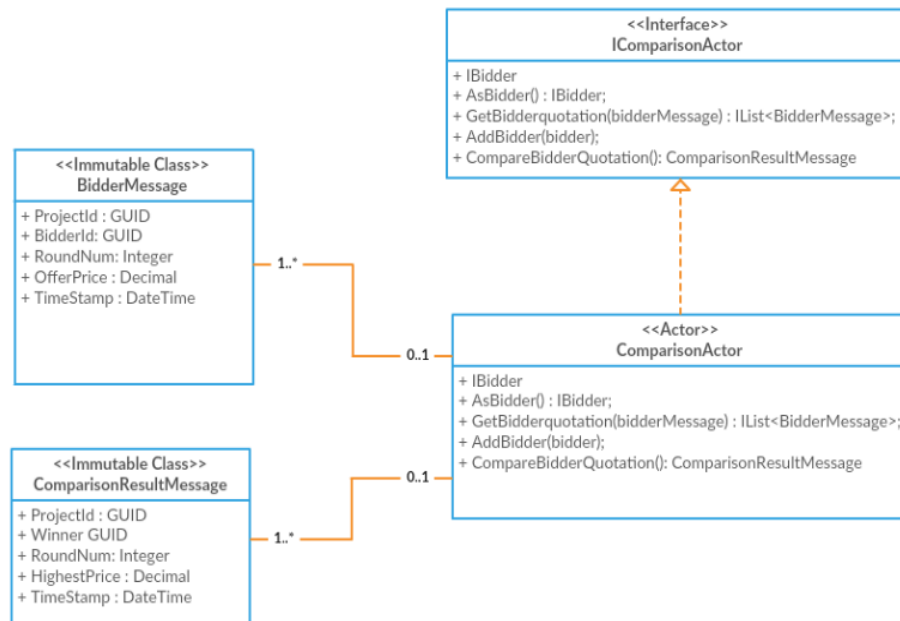


Figure 3.4 A diagram of the comparison actor

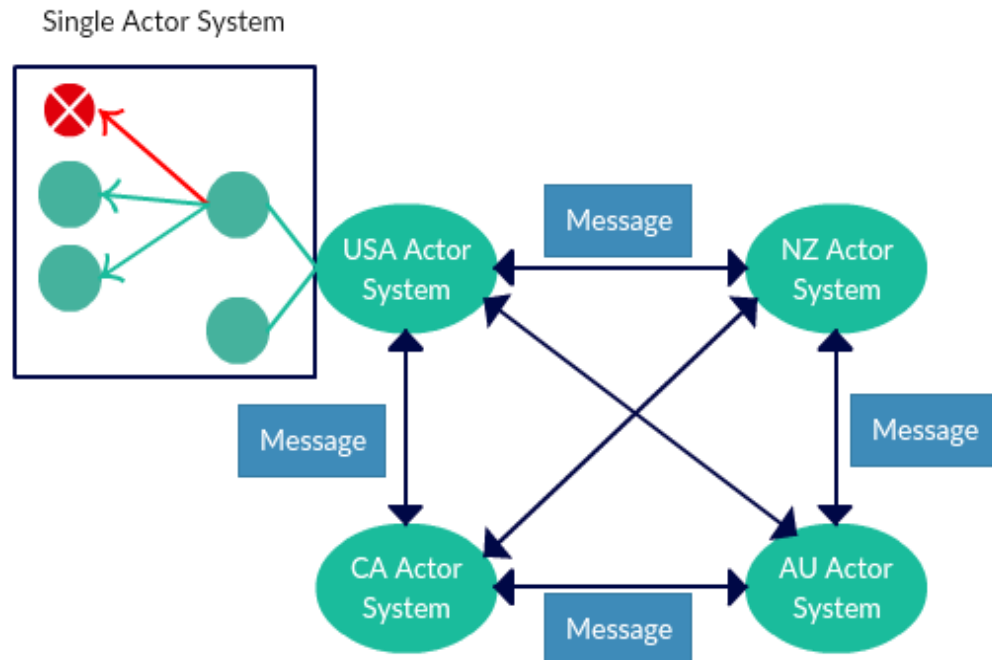


Figure 3.5 Fault tolerance of actor model

The actor models commonly employ hierarchies of supervisor strategy to establish an intensive supervisor – children relationships to achieve an efficient solution of fault-tolerance (Lu et al., 2016). When the failure occurs in nodes, the error message will propagate upwards to the root node. If the error node cannot be activated, the root node will force to restart the node and recover its state and message box (Armstrong et al., 1993).

3.2.3 Location Transparency

Location transparency allows actors and actor systems to easily talk to each other without knowing physical locations. The actor is designed to extend out to a significant number of dedicated servers and allow this actor instantiated at different places (Thurau, 2012). There is nearly no physical address for the actors. They may exist in the purely virtual memory. In this way, the actor model can easily relocate some actors to a different host server, so that we can scale out our web applications (Bernstein et al., 2014). Significantly, the actors no longer heavily rely on the Web API for the actual remoting layer.

3.2.4 State Persistence in Actor Model

There are two options for persistent in actor model. Firstly, we load state from the external relational database, or system information, such as bidder’s user ID, level, and project ID from the authentication token. Moreover, we can populate the variables of the actor. Another option is that we can choose the distributed document database on the cloud, like Document DB on Azure or Mongo DB on AWS. Both document databases provide the shard mechanism. Specifically, the document can be spread with reading and writing transactions across more infrastructures with a high throughput. Technically, the storage that is used to store the state data on the cloud can scale out unlimited shown as Figure 3.4.

The document has a unique ID and a partition key for collection shard. Documents are stored in a collection. Meanwhile, the collection can be shard between different servers in various locations. Even if some data fragments are lost, they could be recovered by using replicas of document database.

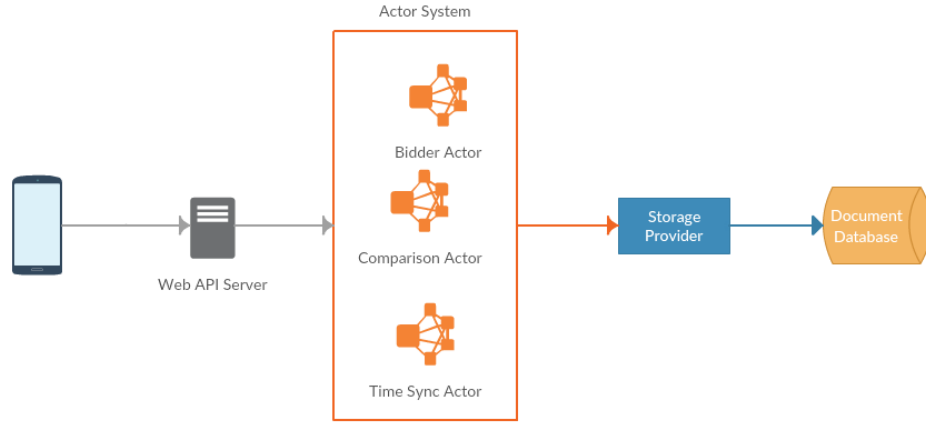


Figure 3.6 State persistence for bidding

3.3 Distributed Database

The traditional relational database has difficulties to fulfil the requirements of the high-concurrent applications. As we all know, its biggest problem is that it is hard to deal with the exponential growth of data collection (Tauro, Aravindh, & Shreeharsha, 2012). If we need to track and record the status of all bidders, the process will generate a plenty of associated data. Additionally, a complex relationship like many to many relationships between tables, which enables data query, has become tough in a relational database. For instance, some of the queries need to cross several tables for retrieving many related data. In this stage, relational databases must join tons of tables together and traverse all data pace by pace.

3.3.1 High Performance Read/Write

To achieve the best performance of an actor model, the databases that are served for it must be disturbed and extendable. Meanwhile, they are also demanded to fulfil the requirement of high performance of reading and writing with high concurrency and low latency (Han, Haihong, Le, & Du, 2011). Most of the document databases, like MongoDB, offer the capability of massively-parallel data processing (Moniruzzaman & Hossain, 2013). Owing to the dependence on the relationship between the tables, the document database is schemaless. We deposit complex data types with BSON or JSON document so that we are able to speed up the access to mass data highly. The access to

Mongo DB is at ten times faster than relational database such as MySQL (Castro & Liskov, 1999).

3.3.2 Fault Tolerance of The Database

Generally, most of the NoSQL databases use replication and sharding to provide the fault-tolerant design.

- Replication allows database scale horizontally. It is also called master-slave replication. In the pattern, only the master database responds to write request; slaves respond to read requests from clients.
- Sharding machines allow us to store separated replication sets into each shard and results in offering high availability and data consistency.

Additionally, another replication algorithm (Byzantine faults tolerate BFT) from MIT can provide highly available service without interruptions like system bugs, accidental operation and malicious attacks (Castro & Liskov, 1999).

3.4 Elliptic Curve Cryptography

3.4.1 Elliptic Curve Digital Signature Algorithm

Elliptic Curve Digital Signature Algorithm (ECDSA) implements in an online transaction based on the blockchain. The elliptic curve for cryptography (ECC) is utilised to the establishment of open key encryption algorithm (Koblitz, 1987; Miller, 1985). Compared to the RSA algorithm, the advantage of using ECC is that it has shorter keys to achieve the same security strength. Elliptic Curve cryptography is based on the Eq.(1).

$$Y^2 = (X^3 + aX + b) \bmod p \quad (3.1)$$

where the possible value of Y^2 should be between 0 to p , we thus have the modulo p .

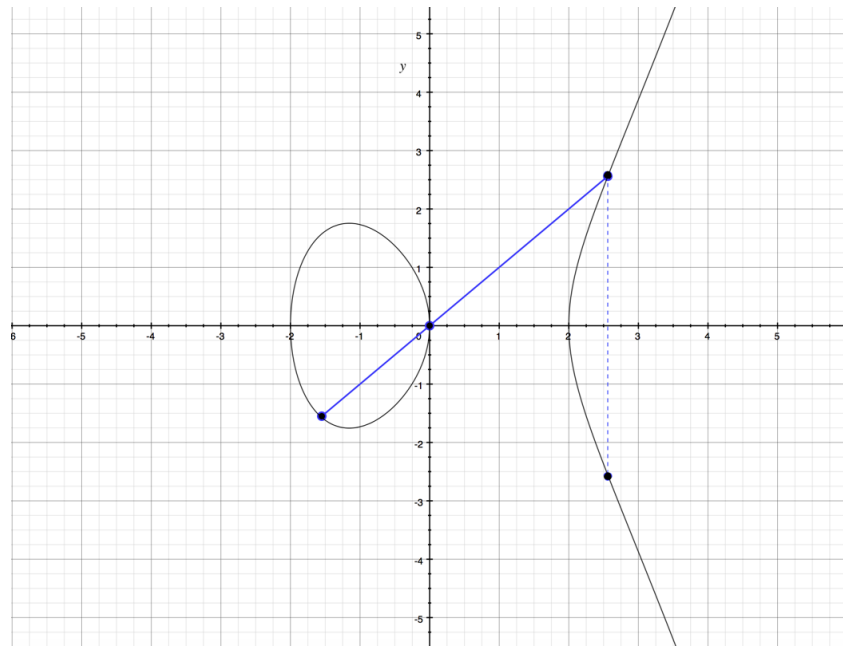


Figure 3.7 The addition of two distinct elliptic curve points

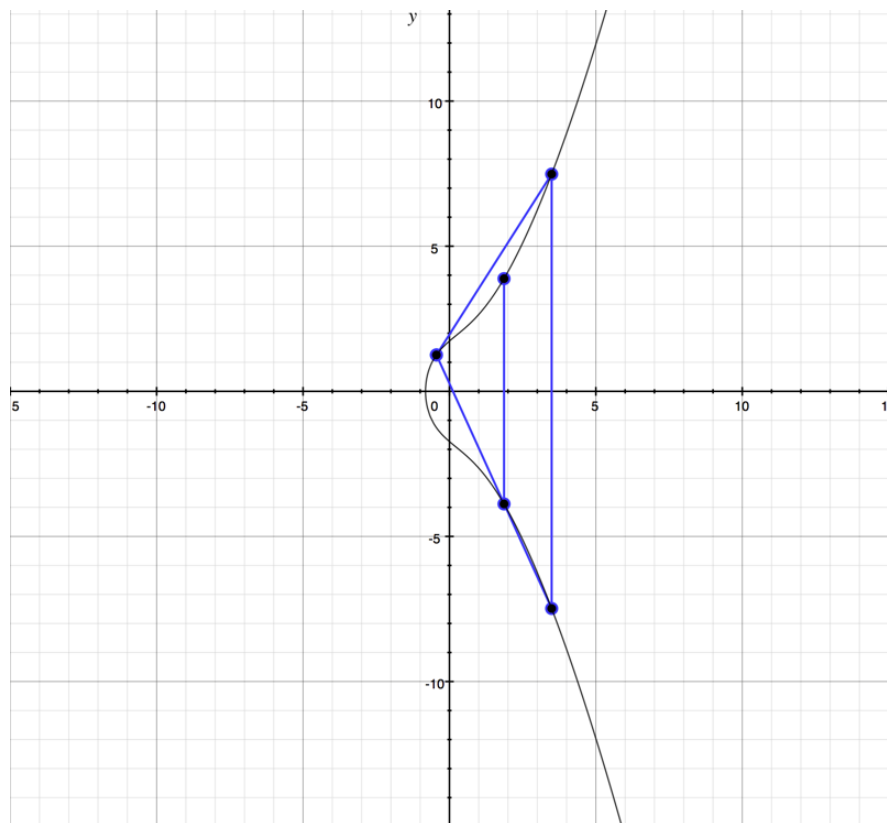


Figure 3.8 Doubling a point on an elliptic curve

There is a significant rule that is called chord-and-tangent rule. For example, let $p = 23$ and the elliptic curve $E: Y^2 = X^3 - 4X + 0$ where $a = -4$ and $b = 0$. We have two distinct points on an elliptic curve shown in Figure 3.6, E : $\mathbf{P} = (x_1, y_1)$, $\mathbf{Q} = (x_2, y_2)$. If we draw a line through \mathbf{P} and \mathbf{Q} , there must be an intersection point \mathbf{R} on the elliptic curve. Then the reflection point (x_3, y_3) of \mathbf{R} is the sum of \mathbf{P} and \mathbf{Q} .

$$x_3 = \left(\frac{y_2 - y_1}{x_2 - x_1} \right)^2 - x_1 - x_2 \quad (3.2)$$

$$y_3 = \left(\frac{y_2 - y_1}{x_2 - x_1} \right) (x_1 - x_3) - y_1 \quad (3.3)$$

In the same way, we have a point $\mathbf{P} (x_1, y_1)$ on the elliptic curve shown in Figure 3.7, $E: y^2 = x^3 + 3x + 3$ where $\mathbf{P} \neq -\mathbf{P}$, and $a = b = 3$. We are able to draw a tangent so that it will intersect with the E at the third point, and its reflection point will be at $2\mathbf{P} = (x_2, y_2)$. Thus, we draw a line from $2\mathbf{P}$ to \mathbf{P} and it will intersect on the curve, and the symmetrical point is $3\mathbf{P}$,

$$x_2 = \left(\frac{3x_1^2 + a}{2y_1} \right)^2 - 2x_1 \quad (3.4)$$

$$y_2 = \left(\frac{3x_1^2 + a}{2y_1} \right) (x_1 - x_2) - y_1 \quad (3.5)$$

One particularity of this point is that if we have a point: $k\mathbf{P} = \mathbf{P} + \mathbf{P} + \mathbf{P} + \dots + \mathbf{P}$ (k times and $k \in \mathbf{Z}^+$ is a positive)

3.4.2 Implementation

Elliptic Curve Digital Signature Algorithm (ECDSA) is implemented to Blockchain for online auctions. In blockchain, we use public key encryption to create a key-pair, which is able to control the acquisition of specific transactions, like virtual currency. The key pair includes a private key and the only public key derived from it. The public key is used to receive the transaction, and the private key is used for the transaction signature when the operation is finished.

Mathematically, the relationship between the public key and the private key is that the private key is able to be utilised for the signature of a particular message. This signature is able to be used to verify the public key. Meanwhile, we do not need to disclose our private key.

When the transaction is finished, the current owner of the virtual currency needs to submit its public key and signature in the transaction (the signatures of each operation are different, but they are made from the same private key). All traders in an blockchain secured online auction are able to be verified by using the submitted public key and signature for the sake of verifying the validity of the transaction.

In an online auction, the whole process uses ECDSA with the secp256k1 curve. We will demonstrate that the above procedure includes three phases: key generation, signature, and verification (Zyskind & Nathan, 2015). To be more specific, the key pair of an entity is associated with a specific set of EC domain parameters (Johnson et al., 2001), $D = (q, FR, a, b, G, n, h)$.

The generation of key pairs follows:

Step 1. Select a random integer d in $[1, n-1]$

Step 2. $Q = dP$ (P is a point of prime order n in the E)

Step 3. The public key is Q , while the d is a private key.

The signature message follows:

Step 1. Select a random or pseudorandom integer k in the interval $[1, n-1]$.

Step 2. Compute $kP = x_1, y_1$ and $r = x_1 \bmod n$ (where x_1 is regarded as an integer between 0 and $q-1$). If $r = 0$, then go back to **Step 1**.

Step 3. Compute $k^{-1} \bmod n$.

Step 4. Compute $s = k^{-1} \{h(m) + dr \bmod n$, where h is the Secure Hash Algorithm (SHA-1). If $s = 0$, then go back to Step 1.

Step 5. The signature for the message m is the pair of integers (r, s)

Verification follows the steps:

Step 1. Verify that r and s are integers in the interval $[1, n-1]$.

Step 2. Compute $w = s^{-1} \bmod n$ and $h(m)$

Step 3. Compute $u_1 = h(m)w \bmod n$ and $u_2 = r_w \bmod n$.

Step 4. Compute $u_1\mathbf{P} + u_2\mathbf{Q} = (x_0, y_0)$ and $v = x_0 \bmod n$.

Step 5. Accept the signature if and only if $v = r$.

3.5 Blockchain Network Architecture

3.5.1 Introduction

We set up our private blockchain on Microsoft Azure for the experimental purpose. Fundamentally, the network is composed of a set of transaction nodes and a set of mining nodes. The network architecture is illustrated in the Figure 3.8.

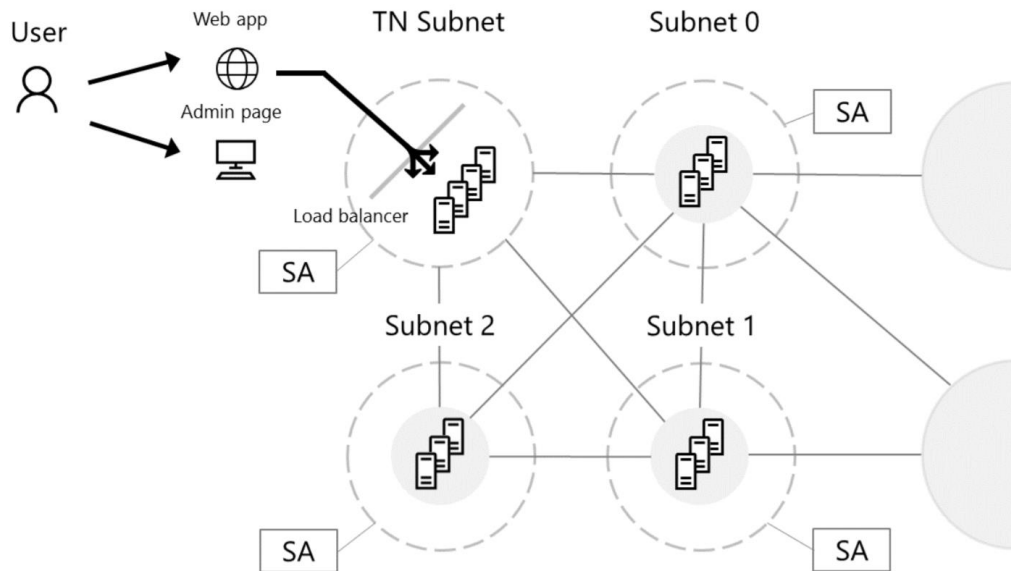


Figure 3.9 Blockchain network architecture

The transaction nodes are responsible for handling the submitted transactions from the application. In the real blockchain application, all members who connect to the same blockchain system share a set of transaction nodes. The loading balance system covers These nodes.

We have explicitly separated the nodes that accepted transactions from the nodes that mine transactions to ensure that the two actions are not competing for the same resources. We have also load-balanced the transaction nodes within an availability set to maintain high availability. According to our requirement, the smallest possible deployment (shown in figure 3.9 and 3.10) for one member will need:

- Four virtual machines (4 cores) will include two miners and two transaction nodes
- One loading balancer
- One Vnet
- One generic type of the storage account
- One DNS server for mapping the public address

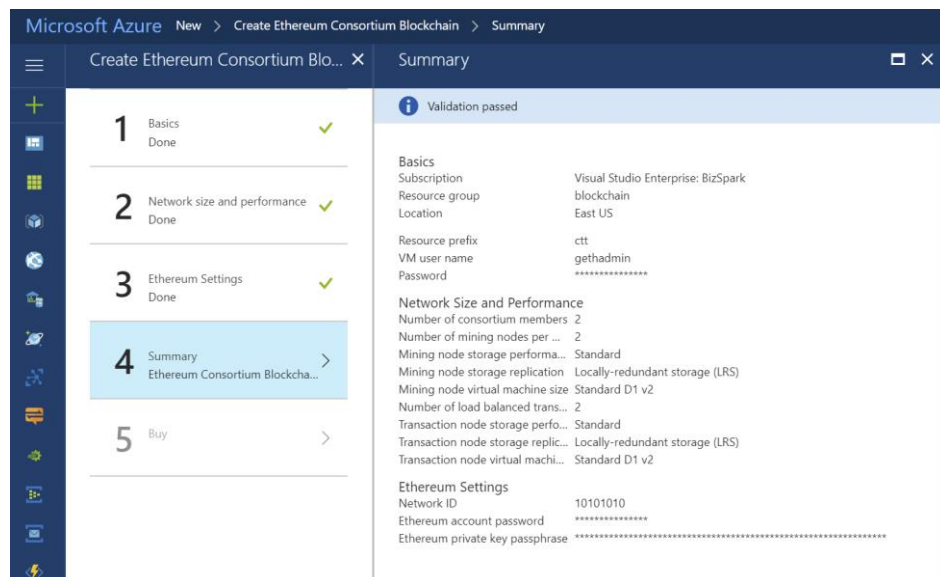






















Figure 3.10 Summary of deployment of our private blockchain

<input type="checkbox"/>	NAME ↑↓	TYPE ↑↓	LOCATION ↑↓
<input type="checkbox"/>	 cttwnniagAvSet	Availability set	East US
<input type="checkbox"/>	 cttwnniag-LB	Load balancer	East US
<input type="checkbox"/>	 nic-mn0	Network interface	East US
<input type="checkbox"/>	 nic-mn1	Network interface	East US
<input type="checkbox"/>	 nic-mn2	Network interface	East US
<input type="checkbox"/>	 nic-mn3	Network interface	East US
<input type="checkbox"/>	 nic-tx0	Network interface	East US
<input type="checkbox"/>	 nic-tx1	Network interface	East US
<input type="checkbox"/>	 cttwnniagMNNsg	Network security group	East US
<input type="checkbox"/>	 cttwnniagTXNsg	Network security group	East US
<input type="checkbox"/>	 cttwnniag-publicip	Public IP address	East US
<input type="checkbox"/>	 l46tabcp4nakqmn0	Storage account	East US
<input type="checkbox"/>	 l46tabcp4nakotx	Storage account	East US
<input type="checkbox"/>	 cttwnniag-mn0	Virtual machine	East US
<input type="checkbox"/>	 cttwnniag-mn1	Virtual machine	East US
<input type="checkbox"/>	 cttwnniag-mn2	Virtual machine	East US
<input type="checkbox"/>	 cttwnniag-mn3	Virtual machine	East US
<input type="checkbox"/>	 cttwnniag-tx0	Virtual machine	East US
<input type="checkbox"/>	 cttwnniag-tx1	Virtual machine	East US
<input type="checkbox"/>	 cttwnniagvnet	Virtual network	East US

Two mining nodes and two transaction nodes for each member.

Figure 3.11 Details of our private blockchain deployment

3.5.2 Mining Nodes

In the private blockchain (Ethereum-based), we set up two members for the experiment purpose. Ideally, four members will provide a standard, decentralized architecture in the real production environment. Additionally, each member is assigned a detached and an identical subnet that includes one or more miners. A storage account supports all mining nodes.

The first virtual machine in the subnet of each member is configured as a boot node. This node is used to discover the other nodes automatically and dynamically in our network. The boot node in the subnet has the list like we have two mining nodes and

two transaction nodes. Once we add a new node in the subnet, the boot node will detect the changes and add the node to its list.

Mining nodes are able to communicate with each other so as to achieve the consensus on the status of the underlying distributed ledger. Because of the boot node, the application layer will not be aware of these node, or communicate with them. Additionally, our private blockchain is deployed on the private networks, so these nodes are detached from public Internet traffics from the outside. The only way to access these nodes to use the Go Ethereum (Geth) to visit the JSON-RPC endpoint that is a remote procedure call protocol. Meanwhile, the outbound internet traffic is acceptable, but we will not allow exposing the Ethereum discovery port to outside.

Inside of our private blockchain network, we allow each member's VMs (nodes) to connect and communicate with another in a separate subnet by using the Ethereum's discovery protocol. This will help the mining nodes to achieve the consensus, even if they are in located in different members.

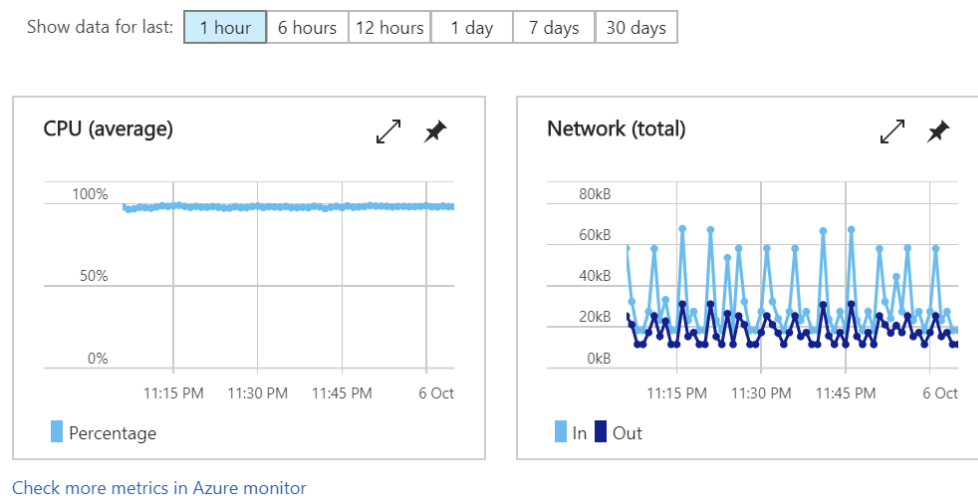


Figure 3.12 Resource allocation of miners

All our mining nodes have already installed the latest Geth client software and are configured to be mining nodes. Thus, we are able to control the resource allocation, and access to smart contract for these nodes using the same Ethereum address and key pair that is protected by the personal account password. To be more specific, we are

able to allocate how many cores of CPU, network, or memory we will assign to each mining node (shown in Figure 3.11). Moreover, we also need to provide the Ethereum passphrase (dynamically generated mnemonic code for generating deterministic key pair, shown in Table 3), when we create the blockchain network, which is used to generate the default account (eth coinbase) for each mining node. After we finish all deployment of our blockchain, all mining nodes begin to work. They will start to mine coins, and that will be added to the coinbase account (shown in Figure 3.11).

How many mining nodes we need in a member entirely depends on the size of the required network. For instance, how many users will execute their transactions per second. Meanwhile, it also depends on the hashing speed for each member. Thus, the larger the private blockchain, the more mining nodes we will need to balance the hashing power.

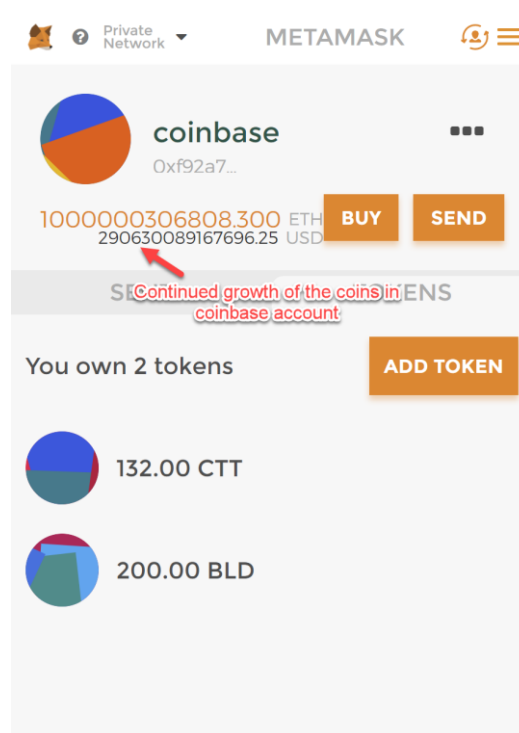


Figure 3.13 Coinbase account

3.5.3 Transaction Nodes

In our private blockchain network, all members share a set of loading balanced nodes of the transaction. These nodes are able to be accessed from outside the virtual network to the application layer is able to use these nodes to submit transactions or deploy and execute the specific smart contract within our private blockchain network. All transaction nodes are installed the Geth client as well. Meanwhile, they are configured to preserve an entire copy of distributed ledger. To ensure that the mining and transaction will not compete for the same resource, we have already clearly separated the nodes that received transactions from the mining nodes (used to mine transactions).

Table 3.1 Mnemonic code and root key

BIP39 Mnemonic	entire canal bonus call arrow there slide march above neutral delay equip vault relief element sick humble carry picnic solve cheap
BIP39 Seed	f9e844f1e1aa01f3b36d934db9d40f06fc1037febf03badd69c52626c04f783985eea3cb7d3c8ff1233785e788c6acf6133e2520c7d2fe291138a52be0b3f1ab
Coin	Ethereum
Root Key	xprv9s21ZrQH143K3QVSQELdTvgCMLh3V2Wk4N8FKCdE3oBZNqGgpDduLHmq795ZCFKomPN8GmQDg34ktQGxF9KBd5NbXFEJ5oTPxM2zRHEDX3n

3.5.3 Actor Framework Works with Blockchain

There are three important characters that are involved in the progress – actor system, shard mongo database, and private blockchain network as shown in Figure 3.13.

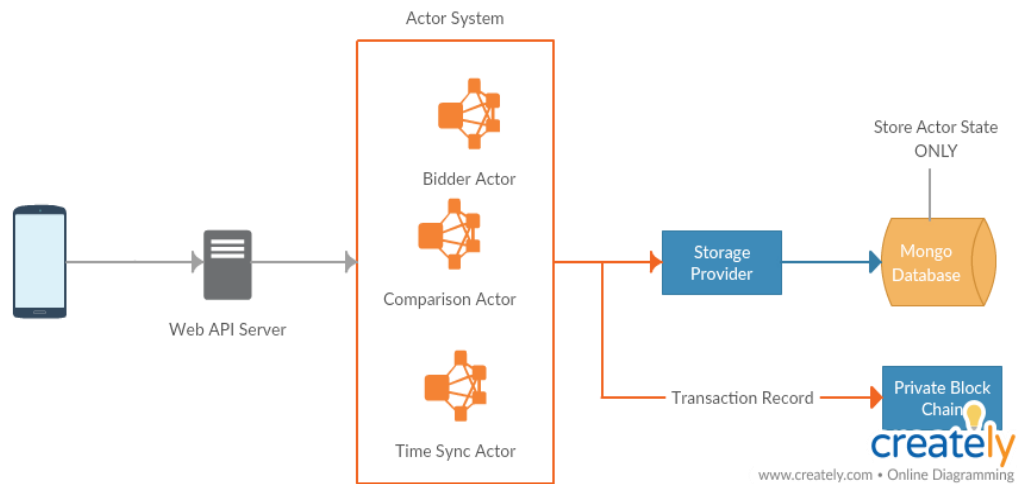


Figure 3.14 Actor system, mongo database and blockchain network

The actor system is able to provide real-time data processing capability in a distributed environment, thus the state changes for very single actor will be stored in mongo database nearby. The shard collections of mongo database will be deployed to different locations for the sake of fulfilling with the requirement of distributed system architecture, which means that we will not store all transaction data into blockchain when actor state changes. The reason is that the amount of state data from actors are extremely large. The transaction will cost large amount of money to pay for the transaction fee. Moreover, mining a single transaction on the blockchain will spend twenty seconds to one minute at least. Thus, the actors will trigger the significant transaction records only in a certain circumstance; for example, the comparison actor is able to trigger the transaction when deposit of an online auction is required to pay, or the final price is settled in an online auction.

To be more specific, the actors are responsible for recording the rapid changes of the transaction state during an online auction, but the changes are not settled. Once the transactions are over, actors will send the transaction record via JSON RPC and Web3 library to the smart contract, which has been deployed on the blockchain network

(shown in Figure 3.14). Thus, the smart contract will execute immediately to transfer money from transferor's account to transferee's account. Meanwhile, the ownership will be transferred as well. All the crucial transactions will not be interrupted by any human intervention. Technically, once the transaction is executed successfully by the smart contract, the transaction information will broadcast to whole main network of blockchain. In addition, the smart contract on the blockchain network is totally transparent, so anyone who is interested in the contract is able to download the contract and the data inside of it.

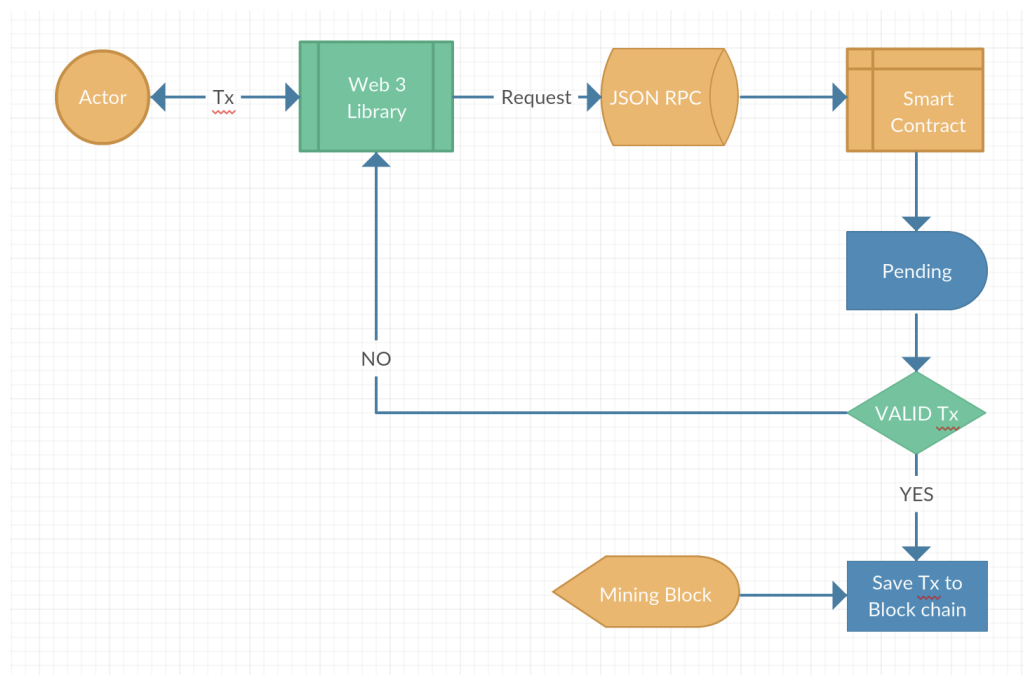


Figure 3.15 Actor communicate with blockchain

The advantages and disadvantages of the actor framework and blockchain are described in the article. Obviously, the cooperation of the actor framework and the blockchain, on the one hand, compensates for the speed of the blockchain in real-time transactions. On the other hand, it provides zero-knowledge proof for online trading systems.

Chapter 4

Results

The main content of this chapter is to introduce schema of the entire method and implementation of actor framework and Elliptic Curve Digital Signature Algorithm. Each step in the Elliptic Curve Digital Signature Algorithm will be detailed; in addition, data collections with the experimental environment will be articulated in this chapter, as well as the results of performance of actor framework with NoSQL database and critical indicators of the blockchain will be clarified. Moreover, the results and findings will be evaluated as well as the limitations of this thesis will be pointed out at the end of this chapter.

4.1 Actor Framework

The actor framework we proposed is Orleans from Microsoft. The latest version of Orleans combined most of the actor models from Erlang and Akka. We implemented three grains (bidder actor, time sync actor and comparison actor) in an actor system. We implemented a single SiloHost server on the local server and calling actors from another laptop. We find that the advanced features of the actor are stronger than the legacy client / server framework shown in Table 4.1.

Table 4.1 A feature comparison of the actor framework and the client-server architecture

#	Actor framework	Client-Server
1	Tracking the states' changes to clients immediately	Needing to request states' changes to clients from the server every time.
2	Providing safe message delivery	Utilising SSL protocol and digital signature technique to guarantees the message security.
3	Distributed to different server	Only implemented on one server
4	At-least-once model guarantees no packet loss	Packet loss in heavy network traffic
5	Supervision strategy and internal state provide data fault tolerant	No fault tolerant support
6	Location transparency	Physical address requirements

#	Actor framework	Client-Server
7	Capability of nodes generation	Do not support children management.
8	Using NoSQL database to tracking persistent state (record states' changes every time)	Support NoSQL or Relational database to store the final result (winner only)

In Table 4.1, we find that the actor is stateful. Each actor shares its inner state via immutable message. We are able to see the state tracking on the server side. When a bidder actor tries to get the timestamp from the ServerTimeSync actor, its behaviours or states will be stored in a database on Azure. The messages passing through the actors are in pairs. Each actor has a unique GUID code so that every behaviour of the actor is trackable. The state persistence ensures the data concurrency and consistency.

Because of this, the actor framework is able to effectively reduce the packet loss rate and effectively increase the transmission speed. As described Table 4.2, we exported from the mongodb, there are four clients got the timestamp from the server side. These states will be pushed to client side at least once to guarantee there is no package loss during the transaction. Additionally, these states are cached in both sides; thus, the transaction speed will extremely fast when states are passed by different actors. The details of the comparison of data transfer will be explained in the next chapter.

Table 4.2 TimeAsync Actor State in MongoDB

User ID	timestamp
f429397c-8f21-4f19-99a9-4af6cca46fcd	17-06-2016 14:52:08
00dace91-e238-40d4-b8bf-a0637877d4ef	17-06-2016 14:52:10

a4f84aa1-c9c2-42db-82bc-a3a8e8f26eaa	17-06-2016 14:52:11
0b323884-05b8-47f1-8b9f-aec863a44fd	17-06-2016 14:52:14
511163ee-59d4-4485-9632-c7d7e8326d52	17-06-2016 14:52:15

In order to observe the price notification between bidder actors, we set up three bidder robots to bid the price randomly in one hundred rounds. Specifically, every time a bidder bids the highest price, the asynchronous notification will be triggered. The Comparison Actor will send the notifications to all bidders (except winner itself) through winner message. The advantage of the asynchronous notification is that the winner actor will not be noticed, which means that the server side will not broadcast to all clients, they only update bidders' state. In this way, we improve the loading balance of the network. In addition, the document database is schemaless. The data with different structures are saved in the same collection. To be more specific, we store the bidder actor, compare actor and server time async actor's states in the same collection though their data structures are entirely different. The benefit of the schemaless is that we do not need to join or transaction data in various tables which result in providing high-performance reading, writing, updating and deleting (CRUD).

4.2 ECDSA

We now used the design algorithms for testing. Our example was a prototype of an online auction which we have developed. Compared with the most of the online auction system, we implemented the blockchain into our system. Meanwhile, we utilise standard cryptographic building blocks in our platform: keys generator, digital signature and verification respectively employed by using the ECDSA prime256v1 curve.

First, the SHA-256 result is shown along with the private and public set of keys.

Input: “37F01AC0-66D5-49DA-AE14-E5F369225C5E”,

SHA -256 Hash

Output: 0d13ba7e63ee5faa77214fde9541e4cc4ec70cc22b5341e415a85ad955b6d46c

We utilise the SHA-256 hash to reproduce the hashcode that is stored in the head of each block. The new block’s hash code is hashed by its parent’s hash code, including previous block hash, timestamp, difficulty and nonce. So, each block is tightly linked together by using the chain. The most significant thing is that the hash is irreversible. This prevents the block from being deleted and changed. Thus, we can easily retrieve the verified history of the transaction (This process is also known as zero-knowledge proof) (Parno, Howell, Gentry & Raykova, 2013). These features are critical to prevent the online auction fraud. For example, we register a diamond, and its certification is based on the blockchain for auction. Before the diamond becomes the smart property, the specific data of the diamond, such as brand, colour, size, certification number, and price, will be broadcasted for verification. Private key signs these broadcast messages, so other networks can easily verify these messages by using the public key. This process will be described in detail in the next section. If the data is verified, the blockchain is able to generate a new node to save the data.

The next step is to generate the paired keys. According to the algebraic description over F_p , the p is a prime number. In our cryptographic applications, p must be a vast random prime number.

Key Pair Generation: 256-bit random private key and corresponding public key. We use the NIST standard curve (P-256) to the implementation of EC cryptography.

The modulus p is:

1157920892103562487626974469494075735300861434152903141955
33631308867097853951

The order n :

115792089210356248762697446949407573529996955224135760342
422259061068512044369

The domain parameter seed is:

c49d3608 86e70493 6a6678e1 139d26b7 819f7e90

According to the P-256 standard, we can generate the paired keys:

-----BEGIN EC PRIVATE KEY-----

MHcCAQEEIKz8GGNNeWs79SyS7oKiceneJ97VZ/oHbLwl1TU+qKYloAoGCCq
GSM49AwEHoUQDQgAEDzylCotL5r+Tmr8eDRBk3mJ0rZbQwlpbBV04P3BZx4J
C/66YCs93DNEvM09v40zS+DamySjZbpCQ8r0SDUb7UA==

-----END EC PRIVATE KEY-----

-----BEGIN PUBLIC KEY-----

MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEDzylCotL5r+Tmr8eDRBk3mJ0r
ZbQwlpbBV04P3BZx4JC/66YCs93DNEvM09v40zS+DamySjZbpCQ8r0SDUb7UA
==

-----END PUBLIC KEY-----

```
{
  "hash" : "00000000569b85207a17444f19fb54aa21870cb4c47a914121870cb4c47a9141",
  "confirmations" : 308321,
  "size" : 285,
  "height" : 0,
  "version" : 1,
  "merkleroot" : "dd29ecf524b030a65261e3059c48ab9e1ecb258548ab9e1ecb258548ab9e1",
  "tx" : [
    "3492cdeab6fbf14cab4cbcb0be831c23dc772bf6831c23dc772bf6831c23dc772bf6"
  ],
  "time" : 1498049926399,
  "nonce" : 20832332434,
  "bits" : "1d00ffff",
  "difficulty" : 1.00000000,
  "nextblockhash" : "00000000569b85207a17444f19fb54aa21870cb4c47a914121870cb4c47a9145"
}
```

Figure 4.1 Blockchain hashcode example

```

priv:
  00:ac:fc:18:63:4d:79:6b:3b:f5:2c:92:ee:82:a2:
  71:e9:de:27:de:d5:67:fa:07:6c:bc:25:d5:35:3e:
  a8:a6:25
pub:
  04:0f:3c:a5:0a:8b:4b:e6:bf:93:9a:bf:1e:0d:10:
  64:de:62:74:ad:96:d0:c2:5a:5b:05:5a:38:3f:70:
  59:c7:82:42:ff:ae:98:0a:cf:77:0c:d1:2f:33:4f:
  6f:e3:4c:d2:f8:36:a6:c9:28:d9:6e:90:90:f2:bd:
  12:0d:46:fb:50
ASN1 OID: prime256v1
NIST CURVE: P-256

```

Figure 4.2 Private key and public key in hexadecimal

Signature:

We utilise the private key to signature (r,s) on the file as shown in Fig 4.2, where

r=0xB1CC56C49D15D43065D6C33856CCA8B0267C8808E4F585DEFC5B6A1007
40870E

s=0xDD37897025A9BA67192604B68BA3EF43AC3BBAC6335AC3966E03C3845
7FD2B6B

```

{
  "confirmations": 308421,
  "size": 285,
  "height": 0,
  "version": 1,
  "merkleroot": "4a5e1e4baab89f3a32518a88c31bc87f618f76673e2cc77ab2127b7afdeda33b",
  "tx": [
    "4a5e1e4baab89f3a32518a88c31bc87f618f76673e2cc77ab2127b7afdeda33b"
  ],
  "time": 1497454344469,
  "nonce": 2083235593,
  "bits": "1d00ffff",
  "difficulty": 1.00000000
}

```

Figure 4.3 The transaction file needs to be signed by private key

Proof of Verification:

We have already known the r and s in the signature (r, s) on the transaction file, so the easiest method to prove the verification is to utilise the OpenSSL. All platforms, such as Linux, Unix and Windows, have this built-in command line. We save the private key into the file: `ec_private_key.pem`, and then store the public key into the file: `ec_public_key.pem`. The signature transaction file is stored in `ec_signature.der`. Meanwhile, the transaction file is kept in the `transaction.json` file. We only need to run and execute the command line below:

```
OpenSSL dgst -sha256 -verify ec_pub_key.pem -signature ec_signature.der
transaction.json
```

We emphasise that this example involves extremely modest-size integer numbers for the sake of explanation of the basic principle of ECDSA. In blockchain applications, these integers are typically 256 bits long. For example, the hash code in the header of the block is actually hashed twice by the SHA256 hash algorithm. Thus, this process will result in dramatically increasing the cost performance of operations above. However, the process also dramatically enhances the cost of hacking. In other words, it is impossible to restore the private key from the public key.

The experimental result demonstrates the fact that the implementation of blockchain in online auction system is able to prevent the online fraud issue. Because the ECDSA is employed, every essential link needs to be signed and verified. So, we are able to quickly retrieve the critical information about the transaction records, like traders' identities, the authenticity of the goods or the trading history of the trading items. We do not need to predict the potential fraud through analysing the behaviour pattern of the transaction of sellers.

4.3 Transaction Verification in Blockchain

The ECDSA is the essence of blockchain applications. This system is a point-to-point (P2P) network whose primary purpose is to propagate transactions that need to be validated to all participants (Antonopoulos, 2014). Generally, the registration of smart properties or the transaction of payment is validated through replicated execution of the nodes that receive this signification information. For example, we register a

house on blockchain for online auction. The house's registration data, such as real estate license, land certificate, holder history information, and so forth, will be signed by our private key and broadcasted to other networks (the third part organisation) for verification. When these messages arrive at other institutions' network, their verification server is able to decrypt these messages by using our public key. We will know where the verification request comes from. It is an excellent idea to ensure the authenticity of the data.

The land certificate data is able to be verified by a government department, who will decrypt verification request and then check whether our information of land certification is entirely matched their database; the historical records of housing transaction are able to be verified by banks. They can retrieve all transaction records of the house in the database to compare our data of trading history. The potential auction price will also be circulated in the community (other online auction organisations) for price comparisons. Validation data contain transaction records of a similar model of units and nearby locations. Thus, another auction organisation will inform us that whether our price is reasonable. The whole process of verification is covered by ECDSA algorithm so that verification request and response are impossible to be tampered. Meanwhile, the verification response with the digital signature is issued by trusted hosting organisations. Thus, these processes are quite useful for providing a high degree of credibility under the untrusted environment.

The properties are able to be registered, or the payments are able to be finished successfully if and only if the validation process is valid. This is a huge difference from the most popular algorithm for fraud detection which utilises the data mining to optimise the fraudulent behavioural patterns from social networks or reputation systems to estimate the possibility. By contrast, the blockchain only stores the solid data for transactions. It is easier to authenticate that the transaction data are fake or not.

4.4 Implementation of Tokens on Private Blockchain

Now, we utilise the ERC20 token (Vogelsteller, 2015) to implement our smart contract. As shown in Figure 4.4, we set up a House token smart contract for our online

action. We are not going to dive into the details of this contract. All design is to facilitate functionality proposed.

We see that our House token is worth one million dollars. Once we deploy this contract on the private blockchain, this contract will own a unique public address (Shown in Figure 4.5). Meanwhile, only the deployer owns this token. The owner of this token is able to transfer some tokens to another shareholder. If one of the clients owns 100% of the token, the ownership will be automatically transferred. The owner is able to transfer the ownership to another users directly. The value of the token can be increased by calling the mint function, or we also can reduce the value of the token by calling the burn function. Furthermore, the owner is able to authorize the agent to assist him/her to transfer a specific amount of token to another client.

```
pragma solidity ^0.4.11;

import './MintableToken.sol';
import './BurnableToken.sol';

/**
 * @title HouseToken ERC20 token
 *
 */
contract HouseToken is MintableToken, BurnableToken {
    string public name = "HouseToken";
    string public symbol = "BLD";
    uint public decimals = 2;
    uint public INITIAL_SUPPLY = 100000000;

    function HouseToken() {
        balances[msg.sender] = INITIAL_SUPPLY;
    }
}
```

Figure 4.4 House token

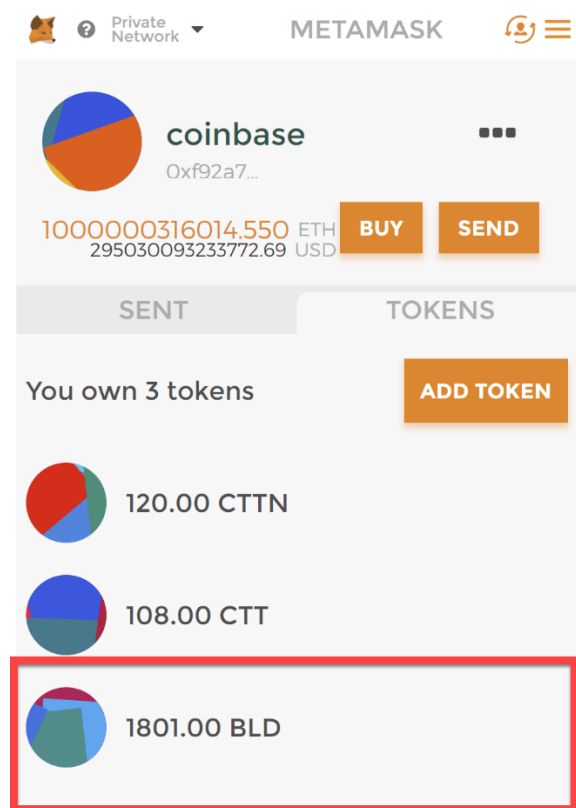


Figure 4.5 House token in wallet

4.5 Bidding on Blockchain

In order to illustrate the bidding process on the blockchain, we set up a scenario to describe the auction process. A company bought a hundred coffee cups from suppliers as shown in Figure 4.7, a desired price (\$10.00/each) was set for this smart contract as shown Figure 4.6. Once the contract is deployed to the blockchain, the suppliers are able to bid the price.

	address	state	created	ad...	del...	name
1	b38d78f576ea18b1b9bb20b07f97172305f4210d	1	1531910905239		0	3 Bed Rooms House
2	083f8ffc28730d8672666de757210ef645664b3f	2	1531911320334		0	Coffee Cup

Figure 4.6 Smart contract of coffee cup

When the suppliers bid the price (Figure 4.8), the data will be signed and store in the state of comparison actors. The buyers are able to track price changes on real time

from the actor state (Figure 4.9). We must emphasize that these data are not sent to the blockchain. The transaction price will be written to the blockchain if the project is closed shown as Figure 4.10 to Figure 4.11.

The screenshot shows a 'New Project' form with the following fields and values:

- Short name ***: Coffee Cup
- Description ***: Glass or Porcelain
- Desired price ***: 200
- Desired delivery date ***: 31/07/2018
- Specification ***: We need to buy a 100 cups

At the bottom right, there is a green circular icon with a 'G' and a character count '25 / 1000'. Below the form are two buttons: 'CREATE' (blue) and 'CANCEL' (white).

Figure 4.7 Set a new project

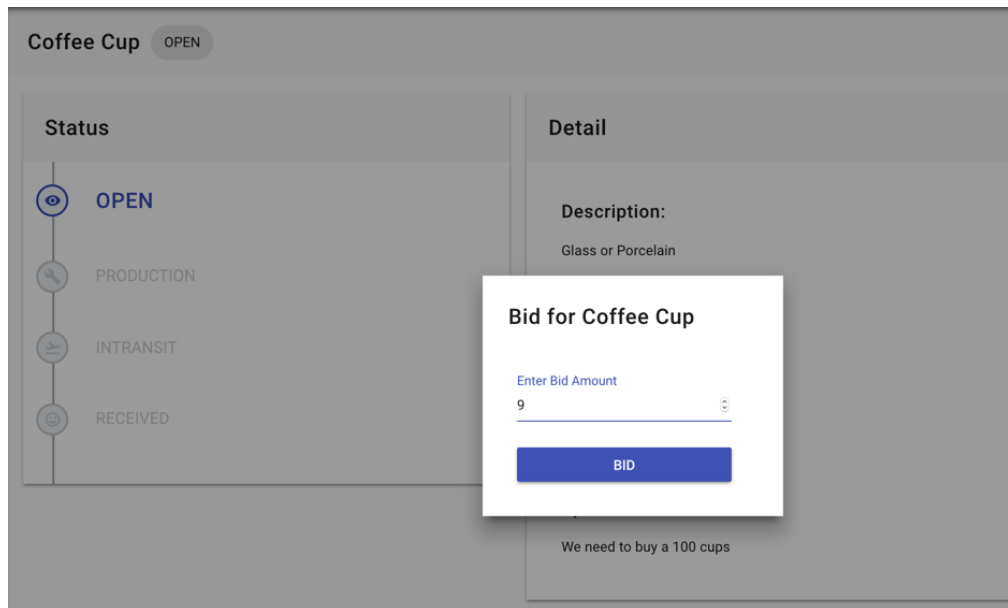


Figure 4.8 Bidding the price

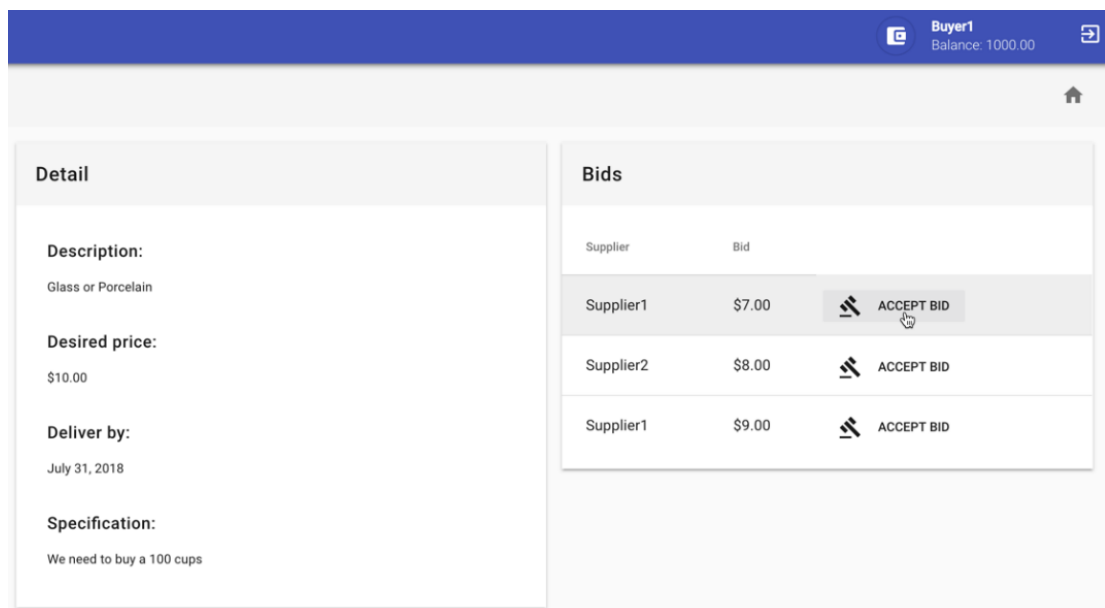


Figure 4.9 Bidding List

Chapter 5

Analysis and Discussions

In this chapter, the discussion and resultant analysis concerning outcomes of the experiments are demonstrated and presented. More specifically, comparisons regarding the performance of the actor framework with NoSQL database and the blockchain will be discussed in this chapter. Finally, the significance will be also stated through analysing the outcomes.

5.1 Analysis

5.1.1 NoSQL Database

The document database, such as MongoDB that we used in the implementation provides high speed of reading, creating, and updating. When the bidder robots get the system timestamp, we track the `ServerTimeSync` actor's state changes. We set up two different databases (SQL Server and MongoDB) as the data record providers. In the process of inserting the first 5000 data, it is hard for us to distinguish who is faster. However, when the amount of data written gradually increased to 15000, the SQL database writing speed is almost ten times slower than MongoDB shown in Figure 5.1.

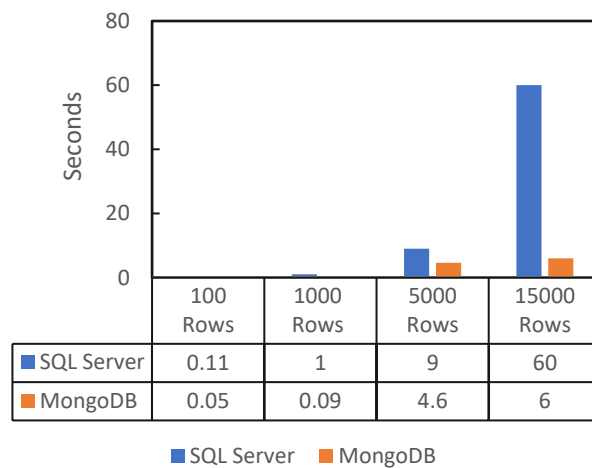


Figure 5.1 Insert `ServerTimeSync` status into database

The state of bidder and comparison is a complex data structure. For instance, states of the comparison actor include bidder actors' states in each round of competition. So, we need to join two or three tables together to query the data in SQL Server. However, the document database only nests the relevant information in a document. For complex data queries, we are not surprised that MongoDB is still faster. When the query data is higher than 15000, SQL takes 16 seconds, and NoSQL Database takes approximately 5 seconds shown in Figure 5.2.

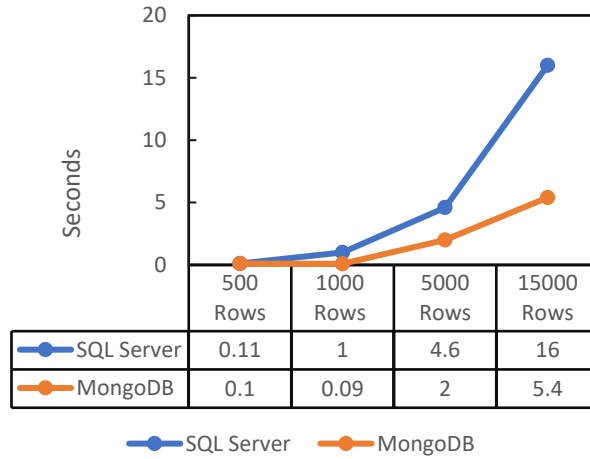


Figure 5.2 Query state from comparison actor and bidder actor

5.1.2 Blockchain

We collect three main index (difficulty, has rate, and cost of each transaction) from the offical website of the Etherscan to demonstrate issues from decentralized blockchian database.

The difficulty of mining a new block is more chanllenge nowadays. Figure 5.3 illustrates the incensement of the difficulty in recent two years of the blockchain network. Especially in recent one year, we see that the difficulty rapidly increases from 80TH over 3397TH. This could be a severe issue for the development of the blockchain. We find a new block will be great hard in future.

Meanwhile, the hash rate of the blockchain is also growing rapidly in recent two years as shown in Figure 5.4. This will result in clients need to spend the very long time to finish a transaction. With the rapid development of blockchain, demands of hashing also soared in the short term. We see that a standard transaction will spend twenty seconds to two minutes waiting for the result of the transaction.

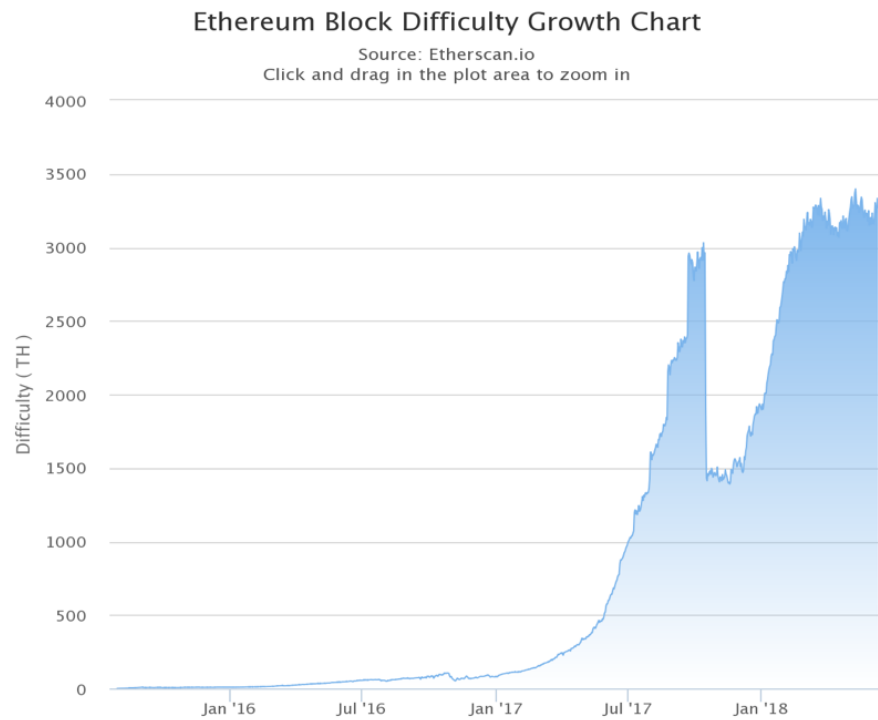


Figure 5.3 Difficulty incensement in recent two years

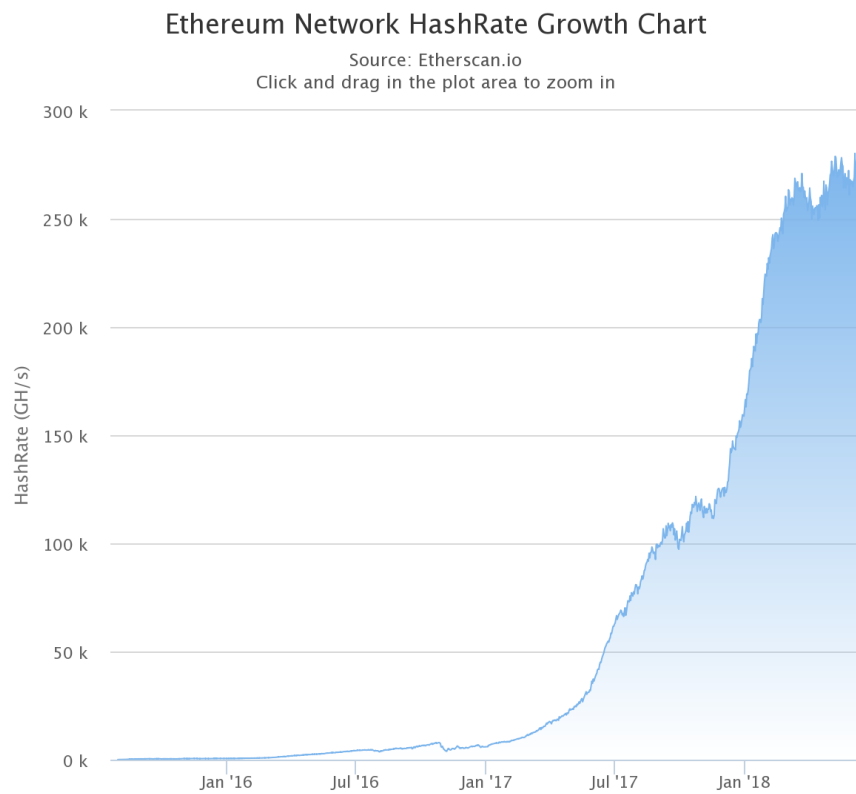


Figure 5.4 Hash rate incensement in recent two years

Also, the cost of each transaction is also growing unexpectedly (as shown in Figure 5.5). Usually, the clients need to pay the commission fee to the third-party trading institutions. Meanwhile, they also need to pay a gas fee to the institutions who are responsible for running the blockchain network. The more gas fee we pay, the faster transaction speed we are able to achieve.

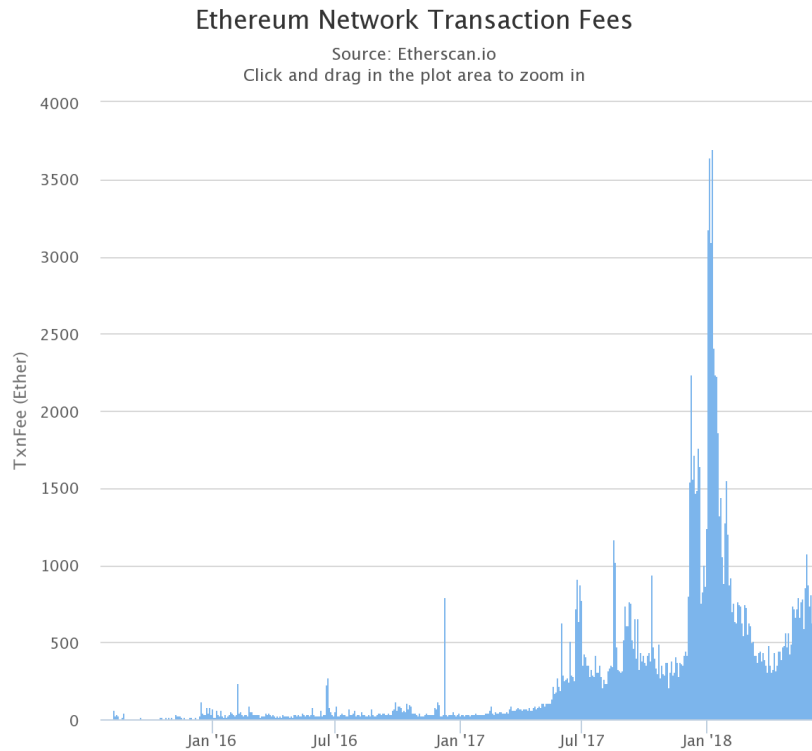


Figure 5.5 Incensement of cost per transaction in recent two years

The last line graph (Figure 5.6) demonstrates the incensement of the block size in recent two years. Because of the distributed architecture of the blockchain, the block size has risen rapidly. Meanwhile, the popularity of the blockchain rises linearly.

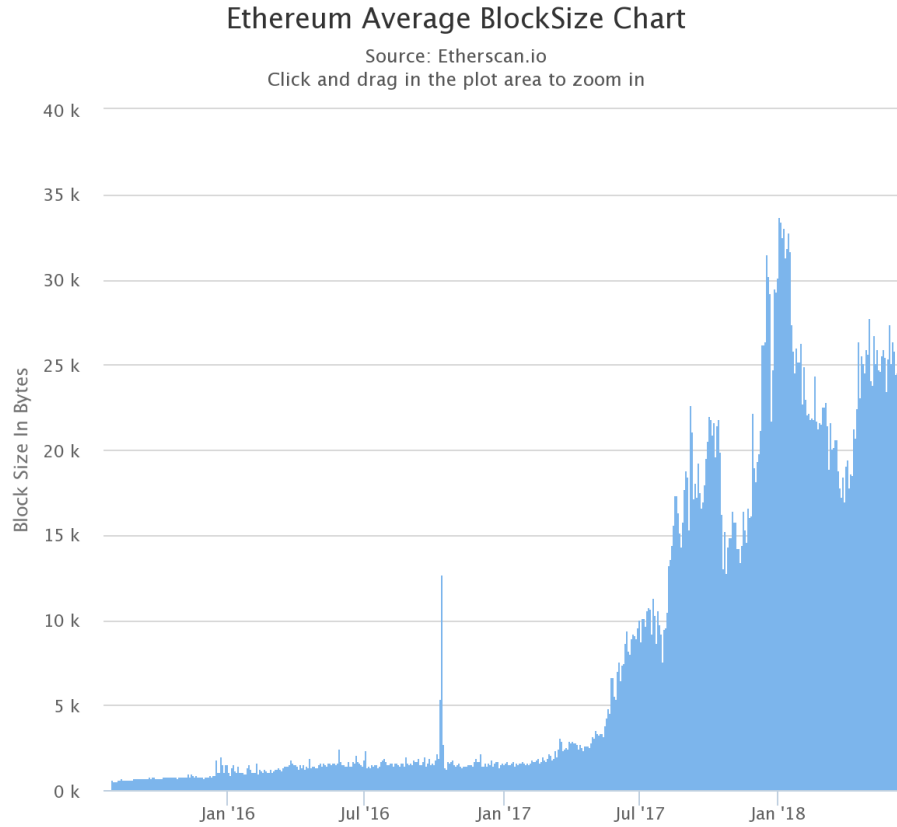


Figure 5.6 Block size incensement in recent two years

5.2 Discussions

Experimental results were detailed and demonstrated in previous Chapters. Firstly, the focus of these experiments is on the actor model framework and No SQL database. The design and implementation of our framework have been elaborated from Section 3.1 to Section 3.2. The purpose is to provide the data consistency and integrity during the high-performance network environment. We improve the packet loss and traffic rates for our cloud online auction model. Moreover, the mechanism of safe message delivery ensures that our bid price cannot be tampered during the data passing from end to end. Secondly, we describe the Elliptic Curve Digital Signature Algorithm, blockchain and smart contract in details from Section 4.2 to Section 4.4. The features of the blockchain are able to help us to establish the trading net under the untrusted environment so that we are able to efficiently address the online fraud issues.

In this thesis, the experiments are detailed in Section 5.1.1 and the discussions will

be followed in Subsection 5.1.2. We see that when the amount of data (simple structure) gradually increased to 15000 rows, the SQL database writing speed is almost ten times slower than NoSQL database. Even if we are using the complex data queries, we are not surprised that NoSQL database is still faster. When the query data is higher than 15000 rows of data, SQL takes sixteen seconds to read, but the NoSQL database takes approximately five seconds. Because of the advantages of the NoSQL database, we track and record all state changes from the client side, while we do not need to worry about the performance of database have negative effect on online action system.

Moreover, Section 5.1.2 illustrates the progression of the blockchain in recent two years. According to our analysis of official data like difficulty, hashing rate, cost per transaction and block size, the virtual concurrency technology grows rapidly in very short term. It provides a new perspective to solve the online fraud issues on the trading network. However, the difficulty of the hashing has directly resulted in the transaction pending issues on the blockchain network. Meanwhile, the incensement of difficulty also proves that blockchain network is getting bigger and slower. Ideally, the solution is that we can establish our private blockchain locally, and then synchronize the private chain with the Ethereum main network. This will increase the performance of the transaction. Moreover, it will decrease the cost per transaction as well.

Chapter 6

Conclusion and Future Work

In this thesis, in-depth articulation of the techniques was discussed which can be utilized to implement a high-performance actor framework and private blockchain network for the online auction. The corresponding approaches for each step have been implemented as the results of this thesis. In this chapter, we will present this thesis at a scholarly level, also highly organize and integrate the conclusion into the context; meanwhile, the future work will be pointed out by the end of this thesis.

6.1 Conclusion

The online auction platform transfers the market by providing reliable and professional service platform. A vital aspect of this shift is the accommodation of clustering services. These services are no longer limited by the interconnected location of buyers and sellers (Ackerberg et al., 2006). There are 80% of online participants (both buyers and sellers) from different countries or regions.

In this situation, the traditional client-server architecture does not have enough capability to manage all bidders in the distinct clustering of trade to ensure the fairness and reliability of online auctions. English auction is an excellent case in point. It has a very high demand for system concurrency and responsiveness. Because the system needs to complete the price delivery, comparison, and notification in accurate time of period.

Thus, the actor framework is employed to provide the data consistency and integrity during the high-performance network environment. We improve the packet loss and traffic rates for our cloud online auction model. Moreover, the mechanism of safe message delivery ensures that our bid price cannot be tampered during the data passing from end to end. The message is wrapped in the web stock; we cannot guarantee that any messages can be received or dispatched each time successfully. Thus, the additional implementation must be taken to actor mode like at-least-once in Orleans. It requires retry when transport losses. Although immutable message guarantees data security within the delivery process, we cannot stop online fraud by using actor model. Additionally, we run a single Silo Host server on the local server, so the network condition is not considered. In future, we set a REST API server to test the performance in a different network environment.

Online fraud detection is extremely hard to be implemented. The traditional methodology does not have enough capability to prevent online fraud in the distinct clustering of trade so as to ensure the fairness and reliability of online auctions. Blockchain 2.0 and 3.0 provided the fundamental solution for these issues. English auction is an excellent case in point. It has a very high demand for protecting the transactions under the untrusted environment. Blockchain provides a complete set of

secure trading mechanisms from the zero-knowledge proof mechanism, smart property to smart contract.

Thus, blockchain is employed to provide the private transaction in the untrusted environment. Once the subject is registered into the blockchain, it will automatically become a smart property. All relevant data of the subject will be stored in the distributed blockchain and are not able to be deleted and modified. In each of the online auction transactions, all relevant data about the subject has been verified (broadcast the information to all distributed blockchain databases for verifications), such as house ownership certification. When the online auction transactions are finished, the smart contract will be automatically verified, and the transaction is completed.

6.2 Future Work

Our future work includes,

(1) We implemented a single SiloHost server on the local server and calling actors from another laptop. In future, we need to implement the framework into the complex environment. For instance, the SiloHost server is able to connect to server-less web API, so that we are able to test the performance of the actor framework from end to end. We hope to achieve more concrete data to support our result.

(2) We established a private blockchain for test our smart contract. We have already proved that this methodology can improve our transaction performance on our private blockchain server. However, we still need to sync with the Ethereum main network. In future, we will need to improve the private blockchain architecture in this thesis for the next massive step for syncing with the main network.

(3) We also need to implement the Hawk-like smart contract in our private blockchain network. Because the blockchain network exposes all information to the public, the Hawk-like contract is able to establish the privacy protections of the smart contract, so we only need to expose the auction information to participants.

References

- Ackerberg, D., Hirano, K., & Shahriar, Q. (2006) The buy-it-now option, risk aversion, and impatience in an empirical model of eBay bidding. University of Arizona.
- Agha, G. A. (1985) *Actors: A model of concurrent computation in distributed systems*. The MIT Press.
- Agha, G., Mason, I. A., Smith, S., & Talcott, C. (1992) Towards a theory of actor computation. In *International Conference on Concurrency Theory* (pp. 565-579). Springer, Berlin, Heidelberg.
- Akerlof, G. A. (1970) The market for ‘lemons’: quality uncertainty and the market mechanism. *Aug*, 84(3), 488-500.
- Alanezi, F. (2016) Perceptions of online fraud and the impact on the countermeasures for the control of online fraud in Saudi Arabian financial institutions. Brunel University London,
- Andrychowicz, M., Dziembowski, S., Malinowski, D., & Mazurek, L. (2014) Secure multiparty computations on bitcoin. In *IEEE Symposium on Security and Privacy (SP)* (pp. 443-458). IEEE.
- Antonopoulos, A. M. (2014) *Mastering Bitcoin: unlocking digital cryptocurrencies*: USA: O'Reilly Media, Inc.
- Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., . . . Stoica, I. (2010) A view of cloud computing. *Communications of the ACM*, 53(4), 50-58.
- Armstrong, J. (2007) *Programming Erlang: software for a concurrent world*:

- Armstrong, J., Viriding, R., Wikström, C., & Williams, M. (1993) Concurrent programming in ERLANG.
- Ba, S., & Pavlou, P. A. (2002) Evidence of the effect of trust building technology in electronic markets: Price premiums and buyer behavior. *MIS Quarterly*, 243-268.
- Ba, S., Whinston, A. B., & Zhang, H. (2003) Building trust in online auction markets through an economic incentive mechanism. *Decision Support Systems*, 35(3), 273-286.
- Becker, J., Breuker, D., Heide, T., Holler, J., Rauer, H. P., & Böhme, R. (2013) Can we afford integrity by proof-of-work? Scenarios inspired by the Bitcoin currency. In *Economics of Information Security and Privacy* (pp. 135-156): Springer.
- Bernstein, D. (2014) Containers and cloud: From lxc to docker to kubernetes. *IEEE Cloud Computing*, 1(3), 81-84.
- Bernstein, P. A., Bykov, S., Geller, A., Kliot, G., & Thelin, J. (2014) Orleans: Distributed virtual actors for programmability and scalability. USA: Microsoft Research Press.
- Birch, D., Brown, R. G., & Parulava, S. (2016) Towards ambient accountability in financial services: Shared ledgers, translucent transactions and the technological legacy of the great financial crisis. *Journal of Payments Strategy & Systems*, 10(2), 118-131.
- Black, J., Hashimzade, N., & Myles, G. (2013) Committee on Payment and Settlement Systems. UK: Oxford University Press.

- Bonneau, J., Miller, A., Clark, J., Narayanan, A., Kroll, J. A., & Felten, E. W. (2015) Sok: Research perspectives and challenges for bitcoin and cryptocurrencies. In IEEE Symposium on Security and Privacy (SP) (pp. 104-121). IEEE.
- Buterin, V. (2014) A next-generation smart contract and decentralized application platform. Ethereum White Paper
- Castro, M., & Liskov, B. (1999) Practical Byzantine fault tolerance. In Third Symposium on Operating Systems Design and Implementation (pp. 173–186)
- Chan, H. C., Ho, I. S., & Lee, R. S. (2001) Design and implementation of a mobile agent-based auction system. IEEE Pacific Rim Conference on Communications, Computers and signal Processing.
- Choi, S.-Y., Stahl, D. O., & Whinston, A. B. (1997) The economics of electronic commerce: Macmillan Technical Publishing Indianapolis.
- Christidis, K., & Devetsikiotis, M. (2016) Blockchains and Smart Contracts for the Internet of Things. IEEE Access, 4, 2292-2303.
- Chua, C., & Wareham, J. (2002) Self-regulation for online auctions: An analysis. Self, 12, 31-2002.
- Chua, C. E. H., & Wareham, J. (2004) Fighting internet auction fraud: An assessment and proposal. Computer, 37(10), 31-37.
- Clack, C. D., Bakshi, V. A., & Braine, L. (2016) Smart contract templates: foundations, design landscape and research directions.
- Corcoran, C. (1999) The auction economy. Red Herring, 69.
- Danezis, G., Fournet, C., Kohlweiss, M., & Parno, B. (2013) Pinocchio coin: building zerocoin from a succinct pairing-based proof system. In Proceedings of the

First ACM workshop on Language support for privacy-enhancing technologies (pp. 27-30). ACM.

De Koster, J., Marr, S., D'Hondt, T., & Van Cutsem, T. (2013) Tanks: multiple reader, single writer actors. In Proceedings of the 2013 workshop on Programming based on actors, agents, and decentralized control (pp. 61-68). ACM.

Delmolino, K., Arnett, M., Kosba, A., Miller, A., & Shi, E. (2016) Step by step towards creating a safe smart contract: Lessons and insights from a cryptocurrency lab. In International Conference on Financial Cryptography and Data Security (pp. 79-94). Springer, Berlin, Heidelberg.

English, M., Auer, S., & Domingue, J. (2016) Block chain technologies & the semantic web: a framework for symbiotic development. In Computer Science Conference for University of Bonn Students, J. Lehmann, H. Thakkar, L. Halilaj, and R. Asmat, Eds (pp. 47-61).

Estrada, R., & Ruiz, I. (2016) Big Data SMACK. Apress, Berkeley, CA.

Eyal, I., & Sirer, E. G. (2018) Majority is not enough: Bitcoin mining is vulnerable. Communications of the ACM, 61(7), 95-102.

Fournet, C., Kohlweiss, M., Danezis, G., & Luo, Z. (2013) ZQL: A Compiler for Privacy-Preserving Data Processing. In USENIX Security Symposium (pp. 163-178).

Franco, P. (2014) Understanding Bitcoin: Cryptography, engineering and economics: John Wiley & Sons.

Froomkin, A. M. (1996) Essential Role of Trusted Third Parties in Electronic Commerce. Oregon Law Review, 75, 49.

Gao, B., Zhang, S., & Yao, N. (2012) A Multidimensional Pivot Table Model Based

on MVVM Pattern for Rich Internet Application. In International Symposium on Computer, Consumer and Control (IS3C) (pp. 24-27).

Garman, C., Green, M., & Miers, I. (2016) Accountable privacy for decentralized anonymous payments. In International Conference on Financial Cryptography and Data Security (pp. 81-98). Springer, Berlin, Heidelberg.

Grazioli, S., & Jarvenpaa, S. L. (2000) Perils of Internet fraud: An empirical investigation of deception and trust with experienced Internet consumers. IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, 30(4), 395-410.

Greenspan, G. (2015) Ending the bitcoin vs blockchain debate. MultiChain blog.

Gupta, M. (2012) Akka essentials: Packt Publishing Ltd.

Haller, P. (2012) On the integration of the actor model in mainstream technologies: the scala perspective. The 2nd edition on Programming systems, languages and applications based on actors, agents, and decentralized control abstractions.

Han, J., Haihong, E., Le, G., & Du, J. (2011) Survey on NoSQL database. In Pervasive computing and applications (ICPCA) (pp. 363-366). IEEE.

Hecht, R., & Jablonski, S. (2011). NoSQL evaluation: A use case oriented survey. In International Conference on Cloud and Service Computing (CSC) (pp. 336-341). IEEE.

Hewitt, C., Bishop, P., & Steiger, R. (1973) Session 8 formalisms for artificial intelligence a universal modular actor formalism for artificial intelligence. In Advance Papers of the Conference (Vol. 3, p. 235). Stanford Research Institute.

Industries, E. (2016) Explainer | Smart Contracts. Eris Industries Documentation.

- Johnson, D., Menezes, A., & Vanstone, S. (2001) The elliptic curve digital signature algorithm (ECDSA). *International Journal of Information Security*, 1(1), 36-63.
- Karmani, R. K., Shali, A., & Agha, G. (2009) Actor frameworks for the JVM platform: a comparative analysis. In *International Conference on Principles and Practice of Programming in Java* (pp. 11-20). ACM.
- Kekre, H., & Bharadi, V. (2011a) Dynamic signature pre-processing by modified digital difference analyzer algorithm. In *Thinkquest* (pp. 67-73).
- Kiayias, A., Zhou, H. S., & Zikas, V. (2016) Fair and robust multi-party computation using a global transaction ledger. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques* (pp. 705-734). Springer, Berlin, Heidelberg.
- Koblitz, N. (1987) Elliptic curve cryptosystems. *Mathematics of computation*, 48(177), 203-209.
- Kosba, A., Miller, A., Shi, E., Wen, Z., & Papamanthou, C. (2016) Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In *2016 IEEE symposium on security and privacy (SP)* (pp. 839-858). IEEE.
- Krasnokutskaya, E., Terwiesch, C., & Tiererova, L. (2016) Trading across Borders in Online Auctions. Report, Johns Hopkins University.
- Kreuter, B., Shelat, A., Mood, B., & Butler, K. R. (2013) PCF: A Portable Circuit Format for Scalable Two-Party Secure Computation. In *USENIX Security Symposium*(pp. 321-336).
- Kroll, J. A., Davey, I. C., & Felten, E. W. (2013) The economics of Bitcoin mining, or Bitcoin in the presence of adversaries. In *Proceedings of WEIS* (Vol. 2013, p. 11).

- Kumaresan, R., & Bentov, I. (2014) How to use bitcoin to incentivize correct computations. In ACM SIGSAC Conference on Computer and Communications Security (pp. 30-41). ACM.
- Li, L., Tang, T., & Chou, W. (2015) A rest service framework for fine-grained resource management in container-based cloud. In IEEE 8th International Conference on Cloud Computing (CLOUD) (pp. 645-652). IEEE.
- Li, X., Chang, D., Pen, H., Zhang, X., Liu, Y., & Yao, Y. (2015) Application of MVVM design pattern in MES. In IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER), 2015 (pp. 1374-1378). IEEE.
- Li, X., Jiang, P., Chen, T., Luo, X., & Wen, Q. (2017). A survey on the security of blockchain systems. *Future Generation Computer Systems*
- Liu, C., Wang, X. S., Nayak, K., Huang, Y., & Shi, E. (2015) Oblivm: A programming framework for secure computation. In IEEE Symposium on Security and Privacy (SP) (pp. 359-376). IEEE.
- Liu, H., Wang, S., & Fei, T. (2003) Multicast-based online auctions: a performance perspective. *Benchmarking: An International Journal*, 10(1), 54-64.
- Liu, L. (2012) Analysis and Application of MVVM Design Pattern. *Application of Micro Computer*, 28(12), 57-60.
- Lu, K., Yahyapour, R., Wieder, P., Yaqub, E., Abdullah, M., Schloer, B., & Kotsokalis, C. (2016) Fault-tolerant Service Level Agreement lifecycle management in clouds using actor system. *Future Generation Computer Systems*, 54, 247-259.
- McAdam, R. (2001) Fragmenting the function-process interface: The role of process benchmarking. *Benchmarking: An International Journal*, 8(4), 332-349.

- McGilvary, G. A. (2014) Ad hoc cloud computing. UK: University of Edinburgh
- McGrath, G., Short, J., Ennis, S., Judson, B., & Brenner, P. (2016) Cloud event programming paradigms: Applications and analysis. In International Conference on Cloud Computing (CLOUD) (pp. 400-406). IEEE
- Meiklejohn, S., Pomarole, M., Jordan, G., Levchenko, K., McCoy, D., Voelker, G. M., & Savage, S. (2013) A fistful of bitcoins: characterizing payments among men with no names. In the Conference on Internet Measurement (pp. 127-140). ACM.
- Michael, A., Armando, F., Rean, G., Anthony, D. J., Randy, K., Andy, K., ... & Matei, Z. (2009) Above the clouds: A Berkeley view of cloud computing. EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28.
- Miller, M. S., Morningstar, C., & Frantz, B. (2000) Capability-based financial instruments. In International Conference on Financial Cryptography (pp. 349-378). Springer, Berlin, Heidelberg.
- Miller, M. S., Tribble, E. D., & Shapiro, J. (2005) Concurrency among strangers. In International Symposium on Trustworthy Global Computing (pp. 195-229). Springer, Berlin, Heidelberg.
- Miller, V. S. (1985) Use of elliptic curves in cryptography. In Conference on the theory and application of cryptographic techniques (pp. 417-426). Springer, Berlin, Heidelberg.
- Pagliery, J. (2014) Bitcoin: And the Future of Money: Triumph Books.
- Parashar, M., Abdelbaky, M., Zou, M., Zamani, A. R., & Diaz-Montes, J. (2015) Realizing the potential of iot using software-defined ecosystems. In IEEE International Conference on Cloud Computing (CLOUD) (pp. 1149-1158).

IEEE.

Parno, B., Howell, J., Gentry, C., & Raykova, M. (2016) Pinocchio: Nearly practical verifiable computation. *Communications of the ACM*, 59(2), 103-112.

Peters, G. W., & Panayi, E. (2016) Understanding Modern Banking Ledgers through Blockchain Technologies: Future of Transaction Processing and Smart Contracts on the Internet of Money. In *Banking Beyond Banks and Money* (pp. 239-278): Springer.

Peters, G. W., Panayi, E., & Chapelle, A. (2015) Trends in crypto-currencies and blockchain technologies: A monetary theory and regulation perspective. *Journal of Financial Perspectives*, Vol.3

Pham, L. M., Tchana, A., Donsez, D., De Palma, N., Zurczak, V., & Gibello, P. Y. (2015) Roboconf: a hybrid cloud orchestrator to deploy complex applications. In *International Conference on Cloud Computing (CLOUD)* (pp. 365-372). IEEE.

Philip, J., & Bharadi, V. A. (2016) Article: Online Signature Verification in Banking Application: Biometrics SaaS Implementation. *The International Conference on Communication Computing and Virtualization*.

Pilkington, M. (2016) Blockchain technology: principles and applications. *Research handbook on digital transformations* (pp.225).

Power, R., & Li, J. (2010) Building fast, distributed programs with partitioned tables. In *9th USENIX Symposium on Operating Systems Design and Implementation: (OSDI)*.

Rastogi, A., Hammer, M. A., & Hicks, M. (2014) Wysteria: A programming language for generic, mixed-mode multiparty computations. In *IEEE Symposium on Security and Privacy (SP)* (pp. 655-670). IEEE.

- Ron, D., & Shamir, A. (2013) Quantitative analysis of the full bitcoin transaction graph. In International Conference on Financial Cryptography and Data Security (pp. 6-24). Springer, Berlin, Heidelberg.
- Sáez, S. G., Andrikopoulos, V., Sánchez, R. J., Leymann, F., & Wettinger, J. (2015) Dynamic tailoring and cloud-based deployment of containerized service middleware. In International Conference on Cloud Computing (CLOUD) (pp. 349-356). IEEE.
- Sasson, E. B., Chiesa, A., Garman, C., Green, M., Miers, I., Tromer, E., & Virza, M. (2014) Zerocash: Decentralized anonymous payments from bitcoin. In IEEE Symposium on Security and Privacy (SP) (pp. 459-474). IEEE.
- Smith, J. (2009) Patterns-wpf apps with the model-view-viewmodel design pattern. MSDN magazine, 72.
- Stutsman, R., Lee, C., & Ousterhout, J. K. (2015). Experience with Rules-Based Programming for Distributed, Concurrent, Fault-Tolerant Code. In USENIX Annual Technical Conference (pp. 17-30).
- Swan, M. (2015) Blockchain: Blueprint for a new economy: O'Reilly Media, Inc.
- Szabo, N. (1997) Formalizing and securing relationships on public networks. First Monday, 2(9).
- Tauro, C. J., Aravindh, S., & Shreeharsha, A. (2012) Comparative study of the new generation, agile, scalable, high performance NOSQL databases. International Journal of Computer Applications, 48(20), 1-4.
- Thurau, M. (2012) Akka framework. University of Lübeck.
- Vinoski, S. (2007) Reliability with Erlang. IEEE Internet Computing, 11(6).

- Vogelsteller, F., & Buterin, V. (2015) ERC 20 token standard. Ethereum Foundation (Stiftung Ethereum), Zug, Switzerland.
- Wood, G. (2014) Ethereum: A secure decentralised generalised transaction ledger. Ethereum project yellow paper, 151, 1-32.
- Yu, P., Xia, M., Lin, Q., Zhu, M., Gao, S., Qi, Z., ... & Guan, H. (2010) Real-time enhancement for xen hypervisor. In International Conference on Embedded and Ubiquitous Computing (EUC) (pp. 23-30). IEEE.
- Zheng, L., Chong, S., Myers, A. C., & Zdancewic, S. (2003) Using replication and partitioning to build secure distributed systems. In Symposium on Security and Privacy (pp. 236-250). IEEE.
- Zyskind, G., & Nathan, O. (2015) Decentralizing privacy: Using blockchain to protect personal data. In Security and Privacy Workshops (SPW) (pp. 180-184). IEEE.
- Zyskind, G., Nathan, O., & Pentland, A. (2015) Enigma: Decentralized computation platform with guaranteed privacy.