Authentication, Authorization and Accounting with Ethereum Blockchain

Mukesh Thakur

Master's Thesis
UNIVERSITY OF HELSINKI
Department of Computer Science

Helsinki, September 13, 2017

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Laitos — Institution — Department
Faculty of Science	Department of Computer Science

Tekijä — Författare — Author
Mukesh Thakur

Työn nimi — Arbetets titel — Title

Authentication, Authorization and Accounting with Ethereum Blockchain

Oppiaine — Läroämne — Subject
Computer Science

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Master's Thesis	September 13, 2017	66

Tiivistelmä — Referat — Abstract

Over past decade cloud services have enabled individuals and organizations to perform different types of tasks such as online storage, email services, on-demand movies and TV shows. The cloud services has also enabled on-demand deployment of applications, at cheap cost with elastic and scalable, fault tolerant system. These cloud services are offered by cloud providers who use authentication, authorization and accounting framework based on client-server model. Though this model has been used over decades, study shows it is vulnerable to different hacks and it is also inconvenient to use for the end users. In addition, the cloud provider has total control over user data which they are able to monitor, trace, leak and even modify at their will. Thus, the user data ownership, digital identity and use of cloud services has raised privacy and security concern for the users.

In this thesis, Blockchain and its applications are studied and alternative model for authentication, authorization and accounting is proposed based on Ethereum Blockchain. Furthermore, a prototype is developed which enables users to consume cloud services by authenticating, authorizing and accounting with a single identity without sharing any private user data. Experiments are run with the prototype to verify that it works as expected. Measurements are done to assess the feasibility and scalability of the solution. In the final part of the thesis, pros and cons of the proposed solution are discussed and perspectives for further research are sketched.

ACM Computing Classification System (CCS):
Computer systems organization
    Architectures
        Distributed architectures
            Cloud computing

Avainsanat — Nyckelord — Keywords
AAA, Ethereum, Blockchain

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — Övriga uppgifter — Additional information

Contents

1 Introduction **1**
1.1 Problem Statement . . . 2
1.2 Methodology . . . 2
1.3 Related work . . . 3
1.4 Outline . . . 4

2 AAA by Cloud Provider **4**
2.1 Authentication . . . 6
2.1.1 Username and Password . . . 6
2.1.2 Public Key Infrastructure . . . 7
2.1.3 Biometrics . . . 7
2.1.4 Multi-Factor . . . 7
2.2 Authorization . . . 8
2.2.1 XACML . . . 8
2.2.2 OAuth 2.0 . . . 8
2.3 Accounting . . . 9
2.4 Potential Vulnerabilities . . . 10
2.4.1 Account Hijacking . . . 10
2.4.2 Distributed Denial of Service attack . . . 11
2.4.3 Man-in-the-Middle attack . . . 11
2.4.4 Data Breach . . . 11
2.4.5 Malicious Insiders . . . 12
2.5 Drawbacks . . . 12

3 Blockchain **13**
3.1 Terminology . . . 14
3.1.1 Peer-to-Peer Network . . . 14
3.1.2 Block . . . 15
3.1.3 Blockchain . . . 16
3.1.4 Distributed Blockchain . . . 17
3.1.5 Smart Contracts . . . 17
3.2 Cryptography . . . 17
3.2.1 Cryptographic Hash Function . . . 17
3.2.2 Hash Pointer . . . 18
3.2.3 Digital signature . . . 18
3.2.4 Merkle Tree . . . 19
3.3 Distributed Consensus . . . 20
3.3.1 Proof-of-work . . . 21
3.3.2 Proof-of-stake . . . 22
3.4 Architecture . . . 23
3.5 How blockchain works . . . 25
3.6 Potential Vulnerabilities . . . 26

3.6.1 The 51% attack 26
3.6.2 Sybil attack 27
3.6.3 Identity theft 27
3.6.4 System hacking 27
3.6.5 Illegal activities 28

4 Blockchain Applications 28
4.1 Bitcoin 28
4.1.1 Transactions 29
4.1.2 Scripting Language 31
4.1.3 Bitcoin Network 31
4.1.4 Advantages 31
4.1.5 Vulnerabilities 32
4.2 Namecoin 32
4.3 Ethereum 33
4.3.1 Ethereum Blockchain and Account 34
4.3.2 Transactions and Messages 34
4.3.3 Smart Contracts 35
4.3.4 Decentralized Autonomous Organization 35
4.3.5 Vulnerabilities 36

5 Rational Behind Choice of Technology 36

6 Prototype Design 38
6.1 Need for Prototype 38
6.2 Software Architecture 38
6.3 Flow diagram 40

7 Prototype Implementation 42
7.1 Hardware Components 42
7.2 Software Components 42
7.2.1 Ethereum wallet 42
7.2.2 geth 43
7.2.3 Docker 43
7.2.4 Ethereum explorer 44
7.3 Prototype Environment 45
7.4 Smart Contract Implementation 46
7.4.1 Solidity 46
7.4.2 Mortal contract 47
7.4.3 Provider contract 47
7.5 Executing the Prototype 48

8 Prototype Testing **50**
8.1 Setup and Execution 50
8.2 Measurement and Results 51
8.2.1 CPU and Memory 52
8.2.2 Blockchain Time 53
8.2.3 Transaction Cost 53
8.3 Analysis of Results 54
8.4 Security Analysis 55

9 Discussion **56**

10 Conclusion and Future work **57**
10.1 Conclusion 57
10.2 Future work 59

References **60**

1 Introduction

Most of us access cloud services almost on daily basis. For example, we use web-based email systems such as Gmail, Outlook, Yahoo! to exchange messages with each other. We use social networking sites like Facebook, LinkedIn and Twitter to share information and get connected to friends. There are on-demand services like Netflix and Hulu to watch TV and movies. Cloud storage services such as Google Drive, iCloud and Dropbox store digital media such as photos, videos and documents. Cloud services are used by enterprises to deploy their applications and services for the purpose of reducing the operational cost and improve cash flow. For example, The Uber, ride-sharing service and social news website Reddit use Amazon Elastic Compute Cloud (EC2) to provide their services. Netflix uses the Amazon Web Services to host their services. Github, a code hosting site is deployed on Rackspace. Thus, it is obvious that cloud services have become an integral part of everyday business.

The cloud services are provided by cloud providers who have primary responsibility of authentication, authorization and accounting (AAA) framework. This framework is needed for providing on demand, scalable, elastic, reliable and redundant cloud services [74]. The AAA framework is based on client-server model where the providers develop and maintain client and server services respectively for the users. The users interacts with client application that communicates to centralized servers with request-response method over the Internet. The user must register with the provider to create her digital identity. In this process the user has to provide private sensitive user data, such as firstname, lastname, username, phone number, email and bank or credit card details. These user data are saved on the centralized server across multiple data center. Also, the user has to create multiple digital identities across multiple providers to access their services respectively. Study shows from a user experience perspective, this process of creating multiple digital identity is inconvenient and cumbersome because the user has to repeat same registration process again and again and remember multiple passwords for different services. These user data are vulnerable to hacking [46, 30]. Also, the centralized servers of the providers are primary targets for hackers.

According to ZDNet in 2016 [26]: Tumblr, a social blogging site suffered 65 million accounts hacks. Yahoo!, one of the leading email providers, announced over billion accounts getting compromised by the hackers. LinkedIn, the leading professional network site suffered $117 million account hacks. Weebly, the web designing giant got 43 million user data stolen and Society for Worldwide Interbank Financial Telecommunication (SWIFT) global financial message system had $81 million financial loss because of hacking. The providers might also share the user data to other outside organization without user consent. National Security Agency (NSA) tapped into tech giant such as Apple, Google, Facebook, Amazon Web Services (AWS) to monitor user activity [41]. These issues have triggered a big debate on user data privacy and users are becoming more concerned about security and privacy of their digital identity.

To address privacy, we designed and developed a prototype (proof-of-concept) using blockchain technology. The prototype could authenticate, authorize and accounting without requiring users to share their private data to the providers. The authentication is done by Ethereum blockchain. The immutable blockchain ledger verifies and ensures that the users, transactions, messages are legitimate. Authorization is done by smart contracts which are written and deployed to blockchain. Cryptocurrency ether is used for accounting. The prototype aims to provide self-contained digital identity system to the user. At the same time it provides a common distributed, decentralized identity backend where cloud providers can deploy their own authorization logic and validate the user identities. The prototype feasibility was tested with multiple users and providers where CPU and memory usage, blocktime and transaction costs are measured. We present the result and analyze them. Finally, the security aspects are described and discussed.

This thesis is dealing with a prototype system. The main objective of the thesis is to understand how blockchain works and how it can be integrated with existing cloud providers system and if is feasible to implement the concept. The target is to test if the prototype is able to prove the feasibility of the concept or architecture with pre-defined test cases.

1.1 Problem Statement

We state our problem as follows. How a cloud user can have self-contained user data ownership and use multiple cloud provider services as well as pay invoices without sharing her private data. The system must be transparent, distributed, decentralized, easy-to-use and almost impossible to get hacked. In addition, anyone in the network should be able to verify transactions, message, data but no one is able to access the content of transactions, message or data except the owner.

1.2 Methodology

We selected Design Science research methodology in order to identify the research problem and develop proof-of-concept for research problem. The research problem was identified based on the extensive study of existing technologies and standards for authentication, authorization and accounting by the cloud provider. The solution was designed based on the research of possible solutions of this problem using blockchain technology. The proof-of-concept was designed and developed to support the solution design. Moreover, the solution was tested and measurements were done to present the feasibility of the solution. Finally, analysis was performed from theoretical and practical point of view and further improvements and suggestions were presented for the future work.

1.3 Related work

Blockchain is relatively new technology for authentication, authorization and accounting compared to the existing AAA framework. There have been multiple attempts, research and approaches to provide self-governed user identity, authentication and authorization of the user against the different cloud services and leverage the payment system of blockchain. Some of popular solutions are bitid [49], nameid [47] and uPort [53].

Bitid is an open protocol which allows simple and secure user login to cloud service by authenticating the user based on the public key and blockchain network [49]. The authentication proves the identity of the user to a service by signing a challenge. A cloud provider can use bitid to enable blockchain login for their service. This eliminates the need of user registration and username/password and gives data ownership to the user rather than the cloud provider. A working prototype of this protocol can be checked in here [50].

NameID is experimental technology based on Namecoin 4.2 and OpenID [20] to provide unique, secure, decentralized and distributed digital identity to the users. Namecoin is decentralized Bitcoin based technology which allows a user to register names which can be associated with user data. This data can be verified by everyone in the blockchain network but cannot be forged or censored by unauthorized attackers and no one can retrieve the data without user consent. OpenID is an open protocol that allows a user to authenticate to multiple services without need of creating multiple different identities and passwords. It provides one unique identity to the user from some trusted identity provider which can be used to sign into other OpenID-enabled services. According to OpenID Foundation: over billion users, including technology giants like Google, Microsoft, Facebook, Sun, Yahoo! support OpenID enabled user accounts. Thus, NameID combines Namecoin and OpenID where Namecoin issues digital blockchain identity and associate with user data (username, email, phone number) and OpenID allows instant sign-in to over billion of OpenID enabled websites.

uPort [53] is a platform that provides blockchain based identity for the user and easy-to-use method to interact with decentralized applications or services. This platform allows end users to establish a digital identity which can be used as user identity across multiple cloud services without any passwords. It gives full control of sensitive user data to the user by allowing user to own and control their digital assets, securely and selectively disclose their data to counterparts to access digital service. Moreover, it allows the user to digitally sign and encrypt documents, data, messages, transactions and to send these over the blockchain network to interact with decentralized applications. It is built on top of the Ethereum blockchain and has three main components: smart contract, developer libraries and a mobile app. The smart contract has the core logic to issue and recover user identity. The developer library allows developers to integrate uPort to the third party application in order to use uPort as identity provider. The mobile app issues the identity to the user and provides identity/key management interface.

The solutions presented above provide self-sovereign identity to the user. These solutions can be mostly used for authentication and authorization by the cloud provider. However, a complete AAA solution from the provider perspective which could authenticate and authorize a user to use cloud resources and at the same time easily allow the cloud provider to write and deploy the authentication and authorization logic is missing. Also, the design on how accounting can be used for invoicing and payment is missing. Thus, there is a clear need of a solution which could address these challenges. This thesis provides a unique solution for a user and cloud providers to leverage the advantages of blockchain. This solution enables easy development and deployment of blockchain application for user authentication and authorization. It also enables invoice payments without giving credit card details. The solution is decentralized, distributes over self-sovereign system with no single point of failure.

1.4 Outline

In this Section, the motivation, problem statement with research methodology and related work are discussed. Section 2 describes the key concepts of authentication, authorization and accounting along with possible vulnerabilities. Furthermore, Section 3 covers the key concept of blockchain technology, its architecture, how it works and its vulnerabilities. Section 4 describes the most popular blockchain applications. Section 5 describes the reason for selecting an Ethereum blockchain for implementing prototype and Section 6 describes the prototype design with software architecture and flow diagram. Section 7 describes a prototype implementation with hardware and software components along with its environment setup and actual implementation and execution. Section 8 covers the testing of the prototype and finally, the discussion is presented which is followed by the conclusion and discussion of possible future work.

2 AAA by Cloud Provider

Cloud provider is the entity which offers cloud resources such as computing, network, network storage, applications that can be rapidly provisioned with minimal management effort and paid only for the resources consumed when needed [55]. These resources form services which are categorized into three models by the National Institute of Science and Technology (NIST) [55]: software as a service (SaaS), platform as a service (PaaS) and infrastructure as a service (IaaS). The cloud provider must have five key characteristics: multi-tenancy or shared resources, massive scalability, elasticity, pay-as-you-go and self-provisioning of resources [57, 55].

Authentication, Authorization and Accounting (AAA) is a framework used by the cloud provider for controlling the access of cloud resources, enforcing policies, auditing and measuring resource usage [74]. The cloud provider must use this framework for effective resource, user, network and security management. The AAA is based on client-server model [65] where the cloud user interacts with client and server has the business logic necessary for cloud resources, user, network and security management. Authentication is a process of verifying the identity of the user and authorization is the process of deciding whether a user has enough rights to use the requested service. Accounting is the process of tracking the resource usage by the user for billing, auditing, data analytics.

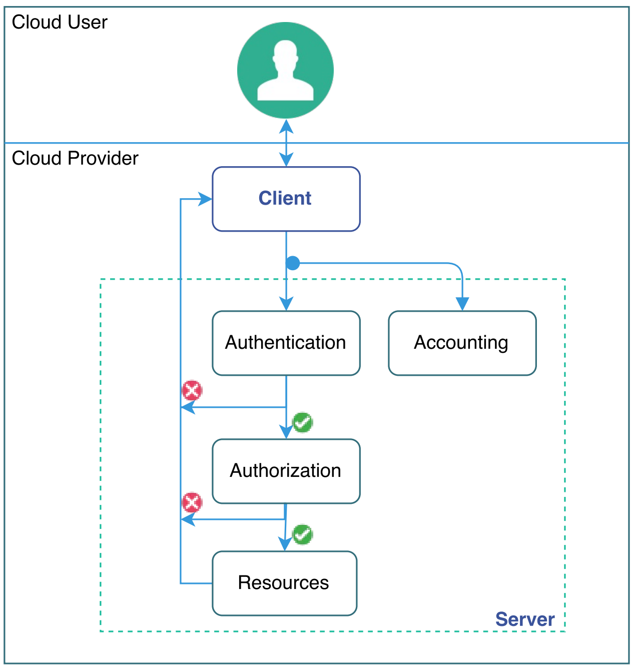


Figure 1: General AAA Architecture

Figure 1 shows the general AAA architecture with client-server model where client is a web or mobile application and server host authentication, authorization, accounting resources and services. The Figure 1 is divided into two parts where the first part has a cloud user and the second part has a cloud provider. The provider has the protected resource which requires authentication and authorization from the user. The user interacts with the provider with client application which sends the request to the server. The server has authentication and authorization services. The authentication service verifies the credential of the user. If verification is successful the request is forwarded to the authorization service. Otherwise, an error is returned and the user is redirected to the client application as shown in the Figure 1. The authorization service determines the authority of the user. If successful, the request is forwarded to the resource service that returns the requested resource by the user. Otherwise, error is returned and user is redirected to the client application. The accounting service intercepts the request between client and server and does the metrics calculation and audition as provisioned by the provider. The AAA are further described in detail in the following subsections.

2.1 Authentication

Authentication is a mechanism by which a cloud provider identifies the cloud user before granting access to the cloud resources [82, 30]. The cloud provider enables the user to use the cloud services based on the credentials provided during the registration [54]. The provider authentication can be categorized into three categories: what-you-know (knowledge), what-you-have (possession) and who-you-are (ownership) [82]. What-you-know means something what the user knows about such as username and password, PIN code and public keys. What-you-have is something what user possess such as smart cards, identity card, e-tokens (identity information encrypted on a flash card). Who-you-are means something that the user owns such as biometric characteristics such as fingerprints and iris scan. Some of the most used authentication methods are described in the next subsections.

2.1.1 Username and Password

Username and Password are the most used authentication method. A cloud user first registers to the cloud provider with user data, such as username, password, email, phone number, credit card details etc [30]. Once the registration is complete, the user is able to access the cloud resources with the username and password provided during registration. This method is easy to implement for the cloud provider and is familiar to lots of users. However, it is not very secure authentication method, since, the security of this method depends on the length and characteristics of passwords. Even if the password is complex, it can be stolen by guessing, brute force. For example, around 21 million accounts were comprised by massive brute force attack on Alibaba's e-commerce site TaoBao [71]. Besides this, complex passwords are difficult to remember, therefore users end up using the same password for multiple cloud provider or use password manager which again increases the chances of password attacks.

2.1.2 Public Key Infrastructure

Public Key Infrastructure (PKI) authentication is based on the cryptographic private-public key generated by a cloud user where the private key remains only with the cloud user while the public key is distributed to the cloud provider [30]. The private key is used to prove the identity of the user. The PKI is also used for the security protocols such as the Secure Socket Layer (SSL/TLS) and Secure Electronics Transaction (SET) with an aim of authentication and data confidentiality, data integrity and non-repudiation. Compared to username and password authentication mechanism, this method provides better security since private-public key is generated cryptographically and cannot be easily cracked. However, this method is not easy for most of the basic cloud users since it requires knowledge of generating the keypair and distributing to the cloud provider. Apart from that, in many deployments there are chances to steal the private key and the hackers are even able to crack it.

2.1.3 Biometrics

Biometrics authentication is one of the most advanced authentication method which uses biometric such as a measurable behavioral trait or physiological characteristics for user's authenticity [82]. The behavioral traits are signature recognition, voice recognition, keystroke dynamics and gait analysis. The physiological characteristics, for example, are fingerprints, iris and retina scans, face, finger, hand recognition. These characteristics or traits are unique in every individual and thus the authenticity can be proved only by the owner. Moreover, it is highly secure compared to other authentication methods as it is very difficult to steal or crack the user characteristics. Biometrics authentication is easy to use, avoids memorizing a password. There is no need of any token and reduces user identity fraud. However, it is expensive to implement as it requires a specific set of hardware and software as well as the accuracy of this technology is still an issue.

2.1.4 Multi-Factor

Multi-Factor authentication is an advanced method of authentication which uses combination of what-you-know and what-you-have or what-you-know and what-you-are to provide authenticity of the user [30]. For example, the user might user ATM card which is PIN or fingerprint protected. Similarly, the user might login into website with a password and pass-phrase from linked hardware (mobile) device. This method is also called two-factor authentication. It is user friendly but requires higher deployment cost.

2.2 Authorization

Authorization is process of determining whether a cloud user has authority to access the requested content or issue certain commands [42]. It is tightly coupled with the authentication as the user must be authenticated in order to get authorized. Authorization is required because in a cloud environment, same physical resource might be consumed by different cloud users who have different access control rights. Thus, cloud environment must have policy to allow access to resources which belong to the user. The most popular authorization system is eXtensible Access Control Markup Language (XACML) and the most popular authorization framework is OAuth 2. These are described in next subsections respectively.

2.2.1 XACML

eXtensible Access Control Markup Language (XACML) is one of the main standards adopted for authorization system. It defines a declarative fine grained attribute-based access control policy language [42, 34]. XACML has four main components: Policy Administration Point (PAP), Policy Decision Point (PDP), Policy Enforcement Point (PEP) and Policy Information Point (PIP) [42, 68]. PAP creates and manages policies to common central repository and PDP is responsible for storing and analyzing policy information from user request. PEP is responsible for authorization decisions based on policies stored in the common repository while PIP provides additional attribute values such as action, resources. These components interact with each other with XACML request-response protocols. The components are deployed and managed by the cloud provider to their servers. Moreover, XACML has three main elements: Rule, Policy and PolicySet [68]. Rule is unit of the policy and PAP combines rules to form Policy and PolicySet contains set of policies. Furthermore, PAP assigns certain role to the users on registration when resource is requested. The role is checked against assigned role from the repository by PDP. On success the user gets response with attributes and is able to access resources otherwise the access to resource is denied.

2.2.2 OAuth 2.0

OAuth 2.0 is authorization framework that allows a third-party application to obtain limited access to a resource on behalf of resource owner and with owner consent [43]. The access is requested by the client or third party-application which can be a web service or a mobile application. For example, a web service might use facebook login which basically allows the user (resource owner) to use facebook credential to authenticate and authorize the user for accessing the web service resources. This framework is successor of OAuth 1.0 and is used by companies like Facebook, Google and Microsoft [64].

OAuth 2.0 defines four roles: resource owner, resource server, client and authorization server [51] [43]. Resource owner is an entity/person who grants access to the a protected resource and resource server is a server which hosts the protected resource. The client is an application which accesses the resource on behalf of the user. The authorization server is the server which issues the access token to the client after successful authentication of the resource owner. This token acts as user identity and is valid for certain interval of time, like 24 hours.

The authorization flow starts with trigger from resource owner who wants to access the protected resources [51, 43]. The trigger is followed by client (application) which asks authorization from the resource owner and returns the authorization grant on success. On success, the authorization server returns the access token and this access token is then sent to the resource server which validates the token and on success returns the protected resource. Thus, a resource owner (user) is able to access the protected resource without providing credential to the client.

2.3 Accounting

Accounting is measurement of resources consumed by a cloud user during certain interval of time [56, 59]. The resources measurement is amount of computing, network or disk space consumed. Accounting is performed by logging of session statistics and resource usage information and is used for authorization controls, trend analysis, auditing and billing. Furthermore, the common business model of accounting is pay-per-use basis where the user periodically pays for the resource s/he has consumed. Pay-per-use basis has resource accounting model which describes all the chargeable resources of the provider and how billing charges are calculated over resource usage. This accounting model is provider specific provided in their web-pages. This accounting model can be used by the user to perform their weekly or monthly or annual accounting estimates.

Accounting system is composed of three basic services: metering, accounting and billing as shown in the Figure 2 [56]. The accounting system intercepts the traffic between client and server as shown in the Figure 1. This traffic is received by metering service which extract the relevant data required for calculating resource usage. These relevant data are called metering data which is forwarded to the accounting services as shown in the Figure 2. The accounting service analyzes and computes the resource usage based on the accounting model and generates accounting data. Finally, the accounting data is used by billing service to generate billing data which is sent to the user as receipt or invoice and the user pays this invoice using the credit card information provided during their registration.

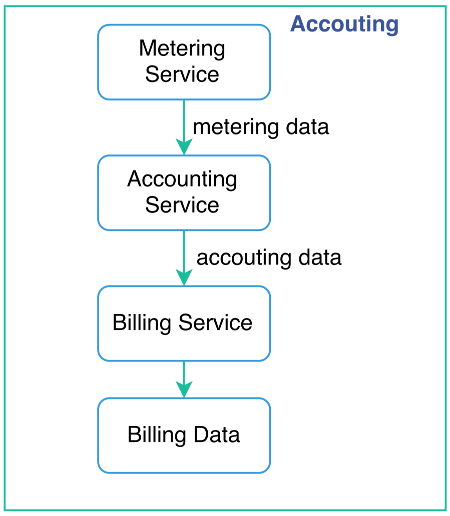


Figure 2: General Accounting Architecture

2.4 Potential Vulnerabilities

Although AAA framework has been used over a decade by cloud providers, it has various major potential vulnerabilities such as distributed denial of service (DDoS) attacks, brute force, man-in-the-middle (MITM), account hijacking and data breach [35, 27, 42]. According to 2014 McAfee (a computer security software company) report of Net Losses and the global cost of cybercrime, there would be the global economic loss of between \$375 to \$575 billion each year which is more than the national incomes of most countries and governments [23].

2.4.1 Account Hijacking

Account Hijacking is an attack where a malicious user (attacker) is able to retrieve a cloud user credential and use it on attacker's favor [27, 42]. With this attack, an attacker can eavesdrop on the user transactions and activity, modify their information, redirect to unauthorized sites and return fabricated data. What makes this attack possible is that sometime the user uses same credential into multiple cloud services which has weak password and security. As a result, the attacker is able to use brute force or guess user's password and get access to their accounts. Likewise, an attacker can even cause financial loss by modifying bank transactions and moving money to attacker's account. For example, Github was hit by massive password guessing attack [37]. Similarly, according to Forbes, hackers hijacked phone numbers and broke into email and bank accounts [73]. This attack can be avoided mostly from the user end, by ensuring the user uses strong two-factor authentication and cloud provider effectively detects unauthorized activities.

2.4.2 Distributed Denial of Service attack

Distributed Denial of Service is an attack where a group of compromised malicious users flood a cloud provider with unnecessary request traffic. The attack results in the denial of service for the legitimate user [35, 27]. The unnecessary traffic might slow down or even crash and shut down the provider services. The main intention of this attack is to disrupt the provider service and prevent legitimate user from accessing the cloud services. So far, one of the latest DDoS attack was on October 21, 2016 when Mirai botnet crippled many of American sites such as Twitter, the Guardian, Netflix, Reddit, CNN [84]. The cloud provider might use intrusion detection or prevention system and increase the number of critical resources to prevent the attack. However, it is almost impossible to avoid this attack since it is very difficult for cloud services to differentiate between good and bad traffic.

2.4.3 Man-in-the-Middle attack

Man-in-the-Middle attack is an attack where a malicious user (attacker) can intercept and modify the communication between two systems [24]. An attacker listens to the traffic between user and cloud service and eventually splits the original traffic into 2 new connections where one connection is between user and attacker and another between the attacker and the service. Thus, the attacker acts as a proxy and is able to read, insert and modify the intercepted data. For example, a researcher from Chinese University of Hong Kong has found flaws in OAuth 2.0 protocol which allow attackers to sign into a billion mobile app accounts using MITM attack [86]. The possible solution for this attack is to ensure the cloud provider encrypts all the data over the network, user uses virtual private network and secure shell to access their resources.

2.4.4 Data Breach

Data breach has been one of the primary cause of online cyber-theft, account hijacking and fraud in cloud computing [35]. The primary reason for data breach are the centralized servers of the cloud providers which creates a honeypot for hackers where they are able to execute multiple cloud vulnerabilities such as phishing, denial-of-service, backdoors, spoofing, clickjacking, MITM [35, 27]. Once, these exploits are successful, hackers can gain full or partial access to user data and end up exposing user data to Internet. Some of the recent breaches are: one billion Yahoo! accounts were compromised in August 2013 [77], LinkedIn reported 117 million accounts details were sold on data sharing website in a possible security breach [66]and hackers launched attack against Sony's PlayStation Network using Amazon EC2 service and stole user data [35].

2.4.5 Malicious Insiders

Malicious Insiders are threats which occur because of lack of transparency in the cloud provider process and policy compliance [27]. A cloud provider might monitor their employees or users without their consent and grant access to their data to third party organization. This leads to corporate espionage and unnecessary surveillance of users. This threat can occur, both from inside and outside of the organization. According to the 2012 CyberSecurity Watch Survey 53% of threats were caused from outside organizations, 24% from inside organization and 17% was unknown or unreported. So far, one of the most alarming recent malicious insider threat was when NSA was able to run Prism program which could monitor theoretically any users who belonged to Giant cloud provider like Apple, Google, AWS, Facebook, Twitter [41]. Similarly, United State Domain Name Service Provider was able to shut down Wikileaks.org for leaking confidential data [60]. The possible solution to this threat is to ensure maximum level of transparency in information security practices, compliance, processing, determining and reporting any security breaches to the user.

2.5 Drawbacks

The major drawbacks of the current AAA framework are user data ownership and high cost to deploy and operate AAA systems [35, 27]. The main reasons behind these drawbacks are the AAA's client-server architecture adopted by the cloud providers and service level agreement (SLA). SLA is an agreement between the cloud provider and the user which defines the terms and conditions for the provisioning and delivery of the services including security measures [29, 58]. Also with SLA, the right to use user data is granted to the cloud providers. Hence, with client-server architecture the user data gets saved to the central servers and with SLA the cloud providers have full access to these data which they are able to monitor, trace, leak and control at their will as described in section 2.4.5. This results to the potential user data vulnerabilities and fraud as discussed in above Subsection 2.4. Thus, though the data belongs to the user, it is eventually controlled by the providers which is one of crucial issues of data privacy and security [52].

Moreover, the SLA requires the provider to guarantee no downtime in the cloud services [29]. So, in order to achieve this, the provider has to maintain data centers which would ensure almost 100% uptime for the cloud services. The maintenance and operation of these data centers is expensive as it requires lots of hardware for computing, network, storage, firewall as well as cooling system with operations to ensure 24/7 support of the system.

Furthermore, the cloud provider has to maintain multiple servers to provide fault-tolerant, distributed and decentralized system so that there is no data loss and the system can be recovered on hardware or software fault. Thus, the current AAA system is expensive to deploy and operate.

3 Blockchain

Blockchain is append-only digital decentralized, distributed ledger. It has network of computers which maintain and validate transactions via consensus with cryptographic audit trails. It was first described by Satoshi Nakamoto in 2008 [61]. Satoshi primarily designed Blockchain as a foundation for cryptocurrency technology such as Bitcoin 4.1. The core idea of blockchain is fast, cheap, secure, reliable, transparent and trustworthy movement of assets between two parties, without any trusted third party such as bank, credit card company, notary [80]. The digital assets could be money, notary documents, properties, contracts. For example, movement of assets is needed for taxes, paying salary, bills, showing possession of particular documents.

Blockchain is based on distributed ledger shared by all the participants in the network and on consensus protocol by which the majority of participants agree on a conclusion [80]. Each created transaction is verified and validated by the majority of participants in the network. The transaction is chained together on spending basis and added to ledger which can never be erased. The ledger is not owned by a central authority or central servers. Rather it is distributed to nodes (computers in the network) over the decentralized network. So, all the nodes on the network have an exact same copy of the ledger. It can be viewed, verified and validated by anyone in the network at any given time. Moreover, the transaction can be even traced to the genesis of transaction block. This essentially removes the dependency to a centralized server and decreases the chances of fraud.

Blockchain can be either private or public. Public blockchain is unpermissioned, allowing anybody to use it. An example is the bitcoin blockchain 4.1. Private blockchain is permissioned and created for a closed group of people working in a certain organization or supply chain. For example, a supply chain company might use private blockchain for their transactions.

Blockchain users range from enthusiast to financial and commercial enterprises. According to survey by World Economic Forum's (WCF) Global Agenda Council, currently $20 billion global gross domestic product (GDP) has been held by Blockchain [45]. But, WCF suggests there is going to be a significant increase in Blockchain growth. So far, blockchain has been very popular in financial, commercial sectors and has gained rapid growth in developing countries.

Financial sector companies such as Nasdaq, Axoni, Deloitte, Finetch, The Linux Foundation, IBM are leveraging the secure, reliability, transparency and the ability to remove the middle man for their advantages [36]. Nasdaq is using blockchain to record private security transactions. Axoni is managing the post trade for credit default swaps along with Depositary Trust and Clearing Corporation. Deloitte is working with startups and customers to develop Smart Identity for banking client. Linux Foundation has an open source collaborative project called Hyperledger for business. IBM has various open source projects and is building the foundation of standardized, production grade digital ledger. Finetch startup backed by 40 global banks is developing a standardized architecture for private ledgers that could significantly cut the cost and time to settle transactions.

In the commercial sector, Everledger and Factom have been actively working with blockchain [36]. Everledger is focused on identity and legitimacy of objects with blockchain. It uses immutable history of transactions and consensus process to provide trust of transactions to its customers. It provides distributed ledger of diamond ownership, which assists in the prevention of fraud in the supply chain. Simlary, Factom is working on securing data with blockchain. It has been funded by U.S Department of Homeland to capture the data from border devices. Moreover, it is working on numerous projects in China to build smart cities. These cities are integrating blockchain with electronic data, notary services, financial transactions to ensure the integrity of data.

In developing countries, on one hand the blockchain can be used for fast and reliable transfer of money from abroad [80]. On the other hand, it can be used to develop trust transactions. Word bank and United Nation (U.N) have initiated various projects for digital identity to be used for land registration and finances for small and medium sized enterprise. Beside these, numerous leading universities such as as MIT [78], Princeton [62], Berkley [4] are researching and supporting blockchain. There is also W3 [5] blockchain community which ensures the guidelines for message formats, public-private blockchain, side chain and evaluates new technologies related to blockchain.

3.1 Terminology

This subsection describes about basic terminologies used in blockchain technology. These terminologies are Peer-to-Peer (P2P) network, block, blockchain, distributed blockchain and smart contracts.

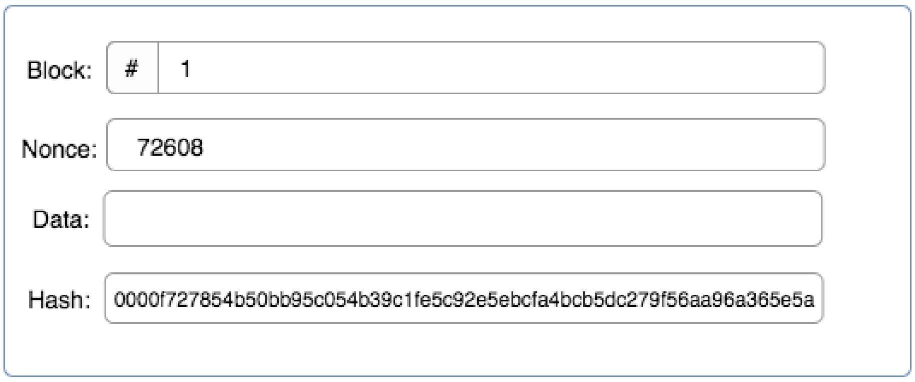
3.1.1 Peer-to-Peer Network

Peer-to-peer (P2P) is distributed network architecture [75] where each participating node (computer) share its hardware resource such as computing, storage capacity, network links with each other. Moreover, these resources are used to provide services like content and file sharing and are available to all nodes directly without need of any central server. Besides this, at given time, a node can be consumer as well producer of the resources. Furthermore, it is also used for anonymising routing of network traffic, parallel computing, distributed file storage, media sharing.

Blockchain uses P2P network architecture to ensure distributed, decentralized networking with no single point of failure [62, Chapter 5]. It has two types of peers in the network: member peers and validator peers. Member peers consume the blockchain services while validator peers are special peers which consume the blockchain services as well as validates and verifies the new transactions in the blockchain at the cost of financial benefit. These special peers are called miners. Each miner has exact same copy of the transaction history across network and has specific responsibility to maintain and propagate the new transaction block across the network. The first miner to successfully validate the block and propagate the result across network gets financial reward.

3.1.2 Block

A block is a single unit in the blockchain (described in subsection 3.1.3) [78] which is building block of blockchain and is composed of transactions with meta-data as shown in Figure 3. A miner (described in 3.1.1) collects the valid data (transactions) of certain time interval to form a block and calculate the cryptographic hash. However, this hash has to be specific format such as the hash must have leading four zeros as shown in the Figure 3. In order to get this specific type of hash, the miner has to randomly guess an arbitrary number which outputs the hash with four leading zeros. This arbitrary number is called number used once or number once (nonce) and the block with nonce is called signed block else it is unsigned. Also, the process of finding nonce is called mining.



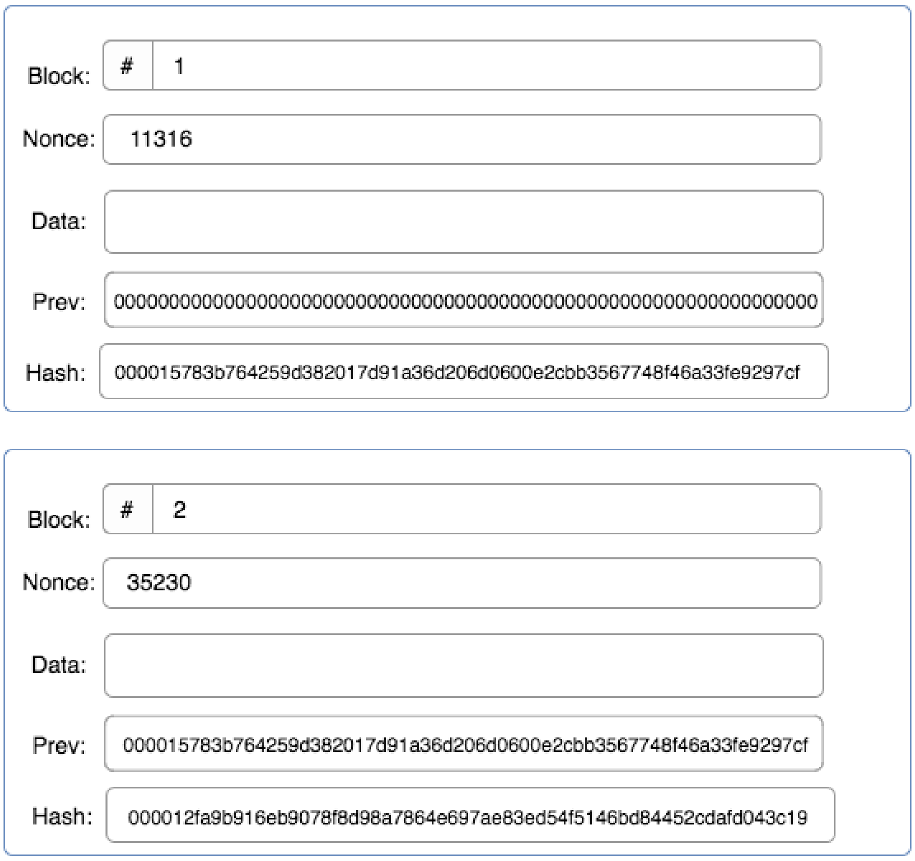
Block:	# 1
Nonce:	72608
Data:	
Hash:	0000f727854b50bb95c054b39c1fe5c92e5ebcfa4bcb5dc279f56aa96a365e5a

Figure 3: Block

A sample block is shown in Figure 3 where a block has a block number, nonce, data and hash. The block number #1 is the unique id of that block and Nonce 72608 is random arbitrary number guessed to find the specific format of the hash. Finally, the Data is the user digital data which is empty in this case and the Hash is the digital fingerprint of the data.

3.1.3 Blockchain

Blockchain is a data structure with linked lists of hash pointers [85]. It is a chain of blocks where each block has a hash pointer to the previous block. This hash pointer allows to verify and validate the digest of previous data. If any value in the chain is changed the digest of that block and the hash pointer of following blocks will change. Thus, this creates tamper-evident log which cannot be changed. Moreover, the hash pointer can be followed till the very first block called the genesis block of blockchain.



Block:	# 1
Nonce:	11316
Data:	
Prev:	0000000000000000000000000000000000000000000000000000000000000000
Hash:	000015783b764259d382017d91a36d206d0600e2cbb3567748f46a33fe9297cf

Block:	# 2
Nonce:	35230
Data:	
Prev:	000015783b764259d382017d91a36d206d0600e2cbb3567748f46a33fe9297cf
Hash:	000012fa9b916eb9078f8d98a7864e697ae83ed54f5146bd84452cdafd043c19

Figure 4: Blockchain

A sample blockchain is shown in figure 4 where two blocks are chained together to form a blockchain. Here, each block has a block number, nonce,

data, previous block hash and current block hash. Moreover, Block #1 is the genesis block of the blockchain and the hash of previous hash pointer is null (00000000000000000000000000000000000000000000000000000000000000
00) integer while Block #2 as shown in figure 4 is the second block of the blockchain which has the hash pointer of the previous block.

3.1.4 Distributed Blockchain

Distributed Blockchain is blockchain distributed over P2P blockchain nodes where all the nodes have an exact copy of blockchain. Thus, if one entry in blockchain is modified and re-mined, the resulting hash becomes different compared to other nodes. As a result, this transaction will get invalidated since other nodes will invalidate this copy. However, a miner could theoretically modify one blockchain entry and re-mine all hashes entry over distributed nodes if it has more computational power than other miners combined.

3.1.5 Smart Contracts

Smart contracts are self executing autonomous computer programs that get executed based on condition defined by programmer [36]. These contracts are capable of facilitating, enforcing and executing agreements between two parties using blockchain. Unlike traditional contracts, where a third party (bank, notary) is required, smart contracts enable independent business between anonymous parties with cheaper fees. For example, one can pay room rent automatically at the end of month without involving a bank in between.

Smart contracts have various possible applications such as trading or loaning of properties, stock or bond trading in distributed markets [36]. Moreover, it can be also used for autonomous transparent digital voting system or autonomous digital notary contract system. In this context, companies like Ethereum, Codius are enabling smart contracts using blockchain to support these applications.

3.2 Cryptography

This subsection describes about the basic cryptography technologies used in the blockchain. They are cryptographic hash function, hash pointer, digital signature and Merkle tree.

3.2.1 Cryptographic Hash Function

Cryptographic hash function is a mathematical function that takes any input string (data) of any length and outputs fixed sized alphanumeric string [62, Chapter 1]. The output string is called hash value or digest or digital fingerprint or checksum. Moreover, the output is of fixed length and unique. The function always produces the same hash from the same data despite the number of times recalculated. The hash cannot be reversed to get the input data and therefore, it can be used to check the integrity of data. Thus, it is also known as one way hash function.

The hash function has three main properties, namely: collision free, hiding and puzzle friendly [62, Chapter 1]. Collision free means it is extremely unlikely to find two different messages that have the same hash. For example, the hash of a string x and the hash of string y are always different, despite how many times it is calculated. Hiding means it is infeasible to find x from given hash of x and puzzle friendly means it is easy enough to calculate a hash of given data.

Blockchain uses Secure Hash Algorithm (SHA) such as SHA-2/SHA-256 which was developed by the National Security Agency (NSA) in 2001 to replace its predecessor SHA-1 to prevent collision attacks [38]. The latest SHA-1 collision attack was reported by Google, which proved two different documents can have same SHA-1 hash signature [76]. Furthermore, SHA-2 is widely used algorithm and so far no vulnerability has been reported. The following is an example, of SHA-256: hash of hello is
2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c1fa7425e73043362938b9824.

3.2.2 Hash Pointer

Hash pointer is a pointer to where data is stored along with digest of that data [62, Chapter 1]. In other words, it is just a hash that is used to reference another piece of known information which can be used to verify the data digest (data has changed or not). The hash pointer can be used to build data structures like blockchain (described in 3.1.3) which is a linked list of hash pointers and Merkle tree (described in 3.2.4) which is a binary tree of hash pointers.

3.2.3 Digital signature

Digital signature is another building element of the blockchain. It uses public-key cryptography to provide the integrity, nonrepudation (obligation of message sent and received by the parties) and authenticity of a message and its source [81]. It has similar properties as a manual signature which can be issued only by the issuer and which is verifiable by other users. A message signed with a digital signature can be verified by other users, but the message can be signed only by signature owner. Beside this, digital signatures are created using public key cryptography. Public key cryptography or asymmetric cryptography uses a key which is a combination of public and private key. The private key is saved only by the owner while the public key is distributed to the other users. The other users can encrypt the message with the owner's public key and the message can only be decrypted by the owner with his/her private key.

Blockchain uses digital signature algorithm such as elliptic curve digital signature algorithm (ECDSA) to generate digital signature [40]. It has three steps to create, sign and verify message with digital signature [81]. The secret key (sk) and public key (pk) are generated by generateKeys method which takes key size as a parameter. The sk is kept only by the owner and pk is distributed over blockchain nodes. The message is signed by using the sk. The sign method takes sk and message as input and generates the signature of the message. This signature can be verified with nodes using verify method which take pk, message and signature as input. If it returns true the message is verified otherwise it is invalidated. Thus, the public key ensures the message has been created by the signature owner and with message verification, user identity is verified. Hence, the public key is used as user identity in the blockchain.

Using distributed blockchain, the users do not need to provide social security number, phone number, email to any central server or authority. They can create their digital identity themselves and distribute their public key to the distributed network. This provides distributed decentralized anonymous identity management to the users. However, in the distributed blockchain all the nodes have copies of all the transactions as described in section 3.1.4. This means a node can see all the history of all transactions. Thus, a node can observe the history of the user transaction and might be able to link or guess the real world identity. Therefore, blockchain provides pseudoanonymity rather than real anonymity.

3.2.4 Merkle Tree

Blockchain uses P2P network where each peer must have same copy of data and new data must be propagated and verified across the network [62]. Propagating and verifying data over P2P network is time consuming and computationally expensive. Therefore, Merkle tree is used which instead of sending data only the hash of the data is sent and the receiver peer checks the hash against the root of the Merkle tree which allows secure and efficient verification of larger data structures as well as ensures data integrity. Merkle tree or hash tree is binary tree of hash pointers 3.2.2 which ensures all the peers/nodes must have same undamaged, unaltered, legitimate data and if a data is changed in one node, changes must be propagated to every node.

Merkle tree is composed of a large number of blocks containing data or transactions as shown in the Figure 5 [32]. These blocks form the leaves of the Merkle tree and the transaction blocks are grouped into pairs of two where each pair has hash pointers respectively which eventually, make the next level up of the tree. Moreover, this process is repeated until the single block is reached as shown in the Figure 5 and the single block is called root

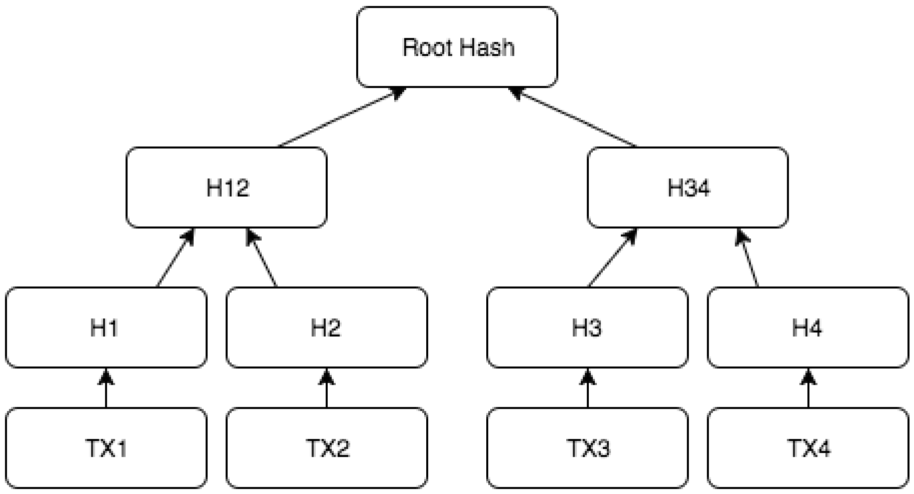


Figure 5: Merkle Tree

hash or the root of the tree.

A model of Merkle tree and basic concept on how root hash is formed is illustrated in figure 5. It has four transactions TX1, TX2, TX3 and TX4 at the bottom where each of the four transaction data pass through a hash function to generate four unique hashes. Moreover, pairs of hashes are then combined and passed though hash function which generates two unique hashes and the two hashes are then combined and again passed to hash function which generates one unique hash which results in the root hash forming complete Merkle tree.

Merkle tree allows detection of any changes to any data within the transaction of block by simply rerunning through the process of transaction and comparing the result of original hash [62, Chapter 1]. If a malicious user tries to change or swap a transaction at the bottom, it will cause changes in the hash of node above, and so on and finally change the root of the tree. In other words, the hash of the block becomes different resulting completely new block which becomes invalid block. Now, this new block needs to be propagated to the other peers. Since, the data of this block is different, other peers can easily invalidate it by comparing with their own hash. Therefore, any changes to any hash of Merkle tree eventually lead to an inconsistent tree.

3.3 Distributed Consensus

Distributed Consensus is the backbone of blockchain (distributed ledger system) [62]. Blockchain uses distributed consensus protocol to reach an agreement on which transactions should be added to the distributed ledger.

No single node or authority can decide which transaction should be added to the ledger. Rather majority of honest nodes in the network participate to reach consensus. This protocol allows to connect devices to work together as a group and in case some node fails, it is still able to reach consensus. Miners solve complex mathematical problem to reach consensus. There are cases when two miners might solve the mathematical problem at almost same time. This results to fork in the blockchain. In this case, the longest branch of the blockchain is selected and others are discarded. Hence, the protocol ensures fault tolerance with built in redundancy and decentralized governance of the transactions.

3.3.1 Proof-of-work

Proof-of-work was first introduced in 1999 by Miguel Castro and Barbara Liskov [33] as the solution to The Byzantine general problem [72, 63]. The Byzantine general problem can be described as follows. A group of generals of the Byzantine army surround the enemy city and are allowed to communicate with each other only via messengers. Therefore, to conquer the battle, they have to reach consensus on a battle plan. However, the problem is that one or more generals might be traitors, hence, they might lose the battle. The question is how many traitorous generals can the army have and still function as a unified force. Similarly, in a distributed system, it can be studied as how many faulty nodes there can be and still reach the consensus. The solution to this problem was proposed with practical Byzantine fault tolerant (PBFT) algorithm [33]. This algorithm suggest secure and fast exchange of message between general to reach consensus. These messages are secured, fault-tolerant and resilient.

Blockchain proof-of-work was first proposed by Satoshi Nakamoto in 2008 to address the double-spending problem [61]. A double-spending problem is a case where a user tries to use the available cryptocurrency more than once. This problem can be solved in a distributed system using proof-of-work where the P2P distributed timestamp server generates a hash of proof-of-work. This form a record that cannot be changed without redoing the proof-of-work. As long as, the majority of CPU is controlled by honest nodes, they reach common agreement forming the chain of honest consensus. This chain is longer than dishonest consensus chain as described in Subsection 3.3. Hence, the honest nodes out-space attacker and proof-of-work ensures the cryptocurrency is spent once.

Proof-of-work reach consensus and add a block to blockchain [72]. A block is added to blockchain after nodes achieve consensus on the solution to a complex mathematical problem which is solved by nodes with repeated random guessing. The first node to solve this problem and broadcast to the network gets the financial reward while other nodes update their ledger since the problem has been solved. They start validating new transaction to get financial reward. The solution of the mathematical problem is difficult to find, it is very easy for other nodes to validate the solution. The situation is identical to number locked bag where it requires multiple repeated random guess to find the correct number sequence of a number locked bag. But, once the sequence is found it is very easy for others to validate by checking if the lock can be unlocked with the correct sequence number.

Proof-of-work has some known issues [62]. For example, lots of computational power is wasted to calculate possible solution which leads to high energy cost for miners and eventually ends up being expensive. Also, only nodes with high computational power can participate in mining. As a result, only limited miners are able to participate in the mining which goes against the idea of decentralization and hence increase the chances of 51% attack (described in Subsection 3.6.1). Besides this, in future miners will get only transaction fees form the users. It will decrease the numbers of miners and this implies that the difficulty to solve the computational challenge will decrease. It follows that the system becomes more vulnerable to 51% attack (described in Subsection 3.6.1). The alternate to proof-of-work is to use proof-of-stake.

3.3.2 Proof-of-stake

Proof-of-stake is a consensus algorithm proposed in 2012 [72] as an alternative algorithm to proof-of-work. It is used to validate a block of transactions in the blockchain network and has a mechanism to punish the nodes that do not follow the consensus protocol. A miner has to bet or put stakes of predefined amount of digital assets for consensus outcome. Unlike proof-of-work, this algorithm chooses randomly a miner from mining pool and the chosen miner is required to solve a simple mathematical problem. Then, if the miner successfully solves the problem an interest or bonus is given on their stake. Otherwise, next miner is chosen randomly. Hence, there is no race to solve the mathematical problem to get economic incentive.

The main advantages of proof-of-stake over proof-of-work are reduced energy consumption and more decentralization resulting decrease in chances of 51% attack [72]. Since, proof-of-stake has only a simple mathematical problem to be solved, miners do not need high end computers to participate in the mining. Rather a less powerful computer is enough. So, far less energy is wasted and there is no fierce competition on building nodes with high computational power to get economic incentive. Moreover, almost any node can participate in mining. Thus, proof-of-stake consumes less energy and motivates wider participation in mining, which increases the decentralization of the blockchain.

3.4 Architecture

The generalized software architecture of the blockchain is shown in the Figure 6 [69]. The architecture is divided into four layers: Application, RPC API, Peer Service and Consensus Layer as shown in the Figure 6.

Application Layer has the blockchain applications which enable users to interact with blockchain network. The blockchain applications are blockchain wallet or blockchain key management software such as ledger wallet [14] which provides blockchain identity to the user by creating public/private key-pair where public key is shared to the network and private key remains only with the user. Moreover, a transaction is created in this layer. The transaction contains sender's digital signature, value of the transaction and receiver's public key as shown in the Figure 6. This transaction is signed by pass-phrase protected sender's private key which creates the unique identifier of the transaction called transaction hash. Depending on the security measure taken by the sender it can also be biometrics (fingerprint, iris) or password protected. Thus, this layer on one hand ensure data integrity of transaction and on other hand provides anonymity to the users. Moreover, the user private keys remains only with the user and no personal data (firstname, lastname, email, phone number) is shared to the network.

Other blockchain applications are web applications such as BitInfoCharts [7] and Blockchain info [3] which provides cryptocurrency statistics as well as interface where any user can query and view transactions. Beside these, the user can develop their own blockchain applications using RPC API Layer.

RPC API Layer is a communication layer between Application Layer and Peer Service layer as shown in the Figure 6. It has the communication and access rules of the blockchain which are pre-defined by the blockchain network. These rules define the endpoint of the database, personal, network, admin, personal, metrics and debug API interfaces of the blockchain. It also sets the rules about which API are exposed to the Application Layer from Peer Service Layer as well as who can access the application. These rules depend on the blockchain environment setup, either development or testing or production. Blockchain specifically uses JSON-RPC API which is simple, transport agnostic, stateless, light-weight remote procedure call [21]. Thus, the Application Layer is able to access only those APIs which are allowed by the RPC API Layer. Furthermore, there are various peers requesting and sending data from blockchain network. There are special nodes called miners that form the Peer Service Layer as shown in the Figure 6.

The Peer Service Layer is the core of the blockchain which performs all the computing necessary for validation and verification of transactions and blocks. Each node has copy of the transaction history which can be verified till the first transaction. This makes the blockchain decentralized, distributed, fault tolerant and resilient since there is no single point of failure and data is distributed over multiple nodes. Moreover, a node in this layer

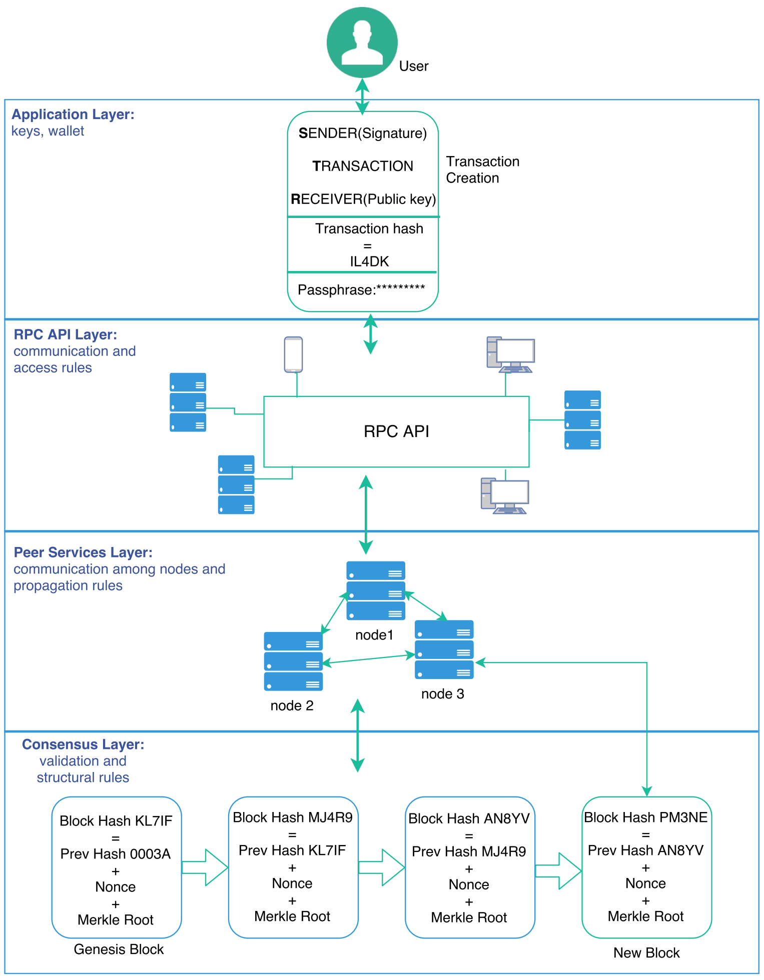

User
Application Layer:
keys, wallet
SENDER(Signature)
TRANSACTION
RECEIVER(Public key)
Transaction hash
=
IL4DK
Passphrase:*********
Transaction Creation
RPC API Layer:
communication and access rules
RPC API
Peer Services Layer:
communication among nodes and propagation rules
node1
node 2
node 3
Consensus Layer:
validation and structural rules
Block Hash KL7IF = Prev Hash 0003A + Nonce + Merkle Root
Genesis Block
Block Hash MJ4R9 = Prev Hash KL7IF + Nonce + Merkle Root
Block Hash AN8YV = Prev Hash MJ4R9 + Nonce + Merkle Root
Block Hash PM3NE = Prev Hash AN8YV + Nonce + Merkle Root
New Block

Figure 6: Blockchain Architecture [69]

validates if the new transaction received from the Application Layer is of right format, is signed by sender digital signature, is feasible (has enough balance to perform transaction). The node also authenticates the sender by checking his digital signature. Finally, the node creates a new block based on the validation and structural rules defined on the Consensus Layer.

The Consensus Layer validates the structure of block created by the node. The block must have a valid block hash, previous block hash, nonce and merkle root as shown in the Figure 6. The block hash is the unique identifier of the new block, previous block hash is the hash pointer to the previous block, nonce relates to proof-of-work and it is result of the solution of mathematical puzzle set by consensus layer. Finally, Merkle root enables verification of all the transaction till first transaction. The node gets financial reward for the proof-of-work and the new block is immediately broadcasted to the network where other nodes updates their own ledger resulting to consensus on the network. Thus, the consensus layer makes the blockchain transparent, immutable audit trail. If one transaction is changed on one node, all other nodes must update this change and re-mine the blocks. Hence, the digital ledger cannot be manipulated and fraud is very difficult.

3.5 How blockchain works

Blockchain work flow has five steps which are Transaction Definition, Transaction Authentication, Block Creation, Block Validation and Block Chaining as shown in the Figure 7 [39, 45]. Transaction Definition is the model of the transaction pre-defined by the blockchain network. It has sender digital signature, the transaction payload and receiver's public key which is cryptographically signed with sender's pass-phrase protected digital key as shown in the Figure 7. Transaction Authentication is the process by which the nodes validate if the A has the asset, enough balance to send the asset and is authenticated to move the asset. Block Creation is process of creating block by a node from the transaction pool where transaction are grouped together based on the creation time. Block validation is process of validating the block by checking if it has previous hash and nonce which provides the proof-of-work. Block Chaining is process of adding the block to the blockchain once the nodes reach on the consensus.

An example of blockchain work, is shown in the Figure 7 where User A transfers a digital asset to User B. The asset can be money or smart contract. In this case, first both users create their digital identity [25] with blockchain wallet. A needs his/her private key and B's public key in order to create Transaction Definition. A receives B's public key either by scanning B's blockchain address QR code or B send the public key to A via email. A creates the transaction which is signed with his/her private key and broadcasts to the blockchain network. This transaction is received by a node which validates and verifies the authenticity of A [72]. If validation fails, transaction is discarded, otherwise it is grouped together with pending transactions from transaction pool and a block is created. This block is propagated to the other nodes and once the network reaches the consensus on block it is added to blockchain and this block becomes permanent after sufficient subsequent blocks get added to the blockchain. Finally, the transaction is confirmed and asset moves from A to B. Furthermore, depending on the blockchain network it might take 2 minutes to 10 minutes for transactions to be confirmed. For example: Bitcoin 4.1 takes on average 10 minutes while Ethereum 4.3 takes 2 minutes to confirm the transactions.

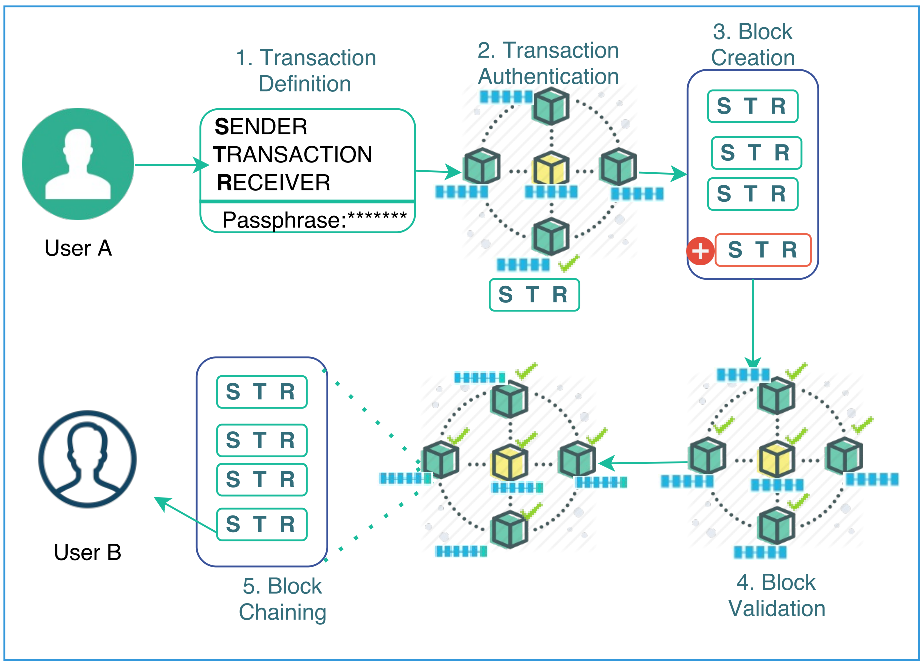


Figure 7: How Blockchain works [39]

3.6 Potential Vulnerabilities

Blockchain is faster and cheaper than centralized system because of its decentralized distributed design. Although, it is reliable and secure because of its consensus protocol, cryptography and anonymity, it still has several potential vulnerability such as the 51% attack, sybil attack, identity theft and code-based attack.

3.6.1 The 51% attack

A 51% attack is when a miner or a mining pool controls 51% of blockchain network computation resources [85]. As a result, they can dominate the validation and verification of transactions as well as they can change the content of blockchain. Moreover, they could invalidate the valid blocks, create and confirm their own fraudulent blocks eventually quicker than the rest of honest miners which could result to double spending. Besides this, the attackers can change the consensus rule, steal assets from others and even prevent cryptocurrency generation.

So far no bad incident has happened with 51% attack although on July 2014, mining pool ghash.io held more than 51% of bitcoin blockchain network [79]. But, shortly after other miners moved out of this mining pool, hence any tragedy was avoided. This attack can be prevented using proof-of-stake consensus algorithm.

3.6.2 Sybil attack

Blockchain does not have a central authority to administer identities of the participants [62]. As a result, attacker can create multiple copies of itself, which might look like separate participants though they are all controlled by the same node. The attacker can try to fill the network with its clients. So, other nodes are likely to connect only to attacker nodes. The attacker can then refuse to relay blocks and transactions from others, disconnect the connecting node from the network or relay only blocks created by itself. This attack can be avoided by only trusting the blockchain with the most proof-of-work since it cannot be easily faked because of the significant mining power requirement.

3.6.3 Identity theft

Although blockchain provides the ownership of the user identity, this digital identity is backed up by the private key that must be kept safely. If the private key is stolen or device storing the private key gets hacked, the victim will lose all its digital assets as well as its digital identity. Moreover, this digital identity cannot be recovered and it will be almost impossible to find the culprit. There are various applications to encrypt and sync the private keys across various devices to recover the private key. But if these applications have some malicious codes or if they get hacked then again the user ends up with identity theft. Apart from that, synchronizing the keys across multiple devices increases the chances of getting hacked.

Besides this, with the rise of quantum computing it may become possible to crack the cryptographic keys used by blockchain technology [36].

3.6.4 System hacking

Blockchain records or data cannot be easily modified or altered. However, the code-base and system which implements blockchain can be modified because, depending on the company or organization, the blockchain code-base can be based on open source [85]. For example, the most popular blockchain applications Bitcoin and Ethereum are open source. Therefore, any user can contribute to the development of these applications and if these contributers provide vulnerable code or there is human-error in the code-base because of the contributor, it might possibly end up in the production system which in turn might cause system hacking. Beside this, a company can fork the blockchain code-base for their own use. If this code-base is poorly maintained, or outdated, it might end up getting hacked. In March 2014, MtGox Tokyo based bitcoin exchange got hacked and 700 millions worth of bitcoin were stolen [85]. Similarly in 2016, hacker exploited software vulnerabilities of Decentralized Autonomous Organization (DAO)(described in Subsection 4.3) and stole 50 million worth of ether (cryptocurrency used in Ethereum blockchain).

3.6.5 Illegal activities

The pseudoanonymity, immutable transaction and decentralized property of the blockchain makes it difficult to monitor and track transactions on blockchain [36]. Moreover, the technology itself is in the early stages of production and the essential regulations for using blockchain application are on early stages. Hence, the system can be misused for money laundering, illegal movement of funds. For example, Silk Road, a website to buy and sell illegal drugs used bitcoin for its payments [44].

4 Blockchain Applications

Blockchain can be used in different application domains such as as financial, non-financial, insurance, Internet of Things (IOT), health-care, Internet, cryptocurrency [36]. Some of the financial applications are Medici, Blockstream, Bitshares and non-financial applications are Stampery, Ascribe, Block Notary. Everldger is Insurance application and IOT Filament, ADEPT platform are IOT applications. Examples of Internet applications are Namecoin, Ethereum. Examples of Cryptocurrency applications are Namecoin, bitcoin. Some of the most popular applications such as bitcoin, Namecoin and Ethereum are described below.

4.1 Bitcoin

Bitcoin is the most popular application developed on blockchain. It was first proposed by Satoshi Nakamoto in 2008 in the paper 'Bitcoin: A Peer-to-Peer Electronic Cash System' that describes a P2P method of sending electronic cash from one person to another without involving any trusted third party [31]. The electronic cash is termed as cryptocurrency. It relies on public-key cryptography to identify users, hashcash algorithm for proof-of-work to detect double spending and consensus algorithm to reach the common agreement on blocks getting added to blockchain.

Bitcoin was developed to address the current financial challenges [62, Chapter 1]. The current finance system is strongly tied with trusted third parties such as banks, credit card companies who relay and process the financial transactions. The trusted third parties validate, safeguard transactions and persist the transaction history which later can be used as proof of financial transactions to avoid frauds. However, the economic collapse in 2008 has shown that these trusted parties can create economic bubble resulting to disasterous economic consequences [70]. Moreover, the trusted third parties are expensive for users to maintain transactions. If the trusted third party is corrupted or hacked users lose their assets. Apart from these issues, user data is owned by trusted third party who can misuse and share it to organizations like National Security Agency (NSA) which may track users.

Bitcoin has been most successful cryptocurrency in the history [31]. The first bitcoin block was mined in January 3, 2009 and on January 9 2009 bitcoin v0.1 was released. The first financial transaction was done in May 2010 where a user bought a pizza with 10,000 bitcoins. Since then, the financial value of bitcoin has exponentially increased compared to fiat currency. The current (May 13, 2017) value of one bitcoin, is equal to $1,716.72 [7]. Nevertheless, it was not until December 2013 when the value of a bitcoin skyrocketed and one bitcoin was equal to $1,100 which caught attention of mainstream investors.

Bitcoin uses P2P networks instead of trusted third party to execute transactions between two users over the Internet [31]. It relies on digital signatures based upon public-key cryptography as described in Section 3.2.3 to establish trust instead of utilizing trusted third party. Each user has set of cryptographic private and public key where the public key broadcasts to all the users in the network while the private key is safely kept by user. So, the user identity remains pseudonymous. However, everyone in the network can see certain Bitcoins were transfers from user A to B.

A bitcoin is sent to the public key of the receiver. This creates a transaction which is protected with the private key of the sender. This is done in such a way that the sender has to prove the ownership of the transaction using his private key and this transaction is verified by miners (described in Section 3.1.1) using sender's public key. Miners ensure the sender has enough bitcoin balance to spend and that the bitcoin is owned by the sender. Moreover, each transaction is broadcasted to all the nodes. After verification and validation the block is recorded to blockchain.

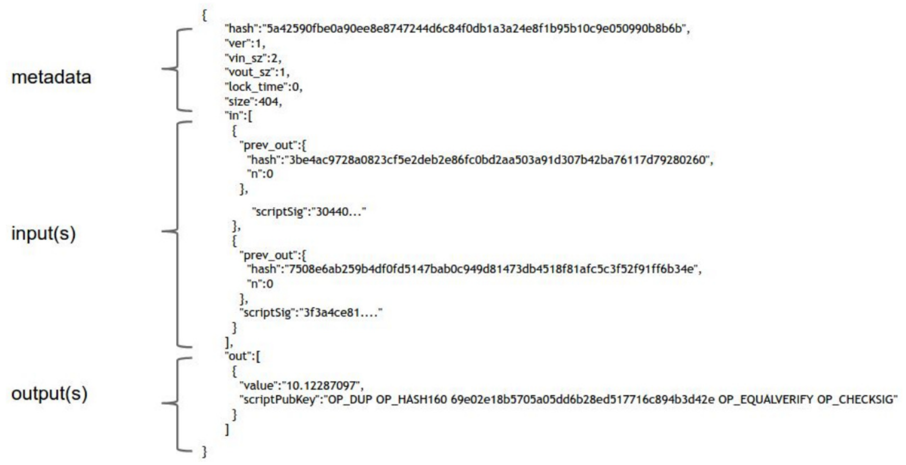
A user can start using bitcoin by creating bitcoin identity. This is done by downloading bitcoin wallet and creating bitcoin account and Bitcoin can be exchanged from fiat currency using bitcoin exchanges such as Bistamp, Bitsquare, Kraken [67, Tab Exchanges].

In addition, the user can develop Bitcoin application like Namecoin (described in Subsection 4.2) by forking the bitcoin core code and modifying it to support own business/use case.

4.1.1 Transactions

Many transactions take place at a given time and any node can collect unconfirmed transactions to form block and broadcast these blocks to the network [31]. Also, the transactions are not ordered on the basis of creation due to propagation delay in P2P network. Therefore, the transactions at given time are grouped together to form block. This block is chained with previous block using previous block hash as described in Section 3.1.2. The block is added to blockchain after miners have solved a complex mathematical problem. The first miner to solve this problem or in other words to be able to generate proof-of-work, broadcast the solution to the network. The average computational effort required for the proof-of-work is high. However, the validation is simple and it is done by executing a single hash function. The difficulty of the problem is adjusted such that, the problem is solved on average every 10 minutes. The miners get financial reward for proof-of-work. This reward was 50 bitcoin in the beginning and it gets halved every four years to ensure limited supply of bitcoin with steady decreasing rate. As a result, the maximum number of Bitcoins will be generated by 2140. Occasionally more than one block will be solved at the same time leading to several possible branches of blockchain, out of which only the longest branch is considered to be valid.

Every bitcoin transaction has three parts, metadata, inputs and outputs as illustrated the figure 8 [62, Chapter 3]. The first part is metadata which has lock time (time to add the transaction to the blockchain), size of transaction and SHA-256 hash of entire transaction. The hash serves as a global unique transaction ID as shown in the Figure 8. The second part is Input which is an array reference to previous transactions with previous transaction hash and its index. It also has scriptSign which contains a signature and the public key as shown in the Figure 8. Finally, the third part is Output which is also an array that has instruction for sending bitcoin to the receiver public key [31].



```
{
  "hash":"5a42590fbe0a90ee8e8747244d6c84f0db1a3a24e8f1b95b10c9e050990b8b6b",
  "ver":1,
  "vin_sz":2,
  "vout_sz":1,
  "lock_time":0,
  "size":404,
  "in":[
   {
    "prev_out":{
     "hash":"3be4ac9728a0823cf5e2deb2e86fc0bd2aa503a91d307b42ba76117d79280260",
     "n":0
    },

      "scriptSig":"30440..."
   },
   {
    "prev_out":{
     "hash":"7508e6ab259b4df0fd5147bab0c949d81473db4518f81afc5c3f52f91ff6b34e",
     "n":0
    },
    "scriptSig":"3f3a4ce81...."
   }
  ],
  "out":[
   {
    "value":"10.12287097",
    "scriptPubKey":"OP_DUP OP_HASH160 69e02e18b5705a05dd6b28ed517716c894b3d42e OP_EQUALVERIFY OP_CHECKSIG"
   }
  ]
}
```

Figure 8: Bitcoin Transaction [62]

Moreover, the Output has an integer value which represents number of cryptocurrency to be sent and scriptPubKey that specifies the ECDSA hash of the public key and a signature validation routine which represents conditions under which this transaction can be redeemed [31]. This cryptocurrency is measured in bitcoin and the smallest unit of bitcoin is called satoshi where 10^8 satoshi is equal to one bitcoin. Output also has a short script snippet: scriptPubKey as shown in the Figure 8 that specifies the ECDSA hash of the public key and a signature validation routine which represents conditions under which this transaction can be redeemed. To successfully redeem the transaction the sciptSig and the scriptPubKey must be executed successfully in the order.

4.1.2 Scripting Language

Bitcoin uses bitcoin scripting language which has been specifically designed and developed for Bitcoin with room for only 256 instructions [31]. The bitcoin has less than 200 commands called opscodes. It is stack based programming language with support for cryptographic operation such as hashing data and verifying signatures. The design of language is ad-hoc, non-Turing, which means it does not have the ability to compute arbitrarily powerful functions and every instruction is executed exactly once in a linear manner. It has no loop support because the miners have to execute scripts submitted by arbitrary participants avoiding any infinite loops. Once the script is executed, there are only two possible outcomes. Either it executes successfully with no errors and transaction is valid or there is an error while the script is executing. In the latter case, the whole transaction will be invalid and is not accepted to the blockchain.

4.1.3 Bitcoin Network

Bitcoin transaction can be executed on three networks, main-net, test-net and private-net [31]. Main-net is the main Bitcoin production network where bitcoin is required for transactions. Test-net is also public Bitcoin network but has different genesis block than main-net and has been reset three times. Test-net does not require real bitcoin and is useful for testing bitcoin applications before deploying to main-net. Main-net and test-net are developed and maintained by Bitcoin core developers while private-net can be deployed by a developer for individual use or company to support their own use case. In addition, mining in the private-net is faster than test-net since there is less users and transactions compared to test-net or main-net.

4.1.4 Advantages

Bitcoin technology has several advantages over existing financial systems which uses a client server architecture where users are registered to the central server and the central server owns the user data [62, Chapter 3]. The Bitcoin technology is secure since transactions are secured by public-key cryptography and trust is established in P2P manner. It is cheaper because the transactions are broadcasted immediately over P2P network and they are propagated as fast as possible to other nodes. Also, the intermediate cost of handling transactions is lower than existing financial system transaction costs. Beside these, the network also makes the transactions distributed and immutable, permanently saved to the distributed ledger. This ledger makes it easier to detect frauds and prove the ownership of the transactions.

4.1.5 Vulnerabilities

Bitcoin has several vulnerabilities (discussed in Subsection 3.6) and limitations such as it is not fully scalable since there is only limited (around 21 million) Bitcoins. It is time consuming to get the transactions confirmed as new blocks are added every 10 minutes to blockchain. Therefore, it takes an hour to get six new blocks added to the blockchain and this is the number of blocks needed before transaction get confirmed.

Bitcoin blockchain size has increased from 50GB to 120GB [7] and it is increasing all the time. Also, the computational power in order to solve the mathematical problem is exponentially increasing all the time. As a result, only certain nodes with high computational power can become miners which in fact defies the principle of distributed system. The energy cost of computational is getting higher all the time. If the hackers steal the user private key, they can steal the identity of user which cannot be recovered and the user loses all her bitcoin. For example, in February 2014 Mt. Gox, the world third largest Bitcoin exchange got 850,000 bitcoins stolen and thus the company had to declare bankruptcy [85].

4.2 Namecoin

Namecoin is an alternative cryptocurrency (alt-coin) based on BitDNS protocol with intend to enable censor-resistance domain name system outside the control of any single entity [46]. The system uses blockchain to manage domain name lookup instead of central authority like Internet Corporation for Assigned Names and Numbers (ICANN) [19]. The latter requires of much trust on a central authority and represent a single point of failure. Namecoin was the first fork of the bitcoin code-base with its own blockchain. It uses Bitcoin core features such as proof-of-work, block creation time and transactions operations with additional features of a name/value store. The name/value store is a blockchain transaction database where user can store arbitrary identity data such as username, email address or website identity. It has been primarily used for the website identities and it enables registration and domain-name resolution for top-level domain (TLD) .bit.

Namecoin squares Zooko Traingle, meaning it makes it possible to have domain name which is human-readable, decentralized and authenticated [46].

Human-readable means a user can pick a name. Decentralized means there is no central trusted party or single point of failure. Authenticated means a strong sense of ownership using cryptographic keypair. Until 2011, designing a system which would exhibit all these three properties was impossible. However, Namecoin was the first system to provide naming system that offered all three properties.

Namecoin cyptocurrency is called namecoin and its unit is represented by NMC. Furthermore, a user can register for .bit TLD at dotbit.me [22] with very a small fee of 0.01 BTC or 5 or 20 NMC at the time of writing. Moreover, similar to bitcoin, Namecoin is also limited to 21 millions coins. The first Namecoin block was mined in April 2011 [46] and as of the time of writing over 327,299 block (which equals to 13,347,132 NMC) have been mined [7, Namecoin tab].

Based on Namecoin, various applications such as OneName and Blockstack has been developed. OneName utilizes the Namecoin blockchain to record data about its members [46] while Blockstack provides similar features with Bitcoin blockchain and introduces separation of control and data plane and additional support of deploying decentralized server-less applications [28].

4.3 Ethereum

Ethereum is the second most popular blockchain application. It was developed to address the weaknesses of the bitcoin. The weaknesses are bitcoin script that has the limit of small instructions and is non-Turing complete. The script is more centered toward bitcoin use case [32]. Developing applications using Bitcoin script requires developers to fork the bitcoin core code-base and add the logic for their own use cases. The forking is time consuming and difficult to maintain. Thus, to address these challenges, Ethereum was developed. The Ethereum provides a platform for programmers to build applications on top of the blockchain called an Ethereum blockchain. It was first proposed in late 2013 by a Bitcoin programmer named Vitalik Buterin in the Whitepaper 'Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform'. This thesis proposes Turing-complete programming language for writing scripts (smart contracts) and Ethereum Virtual Machine (EVM) to execute the smart contracts and transactions.

An Ethereum user can create smart contracts and upload them to the Ethereum Blockchain with a small fee. Other Ethereum users can access these contracts by remote procedure calls provided by Ethereum Application Program Interface (API) [62]. The contracts can store data, send transactions and interact with other contracts. The contracts are executed in bytecode. Once contracts are uploaded to the blockchain, they are stored, executed and interpreted by EVM. EVM requires a small amount of fees to execute transactions. These fees are called gas and the amount of gas depends on the size of instruction. The longer the contract instructions, the more gas is required. In addition, Ethereum has own cryptocurrency called ether and it is presented by the abbreviation ETH. Ether is a type of token that powers applications on the decentralized Ethereum network. The smallest unit of ether is Wei. One ether is equal to 10^{18} Wei. Users can use the Ethereum exchange to change the physical or normal money to ether. At the time of writing (May 13, 2017), exchange value of 1 ether was equal to $86.59 [7].

4.3.1 Ethereum Blockchain and Account

Ethereum blockchain is similar to the bitcoin blockchain with certain differences in the architecture [32]. The differences are listed as following. The Ethereum blockchain contains a copy of both the transaction list and the most recent state. The block creation time is every 12 seconds. Ethereum blockchain uses GHOST protocol to reach consensus branch with proof-of-work called Ethash. The proof-of-work has been planned to be switched to proof-of-stake. Minimum 6 confirmations are required to approve a transaction and a miner gets 5 ether as financial reward which does not half, unlike in bitcoin.

A user needs an account to use Ethereum blockchain [32]. An account has four fields, nonce, ether balance, contract code and storage. The nonce ensures a transaction is processed only once. Ether balance is the amount of money a user has. Contract code contains smart contracts which deployed by the user to the EVM. EVM provides environment to execute the contract and persistent storage for saving data. Ethereum accounts are of two types, externally owned and contract account. The externally owned account is created by user and controlled by private key. The contract account is controlled by the contract code. The accounts generate transactions and messages.

4.3.2 Transactions and Messages

The transaction is a data package signed by sender. It stores data to be sent from an externally owned account or external actor [32]. The transaction contains recipient address, digital signature of the sender, ether amount to be sent to the receiver, data field, startgas and gasprice. Receipt address is receiver public key address and digital signature is used to authenticate the sender. The data field is an optional field and has no function by default. However, it can be used by virtual machine that has opcode (operation code) with which a contract can access the data. Startgas and gasprice are to avoid any accidental infinite loops which might drain computational power. Each transaction has requirement to set a limit of computational steps required to execute a code. The fundamental unit of this computational step is called gas and each computational step requires 1 gas. Moreover, there is a fee for every 5 gas spent and the fee is one ether. Every transactions must specify upfront how much gas it is willing to spend. If the execution runs out of gas it will halt the transaction and consume all the gas. Therefore, it is important to provide enough gas for transaction executions. Ethereum by design is not suitable for computationally heavy contracts as the creation and execution of contracts get very expensive. Creating and executing a contract requires ether, but querying a contract does not require any ether. For example, assume a contract has instruction to register and list user. Then registering user requires ether but listing user does not require any ether.

Unlike transactions, messages are produced by the contract account instead of external owned account [32]. Messages are produced when a contract sends information to the other contract which executes the CALL opcode. The messages only live in the Ethereum execution environment. Similar to transactions, messages also has sender, receipt, ether to be sent, data field and startgas value.

4.3.3 Smart Contracts

Smart contract is the key component of Ethereum. It is a self executing autonomous program that lives in the Ethereum execution environment. It gets executed once triggered by a message or a transaction. It possesses its own ether balance [32]. The smart contract code executes until it reaches an error or STOP or RETURN instruction. These codes have access to three types of space: stack, memory and storage to store data. The stack is last-in-first-out container where values can be pushed and popped. Similarly, memory is an infinitely expandable byte array. Storage is long term key/value pair. Unlike stack and memory which reset after computation ends, the storage persists the data for the long term.

The smart contract is written either using solidity [83] or serpent [48]. Once a program is written using one of these languages, it can be deployed in Ethereum Blockchain using either Mist or Ethereum wallet [9]. Mist and Ethereum wallet is application developed by ethereum foundation. This application allows developers/users to create accounts and ethereum applications. Mist and Wallet application also allows users to test and deploy their ethereum applications to the network (further discussed in Section 11). Furthermore, like in Bitcoin (as described in Subsection 4.1), the contract can be deployed in either main-net or private-net or test-net.

4.3.4 Decentralized Autonomous Organization

Ethereum smart contract and blockchain provide unlimited potential to develop a wide range of decentralized applications (Dapps). Examples are, own cryptocurrency, decentralized autonomous organization (DAO) and decentralized storage system [32]. The own version of cryptocurrency can be used by companies for their internal purposes. The DAO provides decentralized human-less venture capital platform. This platform can be used by investor to invest on startups. The investment is transparent, distributed and decentralized. Decentralized file storage allows users to rent their unused hard drives. Some of the real world applications are Weifund, Uport and Provenance. Weifund leverages smart contracts to provide an open platform for crowd-funding campaigns in Ethereum Ecosystem. Uport provides a platform for users to take complete control over their identity instead of giving the control to the government institutions. Provenance enables tractability in the supply chain business by tracing the product history so that consumers can make informed decisions when they buy products.

4.3.5 Vulnerabilities

Although Ethereum is secure and decentralized it has its own weaknesses. The Ethereum code-base is open source and maintained by group of developers [18]. So, if there is a bug in the code-base or technical flaw, hackers could easily exploit it. For example, in April 2016 The Decentralized Autonomous organization (DAO) was developed by hard fork (no backward compatibility) from Ethereum with the objective to provide decentralized human-less venture capital. But, in May 2016, the vulnerability in the DAO code resulted in loss of $50 million worth of ether.

5 Rational Behind Choice of Technology

As discussed in Subsection 2.4 and 2.5, the current AAA (described in Section 2) has various vulnerabilities and limitations. These vulnerabilities have caused user data hijack and breaches, identity theft and financial loss. These issues are becoming more common and frequent. This has sparked the security concerns over the current AAA framework. The end-users are becoming more concerned about their digital identity and privacy. Beside these issues, repeated user registration across different services is inconvenient. Multiple registrations increases the vulnerabilities of the user data. Thus, an alternative solution is required to address these challenges.

As discussed in Section 3.1.3, Blockchain is technology based on the P2P, consensus protocol and digital signatures. The P2P network is the blockchain network which is by design decentralized, distributed with no single point of failure. The consensus protocol ensures that a transaction (user A sends $1 to user B) happens only once. This transaction is added to public distributed ledgers which cannot be reverted. Also, anyone in the network can validate and verify this transaction. This makes the system transparent and reliable. The user issues their identity with public-private key cryptography. This identity uses digital hash algorithm which is almost impossible to be cracked by current technologies. Moreover, the user's identity as well as data signed by the user can be verified and validated by anyone in the network. But the transactions signed by the private key can be only viewed by the owner. Thus, blockchain ensures the user identity is uncrackable and the user has complete ownership of user data as well anonymity over the network.

Therefore, blockchain decreases the chances of hacking as the data is not shared with the central server. The user data is kept by the user and that data is protected by the latest hash algorithm. This hash algorithm is most advanced hash algorithm and has not been cracked yet. It is easy to use blockchain technology across multiple services. Moreover, user data is only with the user and not sent to central server. These features give the data ownership to the user than service providers. Blockchain also decreases the chances of the user data breach. The breach is only possible with user consent or carelessness. The providers cannot share data with third party organizations as they do not have user data and no control over their data. This technology has its own vulnerability as described in Subsection 3.6. Also, it is not scalable compared to the current system because of the transactions time. But, despite these drawbacks, it is more secure, easy-to-use, trustworthy, reliable, fault-tolerant than the current AAA system as described in Section 2.

It is difficult to develop the applications for blockchain (e.g. bitcoin blockchain) because of the technical depth and architecture. The blockchain architecture is different than most of the existing AAA systems [36, 80]. The technology is in the early stages of the development and lacks the proper development instructions and support. A developer needs to clone the whole blockchain repository and develop the application on top of it with blockchain scripts which is cumbersome and difficult to deploy as described in Subsection 4.1. The user also has to maintain the blockchain and ensure it has the latest changes. The blockchain application must be modified according to these new latest blockchain changes. The blockchain was primarily designed for the Bitcoin. Thus building other types of application based on this design is difficult and challenging.

Ethereum Blockchain is the blockchain platform. It has been designed to develop the blockchain applications on top of the Ethereum blockchain using smart contracts [32, 10]. The smart contracts are written using high-level solidity language. The language is easy to learn and write. It is alike Javascript scripting language. The contracts can be easily deployed to private, test or main network. It is interpreted by the Ethereum Virtual Machine. Therefore, on one hand, Ethereum platform hides most of the technical depth of the blockchain and allows the developer to concentrate on writing his application logic while, on the other hand, it makes the application deployment painless. The large Ethereum community provides active support for the possible issues. For these reasons, Ethereum Blockchain was selected for the prototype development and testing.

6 Prototype Design

The design of prototype is described in this section. The reasoning behind the design and development of prototype as well as the software architecture is explained. It is worth noting that, this prototype is a very basic proof of concept with no strictly defined expectations and the result may vary depending on the which blockchain network is selected.

6.1 Need for Prototype

As described in the Section 1.1, the goal of this thesis is to develop a proof-of-concept on how cloud providers could use one common identity backend to authenticate and authorize users. The proof-of-concept should also provide ownership of user data to the users rather than to cloud providers. The users should be able to pay their invoices without sharing their private financial data.

The actual need of prototype comes from cloud providers having their own identity backends. These identity backends are a single point of failure. They are also vulnerable to different types of attacks and back-door of user data leaks as described in Subsection 2.4. Thus, cloud providers need one common distributed decentralized backend. This backend can authenticate and authorize cloud users with no single point of failure and decrease the possibility of attacks and user data leakages via back-doors.

Another essential purpose of the prototype is the cloud users data ownership. The user data is currently owned by cloud providers once cloud users register for cloud services. Thus, cloud users need a system for their data ownership. The users should be able to use multiple cloud providers with same identity without sharing their private data. This ensures users are able pay their invoices without sharing their user data.

6.2 Software Architecture

The software architecture presented in the Figure 9 shows the complete solution on how the cloud providers could leverage Ethereum blockchain technology for a common identity backend. The solution also presents how cloud users could use blockchain technology for data ownership and pay their invoices without sharing their private financial data. Developing the complete solution as shown in the Figure 9 is out of scope for this thesis. However, the whole complete solution has been discussed from the architecture perspective. Most crucial parts: implementing smart contracts to the blockchain network was implemented and relation between providers, blockchain, smart contract and users was described in detail.

The software architecture has four types of participants : infrastructure providers, cloud providers, Ethereum Blockchain network and a user as shown in the Figure 9. Infrastructure providers are providers which provide infrastructure services such as computing, storage, network to the cloud providers. Cloud providers have the business logic, how to consume and expose the infrastructure services to the end users. In some cases, such as Amazon Web Service (AWS) [1], IBM Bluemix [13], Google Cloud [6], Microsoft Azure [16] cloud provider also provides infrastructure services. Each cloud provider has their own smart contract as shown in the Figure 9, with sets of instruction about how to authenticate and authorize end users. Ethereum Blockchain network provides distributed, decentralized identity backend for the cloud providers. Finally, there is an end user who consumes the infrastructure services through cloud providers' Application Program Interface (API) [2].

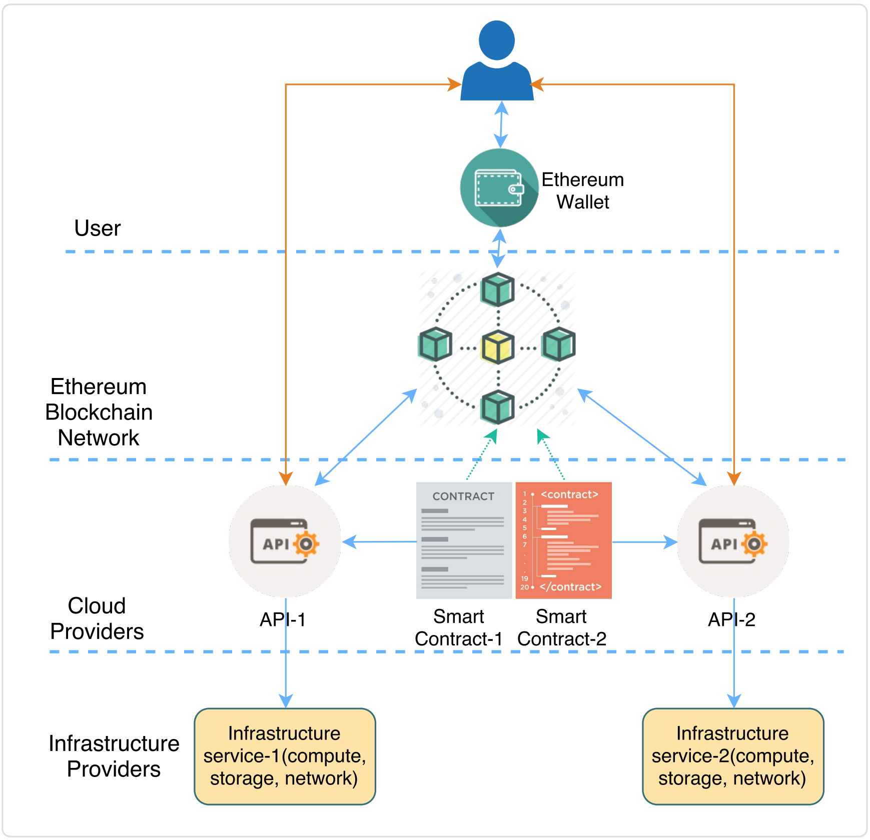


Figure 9: Prototype: Architecture diagram

For simplicity, a user, two infrastructure services 1 and 2 as well as two cloud providers 1 and 2 with their respective smart contracts are considered as shown in the Figure 9. Additionally, the figure can be easily extended for multiple users by adding more users with their own Ethereum wallet, described in subsection 11.

Infrastructure services interact with their respective cloud providers as shown in the Figure 9. Each cloud providers define their authentication and authorization logic with smart contracts. These contracts are deployed to the Blockchain network via their respective API. The user creates their identity with the Ethereum wallet that generates set of private and public key. The wallet stores the private key while deploys the public key to the Ethereum network. Now, the users can access the infrastructure service through the cloud provider API which authenticates and authorize the user with the Blockchain network as shown in the Figure 9.

Multiple cloud providers connect to common identity backend as shown in the Figure 9. The users also connect to the same identity backend. In order for the user to use cloud service, s/he is able to leverage the blockchain for identity without providing any private information to the cloud providers. Also, a user does not need to register for new cloud provider in order to use their services. According to this architecture any cloud provider can basically integrate or connect to the existing infrastructure provider and offer their services without requiring users to register with their services.

6.3 Flow diagram

The flow diagram of the complete solution described in the above section 6.2 is shown in the Figure 10. The flow diagram describes how a cloud user interacts with a cloud provider smart contract which is deployed to the blockchain network. For simplicity, we assume that cloud resources are used. As discussed above, cloud resources are out of scope of this thesis. The key components in the figure are the smart contracts deployed by the cloud provider and a user able to make transactions using the smart contract in the blockchain network.

The cloud user must be authenticated and authorized to access the cloud resources as well as to pay the invoice in cryptocurrency. There are two types of authentication. The cloud user and cloud provider both are authenticated against the blockchain to use cloud resources and execute the transactions respectively. The cloud user authenticity is proved recovering the cloud user public key from the message signature which is signed with cloud user private key. The cloud provider authenticity is proved as all the transactions executed by the cloud provider are signed with the cloud provider private key. These signatures are by default verified by the blockchain before executing the transactions. Hence, this ensures, on one hand, the cloud user is legitimate user and only the right cloud provider is able to run the transaction. The cloud provider authorizes the user by checking if the user address is valid and exists on the blockchain. If the answer is positive, the cloud provider adds the user to its blockchain address database and marks the user as authorized to access its resources. Furthermore, only public key of the cloud provider as well as the public key of the cloud user is distributed to the blockchain. Hence, the prototype maintains the anonymity of the user and provider. Additionally, the user is able to pay the invoice with the cryptocurrency without providing their bank details.

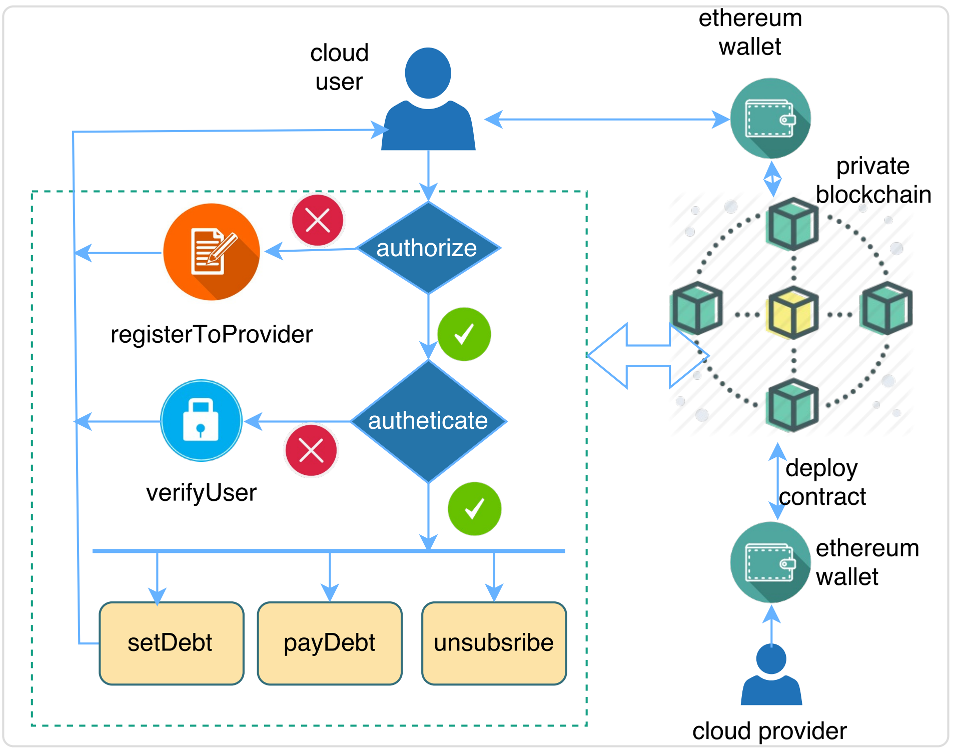


Figure 10: Prototype: Flow diagram

First the cloud provider deploys the contracts to the private blockchain network with ethereum wallet as shown in the Figure 10. Meanwhile, a cloud user creates his/her digital identity using ethereum wallet as shown in the Figure 10. Now, the user can access cloud resources for the purposes such as creating a virtual machine, storage etc. If the user is authenticated and authorized, s/he is able to access the resources. Otherwise, the user needs to do the action registerToProvider. On successful authorization, the user can access resource. The user identity still needs to be verified. On success, the prototype proves the authenticity of the user and the user is authorized and authenticated to access the resources as shown in the Figure 10. The cloud providers keep the track of its resource usage by the user. Eventually the provider sets the debt to the user using setDebt method. The user is able to pay this debt in ether with payDebt method. Furthermore, the user can also de-register or unsubscribe from the cloud provider. This deactivates user from cloud provider and the user needs to register again to access the resource.

7 Prototype Implementation

This section describes the hardware and software components required for implementing the prototype. It also describes the smart contracts, prototype environment setup and how the prototype was executed. The main idea is, after reading this section, it would be possible to setup the development environment and execute the prototype.

7.1 Hardware Components

The prototype was developed and tested on MacBook Pro, Mid 2010 computer [15]. The computer has 2,4 GHz Intel Core 2 Duo processor, 8 GB memory and operating system macOS Sierra version 10.12.3. All the software components described in Subsection 7.2 were downloaded and executed manually in this hardware.

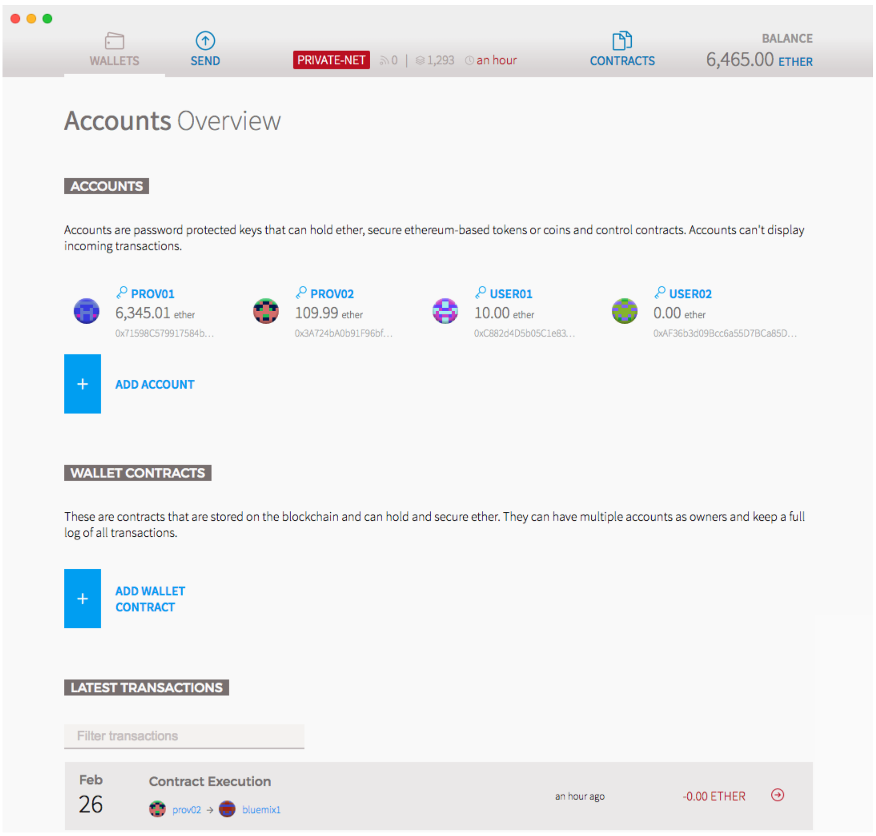
7.2 Software Components

This section describes the software components used for prototype development and testing. The components are Ethereum wallet, geth, docker and Ethereum explorer.

7.2.1 Ethereum wallet

An Ethereum wallet is wallet software developed specifically for Ethereum blockchain using javascript scripting language. It is developed and maintained by Ethereum Foundation [9]. The wallet allows a user to create ethereum blockchain identity, manage their accounts and ether. The user can create multiple identities as well as multisig account with the wallet. Multisig account is an account where one account is associated with more than one private key. It is similar to shared account in the current banking sector. The wallet is connected to blockchain network with blockchain API endpoints. Furthermore, the wallet ensures that each contract is compiled and validated as well as authenticated before deployment.

An Ethereum wallet version 0.8.9 is shown in the Figure 11. It has navigation and view section. The navigation section has wallet, send and contracts tabs. The view section shows the content associated with selected navigation tab. The wallet tab shows accounts overview, wallet contracts and latest transactions items in the view section as shown in the Figure 11. The account overview item shows the list of accounts. The wallet contracts item shows the list of deployed contracts. Latest transactions show latest five transactions as shown in the Figure 11. The send tab is used to send ether



Accounts Overview

ACCOUNTS

Accounts are password protected keys that can hold ether, secure ethereum-based tokens or coins and control contracts. Accounts can't display incoming transactions.

PROV01	PROV02	USER01	USER02
6,345.01 ether	109.99 ether	10.00 ether	0.00 ether
0x71598C579917584b…	0x3A724bA0b91F96bf…	0xC882d4D5b05C1e83…	0xAF36b3d09Bcc6a55D7BCa85D…

ADD ACCOUNT

WALLET CONTRACTS

These are contracts that are stored on the blockchain and can hold and secure ether. They can have multiple accounts as owners and keep a full log of all transactions.

ADD WALLET CONTRACT

LATEST TRANSACTIONS

Filter transactions

Feb 26 Contract Execution prov02 → bluemix1 an hour ago -0.00 ETHER

Figure 11: Ethereum wallet

to another account. Contracts tab is used to create and execute contracts. Apart from these, the navigation section also shows connected blockchain network (private-net), active miner (0), total blocks (1,293) mined, last block time (an hour) and total ether (6,465.00 ETHER) for the user.

7.2.2 geth

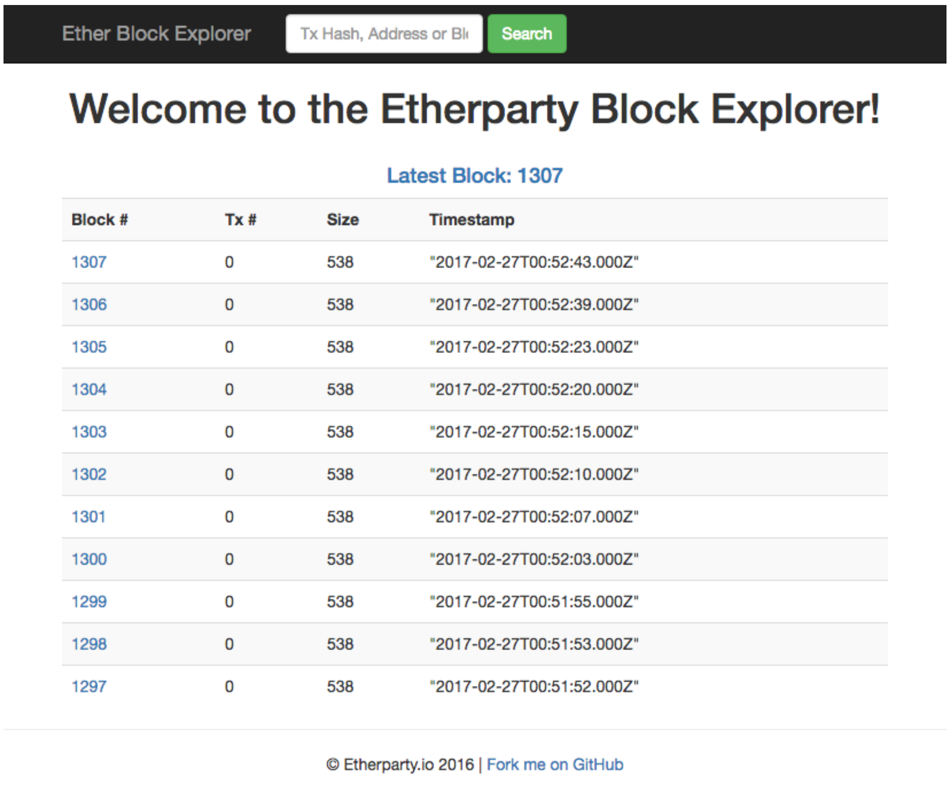
geth is software developed and maintained by Ethereum Foundation [10]. It is a command line interface to run full Ethereum node. This Ethereum node connects either to main-net, test-net or private-net. The geth is used to mine ether, transfer ether between addresses, create contracts, send transactions, debug transactions, explore block history and monitor node.

7.2.3 Docker

Docker is open-source container run-time software designed to create, deploy and run applications [8]. It differs from virtual machines (vms) in that split a piece of hardware to be shared among different users and appear as separate server or machine. Docker virtualizes the operation system, splitting it up into virtual compartments to run container applications. Hence, instead of creating the whole operating system, it allows applications to use the host Linux kernel and only requires the application be shipped with its dependencies and libraries. The docker reduces the size of the applications and makes it easier to ship. Thus, this allows a piece of code to be put into smaller, easily transportable pieces that can run wherever Linux is running.

7.2.4 Ethereum explorer

Ethereum explorer is a lightweight web application. It connects to private blockchain API endpoints and visualizes blocks and transactions [11]. It also allows to search blocks, transactions, address from the blockchain. The explorer web interface shows the total blocks and latest list blocks, number of transactions in the block, size of the block and block creation time. The block numbers when selected give more detailed information about block such as numbers of confirmations, gas used, nonce, size, miner, difficulty, data, gas limit and transactions.



Welcome to the Etherparty Block Explorer!

Latest Block: 1307

Block #	Tx #	Size	Timestamp
1307	0	538	"2017-02-27T00:52:43.000Z"
1306	0	538	"2017-02-27T00:52:39.000Z"
1305	0	538	"2017-02-27T00:52:23.000Z"
1304	0	538	"2017-02-27T00:52:20.000Z"
1303	0	538	"2017-02-27T00:52:15.000Z"
1302	0	538	"2017-02-27T00:52:10.000Z"
1301	0	538	"2017-02-27T00:52:07.000Z"
1300	0	538	"2017-02-27T00:52:03.000Z"
1299	0	538	"2017-02-27T00:51:55.000Z"
1298	0	538	"2017-02-27T00:51:53.000Z"
1297	0	538	"2017-02-27T00:51:52.000Z"

© Etherparty.io 2016 | Fork me on GitHub

Figure 12: Ethereum explorer

An Ethereum explorer is shown in figure 12. The top navigation section has the search field to query transaction, addresses and block as mentioned above. The body section shows the latest block 1307 and list of the blocks with block number (1307-1297). These blocks have 0 transactions, size of each block is 538 bytes and the block creation times varies from 1 second to 8 seconds as shown in the Figure 12.

The Ethereum explorer shown in figure 12 has been modified to meet the prototype need. First it runs in the docker container and the time stamps have been changed from Unix to Coordinated Universal Time (UTC) for readability. Also the footer of the page has been modified to be fixed at the bottom of the web-page.

7.3 Prototype Environment

The software components 7.2 were downloaded manually on the server (computer) 7.1. It was executed with commands listed in the list 1. These commands started the private developer blockchain node, blockchain console, Ethereum wallet and Ethereum explorer respectively.

The private developer blockchain node was started with custom flags: dev, interprocess communication path (ipcpath), remote procedure call (rpc), rpcapi, rpcaddr and rpccorsdomain as shown in the list 1 [10]. The dev flag pre-configures the private network with debug flags. The ipcpath sets the location of ipc pipe which enables ipc communication between clients (blockchain console, explorer, wallet) to share blockchain data. The ipcpath flag is only for local clients communication with the server and is not visible outside this node network. The rpc flag starts the http-rpc server (blockchain server) and it exposes the list of APIs such as database, web3, eth, personal and net. The rpcaddr sets the blockchain server address to localhost on which other clients connect to the blockchain. The rpccorsdomain enables cross origin request of the explorer domain to the blockchain server. Finally, the stdout of the command was redirected to the ethereum.log for persistence logging and further analysis as described in Subsection 8.3.

```
1  # Start private developer blockchain node
2  $ geth --dev --ipcpath $HOME/Library/Ethereum/geth.ipc --datadir
    ↪  $HOME/.ethereum --rpc --rpcapi="db,eth,net,web3,personal" --rpcaddr
    ↪  localhost --rpccorsdomain "*" &> ethereum.log
3
4  # Blockchain node console
5  $ geth --dev attach
6    # Start mining
7    $ miner.start()
8    # Stop mining
9    $miner.stop()
10
11 # Start Ethereum wallet
12 $ /Applications/Ethereum\ Wallet.app/Contents/MacOS/Ethereum\ Wallet --rpc
    ↪  http://localhost:8545
13
14 # Ethereum explorer
15   # Build Ethereum explorer docker image
16   $ docker build -t private-blockchain-explorer:0.1 .
17   # Start explorer container
18   $ docker run -it --rm -p 8000:8000  -v "$(pwd)":/app
    ↪  private-blockchain-explorer:0.1 npm start
```

Listing 1: Prototype environment setup

The blockchain node console was started with command on line 5 as shown in the list 1. The command attached the interactive console to developer private blockchain node. The interactive console enabled starting and stopping of mining with commands 7 and 9 respectively. The Ethereum wallet was started with rpc flag which connected to the private blockchain node. Finally, the docker image of the Ethereum explorer was built with its dependencies (nodejs) and was run on port 8000. The explorer was accessible to the browser on http://localhost:8000.

7.4 Smart Contract Implementation

The core of the prototype is in the smart contracts developed in Solidity language. There are two types smart contracts, Provider and mortal. These smart contracts define the set of rules/instructions about how a cloud user is able to interact with cloud provider and how the cloud provider authenticates and authorizes the users.

7.4.1 Solidity

Solidity is high level, object oriented programming language developed and maintained by Ethereum Foundation. It has been designed for writing smart contracts [83]. Smart contracts contain self-enforcing business logic which is compiled to byte-code and executed on the Ethereum virtual machine.

Currently, solidity is the primary language in Ethereum to develop smart contracts. It provides an easy interface to build Ethereum applications by hiding the complexity of blockchain.

7.4.2 Mortal contract

Contracts in ethereum are by default immortal, which means that once created is created to blockchain, contract owner has no special privileges. Thus, usually contracts inherit the properties of mortal contract which allows the contract owner to kill the contract when no longer needed [9].

In this prototype, the mortal contract has onlyOwner modifier and kill methods. The onlyOwner modifier allows only the contract owner to execute the specified methods. The kill function sends a suicide signal to the contract address which makes the contract useless when the contract is no longer needed. The kill function has onlyOwner modifier which means that only the contract owner can execute it.

7.4.3 Provider contract

Provider contract is the main contract of the prototype. The contract has four functions: registerToProvider, verifyUser, setDebt, payToProvider and unsubscribe. Also, the contract inherits kill function from the mortal contract.

The registerToProvider function registers a user to the cloud provider who has created the contract. The user must have valid blockchain address which is passed as parameter to the registerToProvider function. The verifyUser function ensures the authenticity of the user. It requires message hash, signature of the message and the user address as the parameters from the user. The message hash is obtained by hashing a random message/text with ethereum method web3.sha3 (input). This hash is then signed with the user private key which is protected by the user's paraphrase. The setDebt sets the debt to the user who has already been registered for the cloud provider. This function also requires user address and 256 bit unsigned integer as debt value. The payToProvider allows the registered user to pay the debt, set by the provider. The unsubscribe function de-registers the registered user who has paid all the debt to the cloud provider. Finally, the kill function allows the cloud provider to kill the contract when it is no longer needed.

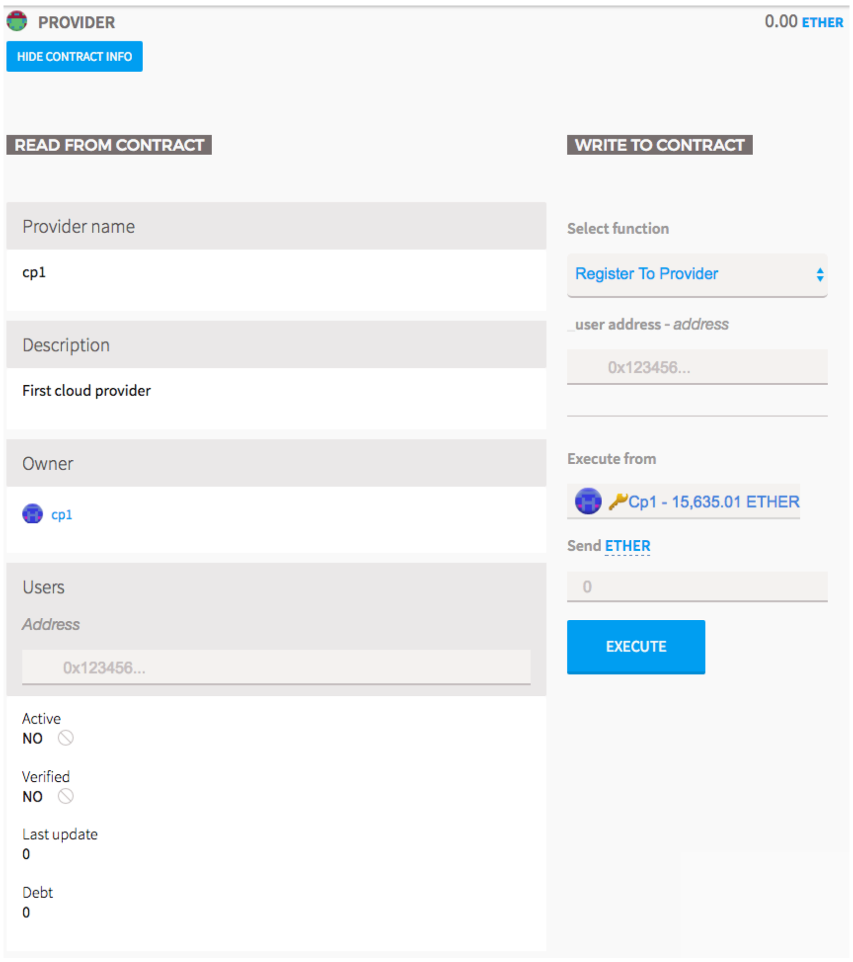
```
// address: variable type that store ethereum address
// mapping: map User object to variable users
mapping(address=>User) public users;

// User object
struct User{
    // user active or not, if subscribed or unsubscribed
    bool active;
    bool verified;
    uint lastUpdate;
    // ether currency needs to be 256
    uint256 debt;
}

// a user is registered with default vaules
function registerToProvider(address _userAddress) onlyOwner{
    // Creates a User struct and saves in storage
    users[_userAddress] = User({
            active: true,
            verified: false,
            lastUpdate: now,
            debt: 0
        });
}
```

Listing 2: Provider contract

A snippet of the Provider contract is shown in the listing 2. A public users address is defined. This address is mapped to the User object as shown in the listing 2. The User object has four fields: active, verified, lastUpdate and debt. The active field represents if user is active (subscribe) or not (unsubscribe). The verified field shows if the user has verified his/her authenticity. The lastUpdate field shows when the user was last time active. The debt field shows the amount of money the user is supposed to pay to the cloud provider. The registerToProvider function takes *_userAddress* as input parameter and has onlyOwner modifier. In addition, this function saves the user object with default values of true, now and 0 for fields active, lastUpdate and debt respectively.

7.5 Executing the Prototype

The prototype was executed on one server (computer as described in Subsection 7.1) with a single instance of private developer blockchain, miner, ethereum explorer and wallet. A cloud provider account, cp1 and a user account user1 were created. The mining was started and the cloud provider account got the ether. The cp1 deployed mortal and provider contracts (PROVIDER) from contracts tab in the wallet. The cp1 was prompted for password before creating the contracts. Once the contracts were created, the cloud provider navigated to PROVIDER contract as shown in the Figure 13. The contract has two columns: read from contract and write to contract. Read from contract reads the contract properties from the blockchain as shown in the Figure 13. It has four sections: Provider name, Description, Owner and Users. Provider name displayed the name of the provider, description displayed the description of the provider set before creating the contract. The owner field shows the cloud provider who deployed the contract. The users section has the input field address which takes a user address as input parameter as shown in the Figure 13. If the user address is supplied to the user input field, the wallet shows if the user was Active, Verified, Last Update and Debt state.


PROVIDER
0.00 ETHER
HIDE CONTRACT INFO

READ FROM CONTRACT

Provider name
cp1

Description
First cloud provider

Owner
cp1

Users
Address
0x123456...

Active
NO

Verified
NO

Last update
0

Debt
0

WRITE TO CONTRACT

Select function
Register To Provider

user address - address
0x123456...

Execute from
Cp1 - 15,635.01 ETHER

Send ETHER
0

EXECUTE

Figure 13: Contract Execution

The cp1 manually executed the Register To Provider function with user address as shown in the Figure 13. As soon as this transaction was mined the user section Active field changed from NO to YES, verified field was set to NO, Last Update timestamps was set to now and Debt to 0. Then, the cp1 executed the second function Verfiy User, which basically checks the authenticity of the user by recovering the user public key from the message signature and comparing it with the user public key provided by the user. If the two public keys match the user is authenticated and verified field is updated to YES. Now the cloud provider can execute Set Debt to the user which basically set the Debt to the user and Debt amount was displayed on the user section once mined. The cp1 executed the Pay To Provider function with the provider address and debt amount which basically pay the debt from user account to the provider account. Finally, if the user had no debt the unsubscribe function de-registered the user from the cloud provider.

8 Prototype Testing

This section describes the method used for testing the prototype. It describes the possible test cases to analyze how well the expectation of the problem statement described in Subsection 1.1 can be met.The section also describes the setup and execution of prototype testing. Moreover, measurements and results are presented along with analysis of results and security aspects.

8.1 Setup and Execution

The test was set up and executed from the cloud provider perspective. The setup and execution was similar to the Prototype Environment (described in Subsection 7.3) and Executing the Prototype (described in Subsection 7.5) respectively with some differences. The differences are following: two cloud provider accounts cp1, cp2 and two users account u1, u2 were created. PayToProvider, unsubscribe functions which are only accessible to registered and verified users were considered as cloud provider resources for simplicity. This setup was executed three times with listed test cases 3 to validate how well we meet the expectations of the problem statement described in Subsection 1.1. The performance of the system was measured. The test setup was manual since it required various manual triggers. However, it was easy to visualize and follow the authentication, authorization and accounting scenarios with blockchain.

In order to test, according to test cases 3, same copy of the provider and mortal contracts were deployed by each cloud provider cp1 and cp2 respectively. First, cp1 executed Pay To Provider functions to u1 as well as u2 and the registration failed, since these users were not registered with the provider and had no access to protected functions. Also, when a wrong user address(fake user) was provided for registration, error was displayed. After u1 was registered to cp1, she had access to Pay To Provider and unsubscribe

- Only authenticated and authorized user can access cloud resources.
- Cloud provider is able to authenticate and authorize users against blockchain.
- Cloud users are able to pay invoice without sharing any bank or credit card details.
- Cloud provider is authenticated to execute the transactions on the blockchain.
- A cloud user can use same identity across multiple cloud providers.
- A cloud user or provider both are anonymous on the network.

Listing 3: Test cases

functions. The cp1 was able to set debt to the u1 and the u1 was able to pay the debt in ether.

The cp2 registered u1. The u1 could access the protected functions of cp2. The cp2 was able to Set Debt to u1 and u1 paid the debt to cp2. Eventually, the u1 unsubscribed (de-registered) from both cloud providers cp1 and cp2. If the debt was paid the unsubscirbe was successful. Otherwise, the u1 could not unsubscribe. Hence, these use cases proves only authenticated and authorized user can access cloud resource and cloud provider was able to authenticate and authorize users with blockchain. The cloud user was able to register to multiple cloud provider(u1 and u2) with same address and pay invoice without sharing any bank or credit card details. All the transactions were visible in the ethereum explorer. These transactions had hashed user id. Thus, it was difficult to associate which user triggered the specific transaction. Therefore, this ensured cloud providers as well as users were anonymous on the network.

The above setup and execution was replicated in three sets. Each set with two cloud provider and two users and CPU, memory, blockchain time and transaction costs were measured. These measurements and results are described in details in the next section.

8.2 Measurement and Results

As mentioned in previous section 8.1, the test setup was executed in three sets and the measurements were done in three sets. During each set, the minimum and maximum hardware resources usage, blockchain time and transaction costs were recorded. The mean value of minimum values were calculated as shown in the tables below 1 2 3 4 represented by avg:min. Similarly, the mean value of maximum values were calculated as shown in the tables below 1 2 3 4 represented by avg:max. Finally, the mean values of the average minimum and maximum were calculated and represented by overall average as shown in the tables below 1 2 3 4.

8.2.1 CPU and Memory

The hardware resources are CPU and memory used by private ethereum blockchain, ethereum wallet and ethereum explorer process. The CPU and memory usage for blockchain are amount of CPU and memory used by the blockchain miner to create, save transactions and reach consensus among nodes as well as send confirmations to other nodes. The CPU and memory usage by ethereum wallet is CPU and memory used by the wallet process to query, send, receive and display transactions on its interface. The CPU and memory usage by the explorer process is the CPU and memory used to query blockchain and display the transactions and block informations on the web interface.

Table 1: CPU usage

	avg:min(%)	avg:max(%)	overall average(%)
Blockchain	59.10	109.10	84.10
Wallet	2.20	14.53	8.36
Explorer	0.36	1.60	0.98

The CPU usage of blockchain, ethereum wallet and explorer is shown in the table 1. The overall average CPU usage of blockchain was 84.1% where 59.1% and 109.1%(the value is more than 100% because 100% refers to full usage of single core and the device has multiple cores) were average minimum and maximum of three sets. The overall average wallet CPU usage was 8.36% and ranged from average 2.2% to 14.53% as shown in the table 1. The overall average explorer CPU usage was 0.98% and ranged from average 0.36% to 1.6%. Thus, blockchain is a CPU hungry process with the highest overall average CPU usage while explorer has the lowest CPU usage.

Table 2: Memory usage

	avg:min(%)	avg:max(%)	overall average(%)
Blockchain	4.80	6.16	5.48
Ethereum Wallet	0.93	9.43	5.18
Explorer	0.06	0.36	0.21

The memory usage of blockchain, ethereum wallet and explorer is shown in the table 2. The overall average memory usage of blockchain was 5.48%

where 4.8% and 6.16% was average minimum and maximum respectively. The average wallet memory usage was 5.18% and ranged from average 0.93% to 9.43%. The average explorer memory usage was 0.21% and ranged from average minimum 0.06% to maximum 0.36% as shown in the table 2. Thus, though the blockchain required more memory capacity compared to wallet and explorer process, it is not a memory intensive process.

8.2.2 Blockchain Time

The blockchain time consist of block creation, consensus and first confirmation time. The block creation time is the time taken by node to aggregate the transactions at the specific time and to create blocks. The consensus time is the time taken by nodes to propagate the block created, to validate it, to reach consensus and add to add to blockchain. The first confirmation is the time taken by the network to propagate the consensus to other nodes and to update their respective blockchain.

Table 3: Blockchain Time

	avg:min (seconds)	avg:max (seconds)	overall average (seconds)
Block Creation	2.67	15.00	8.83
Consensus	4.16	18.66	11.41
First Confirmation	6.28	22.01	14.09

The overall average time to mine and create a block was 8.83 seconds where 2.67 and 15.00 was average minimum and maximum seconds as shown in the table 3. The overall average time to reach consensus for the block was 11.41 seconds where 4.16 and 18.66 were average minimum and maximum time respectively. The first confirmation of the transaction took, on overall average 14.09 second where the minimum was 6.28 seconds and maximum was 22.01 seconds as shown in the table 3. Thus, there was no big difference between block creation time and confirmation time because there was one miner node only. And this small difference was due to other process on the computer using the network resources.

8.2.3 Transaction Cost

The transaction cost consist of transaction creation and execution cost. The transaction creation cost is amount of ether required to execute a smart contract. The transaction execution cost is amount of ether required to execute smart contract function. The cost of the transaction creation depends on two factors, the length of contract and the type of transaction creation method. Longer the contract, more expensive is the cost and shorter the contract, cheaper is the cost. Faster the transaction creation method, the more expensive is the cost and slower the transaction creation method, the cheaper is the cost. The cost of transaction execution dependent on type of execution method, amount and number of input parameters. Faster the execution method, more expensive is the transaction and slower the execution method, the cheaper is the transaction. Similarly, more and longer the input parameter, the most expensive is the transaction and less and shorter the input parameter, cheaper is the transaction. In this prototype, we selected the simple or moderately complex transaction creation and execution method was selected.

Table 4: Transaction Cost

	avg:min (ether)	avg:max (ether)	overall average (ether)
Transaction Creation	0.00118302666	0.00226240666	0.001722716
Transaction Execution	0.00087342	0.0029366	0.00190501

The costs of the transaction creation and execution are shown in table 4. The overall average transaction creation cost was 0.001722716 ether where 0.00118302666 and 0.00226240666 was average minimum and maximum respectively. The overall average transaction execution cost was 0.00190501 ether where 0.00087342 were minimum and 0.0029366 maximum as shown in table 4. The mortal contract was cheapest(0.00118302666 ether) and provider contract(0.00226240666) was most expensive. Similarly, Set Debt function was the cheapest(0.00087342) while Pay to Provider function was the most expensive(0.0029366).

8.3 Analysis of Results

The result of the test showed that one user was able to use two cloud providers with the same user account (address). A user did not need to share any private data except public key to the blockchain network to use cloud services. The cloud providers were able to authenticate and authorize the users against the blockchain network. Moreover, a fake user which did not exist on the blockchain network or with the wrong address was unable to register to the cloud provider. Also, only registered user had access to the cloud resources and a user could pay to the cloud provider in cryptocurrency without sharing any bank or credit card details. All the transactions were accessible to any blockchain user and she was able to verify it. Users were anonymous as only public key was shared with the network. Finally, a user could easily de-register from the cloud provider. Hence, this prototype achieves the expectation of the problem statement stated in section 1.1.

In addition to the above findings, it is obvious the blockchain mining is a CPU hungry process and requires more CPU for faster performance. The transaction length should be kept short to keep the transaction creation and execution cost lower. The transaction should be executed with moderate transaction execution type to keep the cost lower. However, the most expensive transaction costs are very small compared to the existing transaction costs which are required by current bank or financial system. Thus, blockchain solution is cheaper, reliable solution for cloud provider compared to existing authentication, authorization and audition solutions.

It is important to emphasize that these results would be different from the private-net, test-net and main-net. The setup was strictly made for the prototype only. However, the prototype ideas of using one account and registering to multiple cloud providers, cloud provider authenticating, authorizing with blockchain network and the user being able to pay their invoice without sharing their bank or credit card detail would still work. This is because, the blockchain works essentially the same way as it was setup for the prototype with difference in the scale and costs.

During tests it was noticed that the transaction costs and block creation and execution time would vary every time the mining was restarted. This is because, blockchain requires some hours before it stabilizes block creation and also the developer private ethereum blockchain is designed rather for faster testing than accuracy.

8.4 Security Analysis

The prototype proposed in this thesis is a proof-of-concept. Though it inherits the default properties of blockchain such as decentralized consensus, immutable transaction, data security, proof-of-work; it is not distributed and decentralized since it is deployed on one node. However, the setup can be easily deployed on the private-net with multiple nodes or on test-net or main-net. This will make the prototype decentralized and distributed. The prototype inherits the generic blockchain vulnerabilities described in Section 3.6 and has also its own set of weaknesses such as malicious smart contract and public key misuse.

Using the prototype, any node on the network can write their own set of smart contracts, deploy to the network and become cloud provider. Although this approach is secured, transparent compared to client-server architecture 2, it has no protection against malicious smart contracts. The prototype system could cheat user for fake cloud services. There is no specification or organization to set the rules to become cloud providers. Thus, faulty nodes might deploy faulty contracts and try to cheat user with their fake services since there is no mechanism to identify which cloud provider provides non-fake cloud services. This issue can be solved by first formalizing a common set of requirements to become cloud provider and a common set of specifications on how to write and deploy smart contract to become cloud provider. Moreover, these specifications would be deployed to escrow account and each cloud provider must be registered to the escrow account which would ensure the requirements of cloud provider are met. If the requirements are met, cloud provider can offer services else the cloud provider is not allowed to offer any services. In case cloud providers violate the specification, the escrow account would charge penalty for it.

The current prototype assumes all the users are honest users. However, in the real world, there are many malicious users who could basically take the public address of the other user and send a request to the cloud provider to register them as a user. Thus, a malicious user could misuse other user's public key. This issue can be solved by adding a layer of authenticity, which would ask for user credentials when they try to register and use the cloud provider services or cloud provider could maintain a user session with separate user database mapping with blockchain users.

In addition to the above issues, the current prototype system has no backup and account syncing across multiple devices. So, if the hardware gets corrupted or crashes the contracts and accounts cannot be recovered. Thus, for all the cloud provider and users, it is highly recommendable to use main-net and cloud provider as well as users must backup and sync their accounts across multiple devices.

9 Discussion

The solution presented in this thesis can be deployed by individual developer to small or enterprise business for providing an AAA solution. The solution requires minimal hardware changes and cost as it works on the existing hardware and though the process is CPU hungry as described in Subsection 8.2.1, it does not require several servers compared to existing AAA solutions. The solution by design is distributed, decentralized and fault-tolerant which decreases the deployment and maintenance costs.

Despite several advantages of this solution over existing AAA solution as described in Subsection 2, there are various considerations that should be taken into account before deploying the new solution to the production environment by the cloud provider. First, this solution requires fundamental architectural changes to the existing AAA solution which is a major change. The cloud provider must add support for blockchain authentication, authorization and accounting to their existing cloud services. Also, cloud users must be blockchain users. In other words, cloud providers and users must have blockchain wallet, account and registered to the blockchain network. Beside these architectural changes, it is recommended to keep the contract short. The contracts should be deployed and re-deployed only when needed as described in Subsection 8.3 and if the contract is re-deployed the services (applications) must be migrated to use new contracts.

The blockchain core is under heavy development, so the cloud provider must update accordingly with blockchain changes to avoid any downtime in production systems. For example, during prototype development, the contract which worked with ethereum blockchain version 1.5.0 broke with version 1.5.5. If the blockchain core changes such as changing from proof-of-work to proof-of-stake, it will change how the miners reach consensus in the network. Moreover, an outdated blockchain core might be vulnerable and applications might end of getting hacked. This hacking might cause the providers and users to lose their digital assets as described in Subsection 3.6. Thus, cloud provider must have mechanisms to keep the blockchain core updated and at the same time must update contracts in order to avoid any downtime in the production system.

The solution does not include how to revoke the registered users or how to arrange user management with access control. The user management and access control are crucial for the cloud services to differentiate between the user level such as admin, site admin, developer and members. So, cloud provider still needs to maintain separate databases for user management with mapping to the registered blockchain users. The logic of user management, access control and revoking users should be outside smart contract since the contracts should be short as described in Subsection 8.3.

The prototype presented in this thesis is not scalable, as from test, the current average consensus time is 11.41 seconds and even, if this prototype is deployed in main-net, it would theoretically take on average 12 second [32]. This makes it unsuitable for cloud providers such as Google, who processes on average 40,000 search queries per second [12].

Using the proposed solution, any user in the blockchain network can become a cloud provider by deploying smart contracts. So, there should be a mechanism and process to ensure the services offered by the provider are legitimate and cannot close their account when they wish. For these cases, the provider should be registered to an escrow account. This will ensure the services provided by the provider are legitimate. Otherwise a penalty is charged to them. A common structure to write smart contracts should be agreed and also followed to ensure that the provider complies and is able to provide basic AAA to the users.

10 Conclusion and Future work

This section summaries the overall thesis research and implementation. It also recommends possible future work in this area.

10.1 Conclusion

Implementation of the AAA concept with Ethereum blockchain or blockchain in general is in very early phase. This thesis has proposed an AAA solution based on Ethereum blockchain and made contribution in the area of AAA for the cloud environment. The problem has been solved and goal has been achieved.

The architecture of proposed solution consists of three main components: Ethereum wallet, smart contracts and Ethereum blockchain. Ethereum wallet is responsible for authentication of the user against Ethereum blockchain by creating private-public key. The public key is distributed across the blockchain network and private key is kept secret with the user and it is pass-phrase protected. This key-pair acts as user identity where the network is able to verify and validate user authenticity by user's public key. The smart contracts have the core logic of user authorization and the assisting logic of authentication as well as accounting and de-registration of user. The contracts in-charge are the cloud providers who develop and deploy it to the blockchain.

Finally, the Ethereum blockchain is the core of the system which acts as the AAA backend. It ensures all the transaction between users, providers and resources are validated and verified and creates the transactions block which is added to the blockchain after the network reaches the consensus. As soon as block is added to blockchain, this change becomes permanent to the network and is broadcasted and propagated to other nodes. This change cannot be undone or duplicated. Hence, it insures the users are legitimate and authorized. The contract transactions are immortal, anonymous, distributed and decentralized. These transactions can be verified by anyone in the network but cannot be decrypted by anyone than the owner.

A prototype of AAA has been implemented in order to demonstrate the feasibility of the proposed solution. It demonstrates minimal necessary feature set to accomplish the solution. Basic smart contracts are implemented. These contracts authenticate and authorize the user with the blockchain. The contracts also allows to pay invoices. Only authenticated and authorized users are able to access the cloud resources and all the transactions are broadcasted to network. The transactions were immutable once added to blockchain. The prototype proved a user with one identity can use multiple cloud services and pay invoices without sharing any private user data. The prototype also demonstrated that the solution is CPU hungry process and has low memory usage. The solution can be easily deployed and used from small business to large enterprise with existing hardware at minimal cost. The biggest cost is writing and maintaining the contracts logic by developers. Also, the solution has no single point of failure and ensures user data privacy, anonymity at same time as transactions transparency.

Through this research a solution is provided for building cloud-based decentralized, distributed AAA system with blockchain. This solution aims to provide more secure and easier solution compared to the existing AAA framework with existing hardware resource at minimal costs with privacy and user data ownership in the mind. Also, this solution cannot be easily hacked and it is easy to use. Further work/research is needed to improve the technology and test it in the cloud environment.

10.2 Future work

As a prototype, AAA with Ethereum Blockchain is merely rock solid. Since the objective of this thesis was just to develop and test basic proof-of-concept, there are a lot of features and improvements to be done. The remaining components from the architecture diagram 6.2: API could be developed and a real cloud resource could be added to complete the solution. In addition, proper testing environment could be also used for precise usage, blocktime and cost analysis as well to automate the development and execution process.

The web interface for the end users could be develop. The end users could use this web interface to register and de-register to the cloud providers. To support this web interface, an API would be developed which would interact with blockchain API and contract methods without any need of Ethereum wallet. In this case, Ethereum wallet would be only be required for authentication to blockchain. This would decrease the manual effort of executing contracts and provide user friendly interface to the end users.

Finally, the system could be tested with real cloud infrastructure such as openstack [17] or devstack (openstack package for local setup) for real cloud resources and executed on the blockchain main-net to find actual run-time. The real costs of using cloud services such as AWS could be compared and analyzed against this prototype.

References

[1] *Amazon web services (aws) - cloud computing services.* https://aws.amazon.com/, visited on 18-02-2017.

[2] *Application programming interface - wikipedia.* https://en.wikipedia.org/wiki/Application_programming_interface, visited on 18-02-2017.

[3] *Blockchain.* https://blockchain.info/, visited on 13-03-2017.

[4] *Blockchain at berkeley.* https://blockchain.berkeley.edu/, visited on 22-01-2017.

[5] *Blockchain community group.* https://www.w3.org/community/blockchain/, visited on 22-01-2017.

[6] *Compute engine - iaas | google cloud platform.* https://cloud.google.com/compute/, visited on 18-02-2017.

[7] *Crypto-currencies statistics.* https://bitinfocharts.com/, visited on 03-02-2017.

[8] *Docker - build, ship, and run any app, anywhere.* https://www.docker.com/, visited on 19-02-2017.

[9] *Ethereum wallet and mist.* https://github.com/ethereum/mist/releases, visited on 10-02-2017.

[10] *Geth · ethereum/go-ethereum wiki · github.* https://github.com/ethereum/go-ethereum/wiki/geth, visited on 27-02-2017.

[11] *Github - etherparty/explorer: A lightweight ethereum block explorer.* https://github.com/etherparty/explorer, visited on 19-02-2017.

[12] *Google search statistics - internet live stats.* http://www.internetlivestats.com/google-search-statistics/, visited on 04-03-2017.

[13] *Ibm bluemix - cloud infrastructure, platform services, watson, & more paas solutions.* https://www.ibm.com/cloud-computing/bluemix/, visited on 18-02-2017.

[14] *Ledger.* https://www.ledgerwallet.com/, visited on 13-03-2017.

[15] *Macbook pro (13-inch, mid 2010) - technical specifications.* https://support.apple.com/kb/sp583?locale=en_US, visited on 20-05-2017.

[16] *Microsoft azure: Cloud computing platform & services.* https://azure.microsoft.com/en-us/, visited on 18-02-2017.

[17] *Openstack open source cloud computing software.* `https://www.openstack.org/`, visited on 28-03-2017.

[18] *What is ethereum? a step-by-step beginners guide.* `http://blockgeeks.com/guides/what-is-ethereum/`, visited on 09-02-2017.

[19] *Icann*, 1998. `https://www.icann.org/`, visited on 10-02-2017.

[20] *What is openid? | openid*, 2005. `http://openid.net/get-an-openid/what-is-openid/`.

[21] *Json-rpc 2.0 specification*, March 2010. `http://www.jsonrpc.org/specification`, visited on 18-01-2017.

[22] *dotbit.me*, 2012. `https://dotbit.me/`, visited on 10-02-2017.

[23] *Net losses: Estimating the global cost of cybercrime*, June 2014. `https://www.mcafee.com/us/resources/reports/rp-economic-impact-cybercrime2.pdf`, visited on 27-03-2017.

[24] *Man-in-the-middle attack - owasp*, August 2015. `https://www.owasp.org/index.php/Man-in-the-middle_attack`, visited on 22-03-2017.

[25] *Blockchain enigma. paradox. opportunity*, 2016. `https://www2.deloitte.com/content/dam/Deloitte/ch/Documents/innovation/ch-en-innovation-deloitte-what-is-blockchain-2016.pdf`, visited on 21-01-2017.

[26] *These were the biggest hacks, leaks and data breaches of 2016 - page 7 | zdnet*, 2016. `http://www.zdnet.com/pictures/biggest-hacks-security-data-breaches-2016/7/`, visited on 26-03-2017.

[27] Aich, A. and Sen, A.: *Study on cloud security risk and remedy.* International Journal of Grid Distribution Computing, 8(2):155–166, 2015.

[28] Ali, M., Nelson, J., Shea, R., and Freedman, M. J.: *Blockstack: A global naming and storage system secured by blockchains.* In *2016 USENIX Annual Technical Conference (USENIX ATC 16)*, pages 181–194, Denver, CO, 2016. USENIX Association, ISBN 978-1-931971-30-0. `https://www.usenix.org/conference/atc16/technical-sessions/presentation/ali`.

[29] Aljoumah, E., Al-Mousawi, F., Ahmad, I., Al-Shammri, M., and Al-Jady, Z.: *Sla in cloud computing architectures: A comprehensive study.* 8(5):7–32, October 2015. `http://www.earticle.net/article.aspx?sn=257181`.

[30] Balaram, V. V. S. S. S.: *Cloud computing authentication techniques: A survey.* International Journal of Scientific Engineering and Technology Research (IJSETR), 06(03):0458–0464, January 2017, ISSN 2319-8885.

[31] Bonneau, J., Miller, A., Clark, J., Narayanan, A., Kroll, J. A., and Felten, E. W.: *Sok: Research perspectives and challenges for bitcoin and cryptocurrencies.* In *2015 IEEE Symposium on Security and Privacy*, pages 104–121, May 2015.

[32] Buterin, V.: *Ethereum: A next-generation smart contract and decentralized application platform.* Technical report, Florida, USA, 2013. `https://github.com/ethereum/wiki/wiki/White-Paper`.

[33] Castro, M. and Liskov, B.: *Practical byzantine fault tolerance and proactive recovery.* ACM Trans. Comput. Syst., 20(4):398–461, November 2002, ISSN 0734-2071.

[34] Chadwick, W. D. and Fatema, K.: *A privacy preserving authorisation system for the cloud.* Journal of Computer and System Sciences, 78(5):1359 – 1373, 2012, ISSN 0022-0000. `http://www.sciencedirect.com/science/article/pii/S0022000011001644`, {JCSS} Special Issue: Cloud Computing 2011.

[35] Chou, T.: *Security threats on cloud computing vulnerabilities.* International Journal of Computer Science & Information Technology, 5(3):79, June 2013.

[36] Crosby, M., Nachiappan, Pattanayak, P., Verma, S., and Kalyanaraman, V.: *Blockchain technology beyond bitcoin.* Technical report, Berkeley, CA, USA, oct 2015.

[37] Ducklin, P.: *Github hit by massive password guessing attack – naked security*, June 2016. `https://nakedsecurity.sophos.com/2016/06/16/github-hit-by-massive-password-guessing-attack/`, 27-03-2017.

[38] Eastlake, 3rd, D. and Hansen, T.: *Us secure hash algorithm 1.* Rfc 4634, RFC Editor, July 2006. `https://tools.ietf.org/html/rfc4634`.

[39] Froystad, P. and Holm, J.: *Blockchain: Powering the internet of value.* Technical report, EVRY Labs, August 2015. `https://www.evry.com/globalassets/insight/bank2020/bank-2020---blockchain-powering-the-internet-of-value---whitepaper.pdf`.

[40] Gallagher, P.D.: *Fips pub 186-4 digital signature standard.* Technical report, Gaithersburg, MD., USA, July 2013.

[41] Greenwald, G. and MacAskill, E.: *Nsa prism program taps in to user data of apple, google and others*, June 2013. `https://www.theguardian.com/world/2013/jun/06/us-tech-giants-nsa-data`, visited on 18-03-2017.

[42] Hakobyan, D: *Authentication and authorization systems in cloud environments.* Master's thesis, Stockholm, Sweden, 2012.

[43] Hardt, D. and Ed.: *The oauth 2.0 authorization framework.* Rfc 6749, RFC Editor, October 2012. `https://tools.ietf.org/html/rfc6749`.

[44] Hong, N.: *Silk road creator found guilty of cybercrimes*, Feb 2015. `http://www.marketwatch.com/story/silk-road-creator-found-guilty-of-cybercrimes-2015-02-04-151035739`, visited on 22-01-2017.

[45] Hutt, R.: *World economic forum*, 2016. `https://www.weforum.org/agenda/2016/06/blockchain-explained-simply/`, visited on 09-01-2017.

[46] Kalodner, H., Carlsten, M., Ellenbogen, P., Bonneau, J., and Narayanan, A.: *An empirical study of namecoin and lessons for decentralized namespace design.* 2015.

[47] Kraft, D.: *Nameid: Your crypto-openid*, 2013. `https://nameid.org/`, visited on 24-03-2017.

[48] Krug, J.: *Serpent.* `https://github.com/ethereum/wiki/wiki/Serpent`, visited on 10-02-2017.

[49] Larchevêque, E.: *Bitcoin address authentication protocol (bitid)*, August 2016. `https://github.com/bitid/bitid/blob/master/BIP_draft.md`, visited on 24-03-2017.

[50] Larchevêque, E.: *Bitid open protocol - demonstration site*, August 2016. `http://bitid.bitcoin.blue/`, visited on 24-03-2017.

[51] Leiba, B.: *Oauth web authorization protocol.* IEEE Internet Computing, 16(1):74–77, Jan 2012, ISSN 1089-7801.

[52] LeRoux, Y.: *Privacy concerns in the digital world*, October 2013. `http://www.computerweekly.com/opinion/Privacy-concerns-in-the-digital-world`, visited on 02-04-2017.

[53] Lundkvist, C., Heck, R., Torestensson, J., Mitton, Z., and Sena, M.: *Uport: A platform for self-sovereign identity.* Technical report, October 2016. `http://whitepaper.uport.me/uPort_whitepaper_DRAFT20161020.pdf`.

[54] Marcon, A. L., Santin, A. O., Stihler, M., and Bachtold, J.: *A ($rmucon_{ABC}$) resilient authorization evaluation for cloud computing.* IEEE Transactions on Parallel and Distributed Systems, 25(2):457–467, Feb 2014, ISSN 1045-9219.

[55] Mell, P. M. and Grance, T.: *Sp 800-145. the nist definition of cloud computing.* Technical report, Gaithersburg, MD, United States, 2011.

[56] Mihoob, A., Molina-Jimenez, C., and Shrivastava, S.: *Consumer-centric resource accounting in the cloud.* Journal of Internet Services and Applications, 4(1):8, 2013, ISSN 1869-0238. `http://dx.doi.org/10.1186/1869-0238-4-8`.

[57] Mohamed, M. E., Abdelkader, H. S., and El-Etriby, S.: *Data security model for cloud computing.* Journal of Communication and Computer, 10(08):1047–1062, August 2013, ISSN 1548-7709.

[58] Morin, J. H., Aubert, J., and Gateau, B.: *Towards cloud computing sla risk management: Issues and challenges.* In *2012 45th Hawaii International Conference on System Sciences*, pages 5509–5514, Jan 2012.

[59] Murphy, A.: *Accounting in the cloud.* Accountancy Ireland, 43(3):56–57, June 2011.

[60] Mutton, P.: *Wikileaks.org taken down by us dns provider*, December 2010. `https://news.netcraft.com/archives/2010/12/03/wikileaks-org-taken-down-by-us-dns-provider.html`, visited on 18-03-2017.

[61] Nakamoto, S.: *Bitcoin: A peer-to-peer electronic cash system.* Technical report, Bitcoin org, 2008.

[62] Narayanan, A., Bonneau, J., Felten, E., Miller, A., and Goldfeder, S.: *Bitcoin and Cryptocurrency Technologies.* Princeton University Press, feb 2016.

[63] Nelson, M.: *The byzantine generals problem.* Dr.Dobb's Journal, 33(4):30–36, April 2008.

[64] Nouriddine, M. and Bashroush, R.: *A performance optimization model towards oauth 2.0 adoption in the enterprise.* In *Proceedings of the 7th International Conference on Global Security, Safety & Sustainability (ICGS3)*, 2011. `http://roar.uel.ac.uk/1531/`.

[65] Oluwatosin, H. S.: *Client-server model.* IOSR Journal of Computer Engineering, 16(1):67–71, Feb 2014, ISSN 2278-8727.

[66] Perez, S.: *117 million linkedin emails and passwords from a 2012 hack just got posted online | techcrunch*, May 2016. `https://techcrunch.com/2016/05/18/117-million-linkedin-emails-and-passwords-from-a-2012-hack-just-got-posted-online/`, visited on 21-03-2017.

[67] Project, B.: *Bitcoin*. `https://bitcoin.org/en/`, visited on 09.01.2017.

[68] Rissanen, E.: *eXtensible Access Control Markup Language (XACML) Version 3.0*. Technical report, January 2013. `http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html`.

[69] Rodrigues, E.: *The blockchain architecture in a nutshell*. Technical report, Linkedin, September 2016. `https://www.linkedin.com/pulse/blockchain-architecture-nutshell-eder-rodrigues`.

[70] Salsman, M. R.: *The financial crisis was a failure of government, not free markets*, September 2013. `https://www.forbes.com/sites/richardsalsman/2013/09/19/the-financial-crisis-was-a-failure-of-government-not-free-markets/#2e29917c51c3`, visited on 04-04-2017.

[71] Seals, T.: *Massive brute-force attack on alibaba affects millions - infosecurity magazine*, Feb 2016. `https://www.infosecurity-magazine.com/news/massive-bruteforce-attack-on/`, visited on 27-03-2017.

[72] Seibold, S. and Samman, G.: *Consensus immutable agreement for the internet of value*. Technical report, June 2016. `https://assets.kpmg.com/content/dam/kpmg/pdf/2016/06/kpmg-blockchain-consensus-mechanism.pdf`.

[73] Shin, L.: *Hackers are hijacking phone numbers and breaking into email, bank accounts: How to protect yourself*, December 2016. `https://www.forbes.com/sites/laurashin/2016/12/21/hackers-are-hijacking-phone-numbers-and-breaking-into-email-and-bank-accounts-how-to-protect-yourself/#abb71b5360f7`, visited on 27-03-2017.

[74] Spence, D., Vollbrecht, J., Gommans, L., Gross, G., and Laat, C. de: *Generic aaa architecture*. Rfc 2903, RFC Editor, August 2000. `http://www.ietf.org/rfc/rfc2903.txt`.

[75] Stefan, D. B., Farzad, S., Yasir, M., and Bessam, A.: *A peer-to-peer architecture for remote service discovery*. Procedia Computer Science, 10:976 – 983, 2012, ISSN 1877-0509. `http://www.sciencedirect.com/science/article/pii/S1877050912004905`.

[76] Stevens, M., Bursztein, E., Karpman, P., A., Albertini, Markov, Y., Bianco, P. A., and Baisse, C.: *Google online security blog: Announcing the first sha1 collision*, February 2017. `https://security.googleblog.com/2017/02/announcing-first-sha1-collision.html`, visited on 04-04-2017.

[77] Thielman, S.: *Yahoo hack: 1bn accounts compromised by biggest data breach in history*, December 2016. `https://www.theguardian.com/technology/2016/dec/14/yahoo-hack-security-of-one-billion-accounts-breached`, visited on 18-03-2017.

[78] Tucker, C. and Catalini, C.: *Blockchain research at mit.* `http://blockchain.mit.edu/`, visited on 22-01-2017.

[79] Tuwiner, J.: *Bitcoin mining centralization.* `https://www.bitcoinmining.com/bitcoin-mining-centralization/`, visited on 22-01-2017.

[80] Underwood, S.: *Blockchain beyond bitcoin.* Commun. ACM, 59(11):15–17, oct 2016, ISSN 0001-0782.

[81] Wang, L., Ohta, K., and Kunihiro, N.: *Near-collision attacks on md4: Applied to md4-based protocols.* IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, E92.A(1):76–86, 2009.

[82] Weaver, A. C.: *Biometric authentication.* Computer, 39(2):96–97, Feb 2006, ISSN 0018-9162.

[83] Wood, G.: *Solidity.* `https://github.com/ethereum/wiki/wiki/The-Solidity-Programming-Language`, visited on 10-02-2017.

[84] Woolf, N.: *Ddos attack that disrupted internet was largest of its kind in history, experts say | technology | the guardian*, Oct 2016. `https://www.theguardian.com/technology/2016/oct/26/ddos-attack-dyn-mirai-botnet`, visited on 21-03-2017.

[85] Xu, J. J.: *Are blockchains immune to all malicious attacks?* Financial Innovation, 2(1):25, 2016, ISSN 2199-4730.

[86] Yang, R., Lau, W. C., and Liu, T.: *Signing into one billion mobile app accounts effortlessly with oauth2. 0.* Technical report, 2016. `https://www.blackhat.com/docs/eu-16/materials/eu-16-Yang-Signing-Into-Billion-Mobile-Apps-Effortlessly-With-OAuth20-wp.pdf`.