

NAIST-IS-DD1461014

Doctoral Dissertation

**The Bitcoin Network as Platform for
Role-Based Access Control and Electronic
Voting: Using Blockchain-Based Technology to
Create Innovative Systems**

Jason Paul Miranda Cruz

March 10, 2017

Department of Information Systems
Graduate School of Information Science
Nara Institute of Science and Technology

A Doctoral Dissertation
submitted to Graduate School of Information Science,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
Doctor of ENGINEERING

Jason Paul Miranda Cruz

Thesis Committee:

Professor Kenichi Matsumoto (Supervisor)

Professor Kazutoshi Fujikawa (Co-supervisor)

Visiting Professor Yuichi Kaji (Co-supervisor)

The Bitcoin Network as Platform for Role-Based Access Control and Electronic Voting: Using Blockchain-Based Technology to Create Innovative Systems*

Jason Paul Miranda Cruz

Abstract

Bitcoin is the first decentralized global currency cryptosystem and a complete digital money that has increased in value and popularity since 2009. It is a collection of cryptographic protocols that allows secure online transactions between users and is based on a peer-to-peer network powered by its users. In this study, we investigate the use of Bitcoin and its underlying technologies as platform for innovative systems. In particular, we use Bitcoin as an infrastructure to realize a trans-organizational role-based access control (RBAC) system and an electronic voting (e-voting) system.

The RBAC is a natural and versatile model of the access control principle. In the real world, it is common that an organization provides a service to a user who owns a certain role that was issued by a different organization. However, such a trans-organizational RBAC is not common in a computer network because it is difficult to establish both the security that prohibits malicious impersonation of roles and the flexibility that allows small organizations and individual users to fully control their own roles. Therefore, we propose a trans-organizational RBAC mechanism that makes use of Bitcoin to represent the trust and endorsement relationship that are essential in RBAC and a challenge-response authentication protocol that verifies users' ownership of roles.

*Doctoral Dissertation, Department of Information Systems, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-DD1461014, March 10, 2017.

E-voting is a promising platform that aims to provide a secure, convenient, and efficient voting environment over the Internet. Various cryptographic schemes have been proposed to realize secure and efficient e-voting systems, but these systems are hardly used in practical voting. One of the technical reasons for this unfortunate situation is that many e-voting systems require an anonymous communication channel, which is difficult to implement over the Internet. Therefore, we propose the use of Bitcoin and complement it with known protocols, such as the blind signature protocol and digital signature protocol, to realize an e-voting system that is secure, anonymous, and transparent. We discuss several important properties of e-voting systems, including fairness, eligibility, anonymity, robustness, and verifiability, and show that the use of the Bitcoin protocol provides favorable features besides the anonymity of the communication.

Keywords:

Bitcoin, role-based access control, electronic voting, blind signature, blockchain

Contents

1	Introduction	1
1.1.	Research Motivation	3
1.1.1	Bitcoin	3
1.1.2	Role-Based Access Control	4
1.1.3	Electronic Voting	6
1.2.	Research Contribution	7
1.2.1	Contribution to RBAC	7
1.2.2	Contribution to E-voting	8
1.3.	Dissertation Layout	8
2	The Bitcoin Protocol	10
2.1.	Bitcoin Addresses and Wallets	12
2.1.1	Generating the key pair	14
2.1.2	Bitcoin Addresses	14
2.1.3	Wallets	15
	Nondeterministic Wallets	15
	Deterministic Wallets	15
	Types of Wallets	17
2.2.	Transactions	18
2.2.1	Transaction format	19
2.2.2	Transaction Inputs and Outputs	19
2.2.3	Transaction Fees	20
2.2.4	Transaction Details	21
2.3.	Blocks and the Blockchain	23
2.4.	Block structure	23

2.5.	Mining and Proof-of-Work	25
2.5.1	Transaction Verification	26
2.5.2	Transaction Priority and Fees	27
2.5.3	Proof-of-Work Algorithm	28
2.6.	Bitcoin as Infrastructure	30
2.6.1	RBAC	30
2.6.2	E-Voting	30
2.7.	Attacks on the Bitcoin Network	32
3	Role-Based Access Control (RBAC)	34
3.1.	Models for RBAC	34
3.1.1	Related System based on Hierarchical ID-Based Encryption	36
3.2.	Proposed RBAC System using the Bitcoin Network	37
3.2.1	Procedures	38
	Prerequisites	38
	Creating the role-issuer and user connection	40
	Verifying a user-role assignment	41
	Challenge-Response Protocol	41
	Comparison of Practicality and Costs	41
3.3.	Role Management	44
3.3.1	Personalization of roles	44
3.3.2	Role Re-issuance	46
3.3.3	Additional Security Measures	46
3.3.4	Endorsement	47
3.3.5	Trusted Timestamping as Proof of Validity or Expiration Dates	48
4	E-Voting	50
4.1.	Blind Signature Scheme	51
4.2.	Secret Voting for Large Scale Elections	51
4.3.	Blind Multisignatures	53
4.4.	Public Registration Boards and E-Voting	55
4.5.	Using the Bitcoin Blockchain for secure, independently verifiable, electronic votes	56

4.6.	Proposed E-Voting Protocol	57
4.6.1	Initialization and Preparation	57
4.6.2	Administration	57
4.6.3	Bitcoin Address Registration	58
4.6.4	Voting	59
4.6.5	Opening and Counting	60
4.7.	Security Analysis	60
5	Conclusion and Future Work	63
	References	65
	References	65
	Acknowledgements	74
	Appendix	76
A.	Constant-Sum Fingerprinting for Winternitz One-Time Signature	77
A.1	Introduction	77
A.2	Winternitz OTS	79
A.3	Constant-Sum Fingerprinting	82
	Specification of the Function	82
	Construction of the Function	83
A.4	Winternitz OTS with Constant-Sum Fingerprinting	85
	Description of the Scheme	85
	Signature Size and Operation Costs	85
A.5	Conclusion	87
B.	Trans-Organizational Role-Based Access Control in Android	88
B.1	Introduction	88
B.2	Models for the Role-Based Access Control	90
B.3	Hierarchical ID-Based Encryption	91
B.4	Proposed Scheme	94
	Overview	94
	Procedures	95
	Managing Roles	96
B.5	Realization	98

B.6	Conclusion and Future Work	101
C.	Blockchain 2.0	103
C.1	Problems with the Bitcoin blockchain	103
C.2	Uses of Blockchain Technology	104
C.3	Ethereum	105
	Smart Contracts	105
	How Smart Contracts Work	106
	Publication List	109

List of Figures

1.1	The Bitcoin logo.	2
1.2	The RBAC framework.	5
1.3	Trans-organizational RBAC framework.	5
2.1	A list of some of the top Bitcoin exchanges [24].	12
2.2	Bitcoin ATMs [25].	13
2.3	Conversion from the public key to Bitcoin Address [28].	16
2.4	A web wallet from Blockchain.info.	17
2.5	A brainwallet with pass phrase of “jason paul cruz” from bitaddress.org.	18
2.6	Structure of a Bitcoin transaction [30].	19
2.7	Example of a transaction viewed from a blockchain browser.	22
2.8	Simplified representation of the Bitcoin blockchain.	23
2.9	Structure of a block [36].	24
2.10	Structure of a block header [37].	24
2.11	Example of how the Merkle tree of Bitcoin transactions is created [38].	25
2.12	Results of applying SHA256 to a phrase with different integers appended to it.	29
2.13	Example of a Prepaid Bitcoin card [43].	33
3.1	Overview of the proposed RBAC using Bitcoin.	39
3.2	The signing (top) and verifying (bottom) of a message features of a Bitcoin wallet.	42
3.3	A user leaking the private key to other users.	45
3.4	A public note attached to a transaction.	46

3.5	Example case where a user endorses another user.	48
4.1	Representation of the blind signature protocol using RSA.	52
4.2	An e-voting scheme that uses a public bulletin board system.	56
4.3	Overview of the proposed structure.	58
5.1	Overview of the proposed RBAC using HIBE.	95
5.2	(a) RkeyGenerate, (b) Role Response, (c) Role Challenge, and (d) QR Code functions of the application.	99
5.3	The Ethereum Logo.	106
5.4	Example of a smart contract deployed using the Ethereum wallet.	107
5.5	Functions that pertain to specific pieces of the code in the smart contract.	108
5.6	Some of the Dapps using blockchain technology [98].	108

List of Tables

- 5.1 Parameters, signature sizes and costs of Winternitz OTS 81
- 5.2 Relation among l , w and $N_{l,w} = \log_2 |\mathcal{T}_{l,w}|$ 82
- 5.3 Smallest l that makes $N_{l,w} \geq 160$, and the efficiency of the scheme 83

Chapter 1

Introduction

In the past decades, many different applications and systems have been created and conceptualized to solve diverse problems through the Internet in a cooperative and distributed manner. Some of the well-known community-driven systems include anonymous and untraceable electronic communication [1], Hash-Cash (proof-of-work algorithm) [2], BitTorrent (peer-to-peer file sharing) [3], and Spotify (peer-to-peer music-on-demand streaming) [4], to name a few.

Soon after the inception and deployment of these applications, practical implementations and improvements follow, with the exception of applications in the field of digital currency or digital money. The early attempts on digital money [5, 6] require a central entity or bank. Some systems, such as B-money [7], Bit Gold [8], and RPOW - Reusable Proofs of Work [9], tried to remove the bank as central authority by using cryptographic puzzles to mine the digital money. However, these systems still require a central entity to maintain the validity of ownership and transaction records.

In general, a central entity or bank is needed to detect and prevent double-spending of the digital money. Double-spending is an inherent problem in digital currency systems where a user issues two transactions containing the same digital money in parallel, that is, a user makes a copy of the digital money and transfers the copy and the original to different recipients (or the user can even keep the original). To eliminate the central entity, the record of the ownership of the digital money should also be distributed.

In 2008, Bitcoin* has been conceptualized and its design and protocols provide a secure and decentralized digital payment system. The key innovation of Bitcoin is the use of a distributed computation system, or a proof of work (PoW) algorithm, to conduct a global update every 10 minutes to allow the decentralized network to achieve a *consensus* about the state of the transactions. This PoW algorithm provides a solution to the double-spending problem. The main purpose of Bitcoin is to enable a payment system and complete digital money** that is secure and decentralized; that is, it is a peer-to-peer (P2P) network powered by its users and with no central authority. To achieve security and decentralization, transactions are publicly announced and the participants agree on a single history of these transactions. The transactions are grouped into blocks, given timestamps, and then published. The hash of each block includes the hash of the previous block to form a chain, making accepted blocks difficult to alter.

Bitcoin is a collection of technologies that work perfectly together to create a digital money ecosystem. These technologies provide many features that open possibilities for use aside from sending digital money from one user to another. Some examples of the use of Bitcoin aside from transferring money include, but not limited to, trading ownership, notarization, dispute mediation, user authentication, voting, and crowdfunding.



Figure 1.1. The Bitcoin logo.

*In this study, the term “Bitcoin” pertains to the entirety of the mechanism. Additional terms will be included when referring to individual elements under Bitcoin, such as “Bitcoin address” or “Bitcoin blockchain”.

**The digital money used in Bitcoin will be referred to as “bitcoin/bitcoins” with “BTC” as the unit of currency.

1.1. Research Motivation

Following the success of Bitcoin as a decentralized global currency cryptosystem, we investigate its potential as a complementing infrastructure to realize other secure, practical, and decentralized systems. In particular, we use Bitcoin as platform to realize a trans-organizational role-based access control (RBAC) system and a secure and practical electronic-voting (e-voting) system.

1.1.1 Bitcoin

Since its inception in 2008, Bitcoin has increased in value and popularity as a revolutionary digital cryptocurrency system—the term cryptocurrency, which was first described in 1998 by Wei Dai in the cypherpunks mailing list, suggests the idea of a digital money that uses cryptography to control the creation of digital coins and transactions [10]. The first Bitcoin specification and proof of concept was introduced to the world by Satoshi Nakamoto, who published a paper entitled “Bitcoin: A Peer-to-peer Electronic Cash System” [11] in The Cryptography mailing list. The true identity of Satoshi Nakamoto remains unknown until today, and the name is believed to be a pseudonym for a group of people or organizations.

As of March 2017, Bitcoin has a market capitalization of almost 20 billion USD, market price per bitcoin of approximately 1,200 USD, and on average, 300,000 transactions daily [12]. Over the years, Bitcoin has been embraced by the public and has a good track record in providing a secure digital payment system. The Bitcoin network features many favorable properties, including easy mobile payments, reliability, full control of one’s own money, high availability, fast international payments, zero or low fees, protected identity, and privacy [13]. These properties make Bitcoin attractive for use to create technologies that can be applied to provide solutions in different fields. Some examples of services and solutions where Bitcoin technology can be used are as follows:

- **Proof of Existence/Notarization** - Every electronic file has a corresponding hash, which is a string of characters that can be used as a digital fingerprint to uniquely identify a file. If a user wants to prove the existence of a particular document at a particular time, then the hash of the document can be converted to a Bitcoin address and a transaction to that

address can be created so that it can be permanently recorded in the Bitcoin blockchain with a timestamp.

- **Dispute Mediation** - Bitcoin's multi-signature addresses can be used by a third party to approve or reject transactions between the other parties without having control of the money of the other parties.
- **Authentication and Trading Ownership** - Different items, such as tickets, products, and subscriptions, can be linked to Bitcoin addresses and recorded in the blockchain where only the rightful owner will be able to prove ownership of such items.

The application of Bitcoin is virtually unlimited, and the most interesting uses of Bitcoin may not have even been discovered yet.

1.1.2 Role-Based Access Control

Roles and titles are often used to distinguish the eligibility of people to access certain services. Such mechanism is modeled as the role-based access control (RBAC) [14] framework, which describes the access control relation among users and services. In RBAC, users are associated with roles, and roles are associated with services. This framework is compatible with the access control requirements of real-world organizations and is employed in the computer systems of many organizations and companies. For example, in a university, professors are given access to certain files in computer servers while students are denied of such service, as shown in Figure 1.2.

However, it must be noted that RBAC is a versatile framework, and roles are often used in a trans-organizational manner. For example, students are often allowed to purchase computer software at an academic-discounted price, as shown in Figure 1.3. In this example, the "student" role that is issued by an organization (University) is used by another organization (Computer Shop) to determine if a guest is eligible to receive a certain service (Discounted Price). This kind of trans-organizational use of roles is common in face-to-face communication, but it is not obvious in computer networks. Even if a person has a certain role (student role) that is issued by an organization (University), he/she has no systematic way

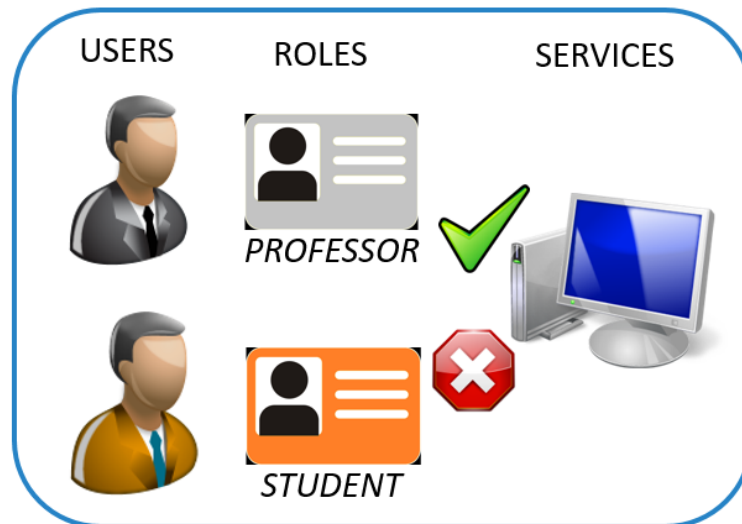


Figure 1.2. The RBAC framework.

of convincing a third-party organization (Computer Shop) that he/she really has that role.

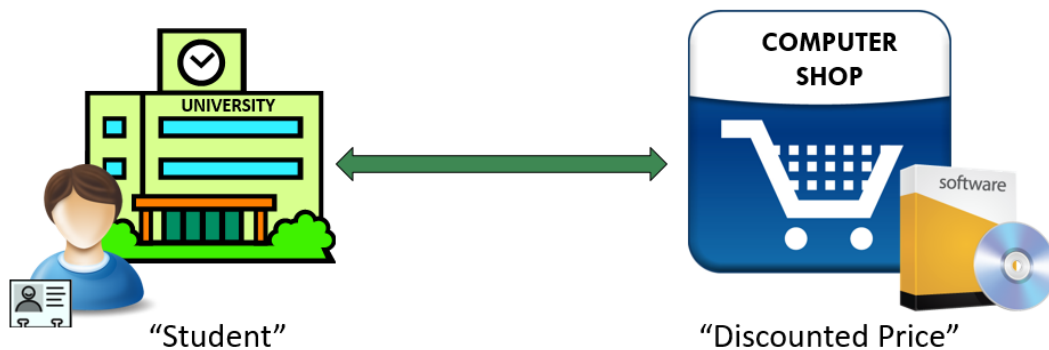


Figure 1.3. Trans-organizational RBAC framework.

To realize a trans-organizational RBAC mechanism in a computer network, a mechanism that prevents malicious users from disguising their roles is necessary. This requirement is naturally accomplished in real-world services with the use of physical certificates, such as passports and ID-cards, which are expected to be difficult to forge or alter. This problem, however, is not obvious in a computer system. Digital certificates [15] can be utilized as an analogue of physical

certificates, but the use of digital certificates is not favorable because it requires considerable and continuous elaborations to maintain secure public-key infrastructures. Another less sophisticated approach to the security problem is to let a service-providing organization (the Computer Shop in the above example) inquire a role-issuing organization (the University in the above example) about the user-role assignment. This approach works in some cases [16], but a focal point of this approach is the necessity for the agreed beneficial relationship among organizations. Consequently, a new organization will experience difficulties (or even not be able) to join the partnership, severely restricting the trans-organizational utilization of roles.

1.1.3 Electronic Voting

Electronic voting (e-voting) is a promising platform that aims to provide a secure, convenient, and efficient voting environment over the Internet. However, voting through the Internet introduces several concerns, such as fraud, anonymity, and abuse, among others. Previous research have introduced different e-voting systems and protocols with different encryption schemes of varying complexity [5, 17, 18, 19, 20]. These systems make use of a combination of different protocols, such as blind signatures, threshold blind signatures, discrete logarithmic encryption, untappable channels, and anonymous channels or public bulletin boards, to satisfy the most properties that make e-voting systems secure. Even though these systems are comprehensive theoretically, a complete solution that can be implemented in the practical domain is yet to be found. One of the most critical problems in the practical implementation of e-voting systems is the realization of an anonymous communication channel, which is assumed in many theoretical schemes and is considered to be one of the most important feature that can satisfy a number of the properties of e-voting systems. An e-voting system can be considered secure if it satisfies the following properties:

- **Completeness:** An eligible voter is always accepted by the administrator and all valid votes are counted correctly.
- **Robustness/Soundness:** Dishonest voters and other participants cannot disturb/disrupt an election.

- **Anonymity/Privacy:** All votes must be secret; and neither voting authorities nor anyone else can link a vote to the voter who has cast a vote.
- **Unreusability:** A voter cannot vote more than once.
- **Fairness:** Early results should not be obtained, as they could influence the remaining voters.
- **Eligibility:** Only legitimate voters can vote.
- **Individual verifiability:** A voter can verify that his/her vote was really counted.
- **Universal verifiability:** Anybody can verify that the published outcome really is the sum of all votes.

In general, satisfying all properties is difficult to accomplish, and a particular focus on some properties may compromise the other properties.

1.2. Research Contribution

1.2.1 Contribution to RBAC

In this study, we aim to develop a practical system that uses Bitcoin as an infrastructure to realize the trans-organizational utilization of roles. We investigate a realization of a user-role assignment that is secure (users cannot disguise roles), user-oriented (users can disclose their roles to any organization), and open (anyone can verify if a user has a certain role that is managed and issued by another organization). The key ideas are to define correspondence between the roles issued by organizations and the users and to employ a challenge-response authentication protocol that will be used for verifying if a user really has an asserted role. Bitcoin's protocol and cryptography make the proposed system suitable for the trans-organizational utilization and authentication of roles, and furthermore, allow flexible role management operations, such as the endorsement and management of roles, with relatively small realization cost.

1.2.2 Contribution to E-voting

In this study, we investigate the idea of replacing the anonymous communication channel, which is assumed and needed in many e-voting protocols, with the propagation of Bitcoin transactions. Bitcoin transactions are communicated over a P2P network, and with appropriate management, Bitcoin users (not the Bitcoin addresses) cannot be identified and linked to transactions they create. This feature contributes to enabling anonymity in voting, which is a practical obstruction in many existing e-voting systems.

Bitcoin's protocol and cryptography make the proposed system suitable for an efficient and practical e-voting system wherein the voters are protected and given control over the important aspects of the voting process to protect their vote, privacy, and anonymity, while minimizing the trust and power of other entities to prevent them from performing malicious actions. For example, the administrator cannot introduce dummy votes and the counter cannot falsify the results because only the voters have access to the Bitcoin addresses that will be used for voting and the transactions that contain the important details of the voting process are recorded publicly in the blockchain.

We would like to note that it is not our intention to propose an e-voting scheme that is superior to all other existing schemes. Through this study, we would like to discuss what kind of problems in e-voting can be solved by using Bitcoin, and what kind of problems remain unsolved. The discussion will contribute to clarifying the potential of Bitcoin as a general infrastructure over which a secure application is constructed.

1.3. Dissertation Layout

The rest of this dissertation is organized as follows:

Chapter **2** introduces the Bitcoin protocol, including detailed discussion about Bitcoin addresses and wallets, transactions and transaction fees, blocks and the blockchain, mining and proof-of-work, and attacks on the Bitcoin network. This chapter also introduces how Bitcoin can be used as an infrastructure for RBAC and e-voting systems. Chapter **3** discusses RBAC and the different models associated with it. This section also presents the structure, procedures, and role

management features of the proposed RBAC system. Chapter 4 discusses the protocols used in existing e-voting systems, which are then analyzed to determine their strengths and vulnerabilities. This section also presents the proposed e-voting system and analyzes the security it provides based on the defined properties of e-voting systems. Chapter 5 provides the conclusion and a discussion of the future of Bitcoin, with a particular focus on blockchain technology.

Chapter 2

The Bitcoin Protocol

In October 2008, Satoshi Nakamoto uploaded the first specification and proof of concept of “a new electronic cash system that’s fully peer-to-peer, with no trusted third party” [21] in The Cryptography Mailing List. The paper is entitled “Bitcoin: A Peer-to-Peer Electronic Cash System” [11] and Nakamoto described the system with the following main properties:

- Double-spending is prevented with a peer-to-peer network.
- No mint or other trusted parties.
- Participants can be anonymous.
- New coins are made from HashCash style proof-of-work.
- The proof-of-work for new coin generation also powers the network to prevent double-spending.

In January 2009, the operation of Bitcoin commenced with the creation of the first block (genesis block). Shortly thereafter, Bitcoin has attracted the attention of many individuals worldwide and it continuously increases in value and popularity. Since the genesis block, Bitcoin has never been attacked successfully and has been operating continuously (i.e., no/zero downtime).

The main purpose of Bitcoin is to enable a payment system and complete digital money that is secure and decentralized; that is, it is a P2P network powered by its users and with no central authority. To achieve this, transactions

are publicly announced and the participants agree on a single history of these transactions. The transactions are grouped into blocks, given timestamps, and then published in some kind of public ledger. The hash of each block includes the hash of the previous block to form a chain, making accepted blocks difficult and almost impossible to alter.

Bitcoin is a collection of cryptographic protocols that allow secure online transactions between users [22, 23]. Bitcoin users commonly use digital wallets that handle the creation and storage of private keys and the corresponding public *Bitcoin addresses* as well as the sending and receiving of bitcoins. Bitcoin is open-source and is easily available for use on a wide range of computing devices, such as computers, laptops, and smartphones.

A user can send a certain value or amount of bitcoins to another user by creating a *transaction* with the sender's Bitcoin address/es as input/s and the receiver's Bitcoin address/es as output/s. Users can acquire bitcoins in multiple ways, including receiving bitcoins as payment for goods or services rendered, purchasing bitcoins at Bitcoin exchanges, using Bitcoin automated teller machines (ATMs), and earning bitcoins through competitive *mining*, as shown in Figures 2.1 and 2.2.

Transactions are validated by *miners* and then recorded in a global public ledger called the *blockchain*. Unlike traditional currencies, bitcoins are completely virtual and the ownership of bitcoins is referenced from the blockchain. The validation of transactions requires some amount of computation, and the miner who succeeds in validating transactions is rewarded or compensated with bitcoins and the transaction fees for his/her efforts. Validated transactions cannot be altered unless an attacker has computation power that overwhelms the total computation powers possessed by all other miners. The codes and algorithms used for the mining process are built-in in the Bitcoin protocol. The protocol also sets the rate at which new bitcoins are created, which is halved every 210,000 blocks or every 4 years, and limits the total number of bitcoins that will be created to a fixed 21 million bitcoins.



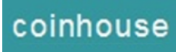













Rank	Site	Location +currency	Beginner- friendly	Trust score	Buy with...	User-vote	Score
TOP	 coinbase	San Fran, USA USD EUR GBP CAD	✓	A+	 + bank transf.	+ (127 Votes) ★★★★★	9.85
1	 coinhouse	Europe (EU) EUR	✓	A+	 + cash (neosurf) + Ethereum	+ (27 Votes) ★★★★★	9.80
2	 POLONIEX cryptocurrency exchange	Delaware, USA 75+ crypto pairs	✗	B+	 CRYPTO-CURRENCY	+ (87 Votes) ★★★★★	9.70
3	 LocalBitcoins	local all currencies	✓	A	 + paypal + bank transf.	+ (42 Votes) ★★★★★	9.65
4	 PAXFUL	Delaware, USA	✓	B+	 + debit card + onevanilla + amazon gift	+ (30 Votes) ★★★★★	9.50
5	 BITSTAMP	London, UK USD EUR	✓	A	 + bank transf.	+ (56 Votes) ★★★★★	9.30
6	 kraken	San Fran, USA USD EUR CAD GBP JPY	✓	A	 BANK TRANSFER + altcoins	+ (29 Votes) ★★★★★	9.15
7	 bit-square cryptocurrency exchange	p2p [decentralized] 59+ crypto pairs	✗	n/a	 CRYPTO-CURRENCY + bank transf.	+ (6 Votes) ★★★★★	9.15

Figure 2.1. A list of some of the top Bitcoin exchanges [24].

2.1. Bitcoin Addresses and Wallets

In Bitcoin, a user proves ownership of bitcoins through *digital key pair*, *Bitcoin address*, and *digital signature*. Bitcoin wallets are typically used for generating the key pair and the Bitcoin address and for creating the digital signature.

The key pair, which is created using public key cryptography, is composed of a private key and a public key. The private key is generated first, and then the public key is derived from the private key so that each private key corresponds to only one public key. The public key can be broadcast to anyone and is used to receive bitcoins, and the private key is used to sign transactions that allow the transfer of bitcoins stored in the corresponding public key. There is a mathematical relationship between the key pair that allows the private key to generate signatures on transactions. The signature can be used to prove ownership of the public key (hence the Bitcoin address) without revealing the private key. In other words, during a transaction, the sender presents the public key and generates a signature from the private key (the signature, although generated from



Figure 2.2. Bitcoin ATMs [25].

the same private key, is different each time) so that anyone can verify and accept the transaction as valid.

The private key is the most important part that is necessary for controlling the bitcoins associated with the public key. Therefore, the private key should always be kept secret and never revealed to anyone else (as that person would have access to the bitcoins). Equivalently, the private key should not be lost or forgotten because otherwise the bitcoins stored in the corresponding public key would be lost forever as well.

2.1.1 Generating the key pair

The private key is an integer between 1 and $n-1^*$ and the public key is derived from the private key using Elliptic Curve Digital Signature Algorithm (ECDSA) [26], which is a one-way cryptographic function. The private key can be generated in multiple ways, including manual generation by coin tosses or digital generation by introducing a secure source of entropy or randomness, as the goal is to generate the private key “as random as possible”.

2.1.2 Bitcoin Addresses

A Bitcoin address is 160-bit hash of the public key of an ECDSA key pair. The public key undergoes several cryptographic processes to be converted into a valid Bitcoin address. First, a Secure Hash Algorithm (SHA), specifically SHA-256 which produces an almost-unique and fixed size of 256-bit (32-byte) number**, is performed on the public key. Then, a RACE Integrity Primitives Evaluation Message Digest (RIPEMD) hash, specifically RIPEMD160 which produces a 160-bit (20-byte) number [27], is performed on the result of the SHA-256. Then, a version is added in front of the RIPEMD160 hash as prefix (0x00 for the Main Network and 0x6f for the Test Network).

Bitcoin addresses are always presented to users in Base58Check Encoding, which includes a checksum to avoid ambiguity and errors. To implement this,

* n is a constant that is slightly less than 2^{256} and is defined as the order of the elliptic curve defined in the Bitcoin specification.

**To further show that SHA-256 and hash algorithms, in general, are state-of-the-art algorithms that provide security, another paper of the authors is presented in Appendix A.

two rounds of SHA-256 hash operations are performed on the extended version of the RIPEMD160 hash (with the prefix). The first 4 bytes of the result of the double SHA-256 hash is the checksum for the Bitcoin address. This checksum is appended at the end of the extended RIPEMD160 hash to obtain a 25-byte binary Bitcoin address. Finally, the byte string is converted into a Base58 string using Base58Check encoding to obtain the commonly used and distributed Bitcoin address format. This format is composed of random number and letters, for example **19zBWfkNicdLdTTweZe37XRj2aFoYmHEX6**. A summary of the conversion is shown in Figure 2.3.

2.1.3 Wallets

Bitcoin wallets are used by most Bitcoin users as containers to store their private keys (and the corresponding public keys and Bitcoin addresses) and to perform different functions, such as generating signatures, sending and receiving bitcoins, and logging their transactions. Users can choose from different kinds of wallets depending on how they want their key pairs, and therefore bitcoins, to be handled. Some of the more frequently used wallets are discussed below.

Nondeterministic Wallets

Nondeterministic wallets generate random private keys as needed by the user. Therefore, the user must keep copies of all generated private keys and perform a backup of these keys as frequent as the keys are generated.

Deterministic Wallets

Deterministic wallets, also referred to as “seeded” wallets, contain private keys that are all derived from a seed. These private keys are derived by using a one-way hash function. The seed is a number that is usually generated at random and then combined with other data, such as an index number, to derive the private keys. In this kind of wallet, only the seed is necessary to generate and recover all derived private keys. Therefore, only one backup is necessary, i.e., only the seed is backed up one time.

Elliptic-Curve Public Key to BTC Address conversion

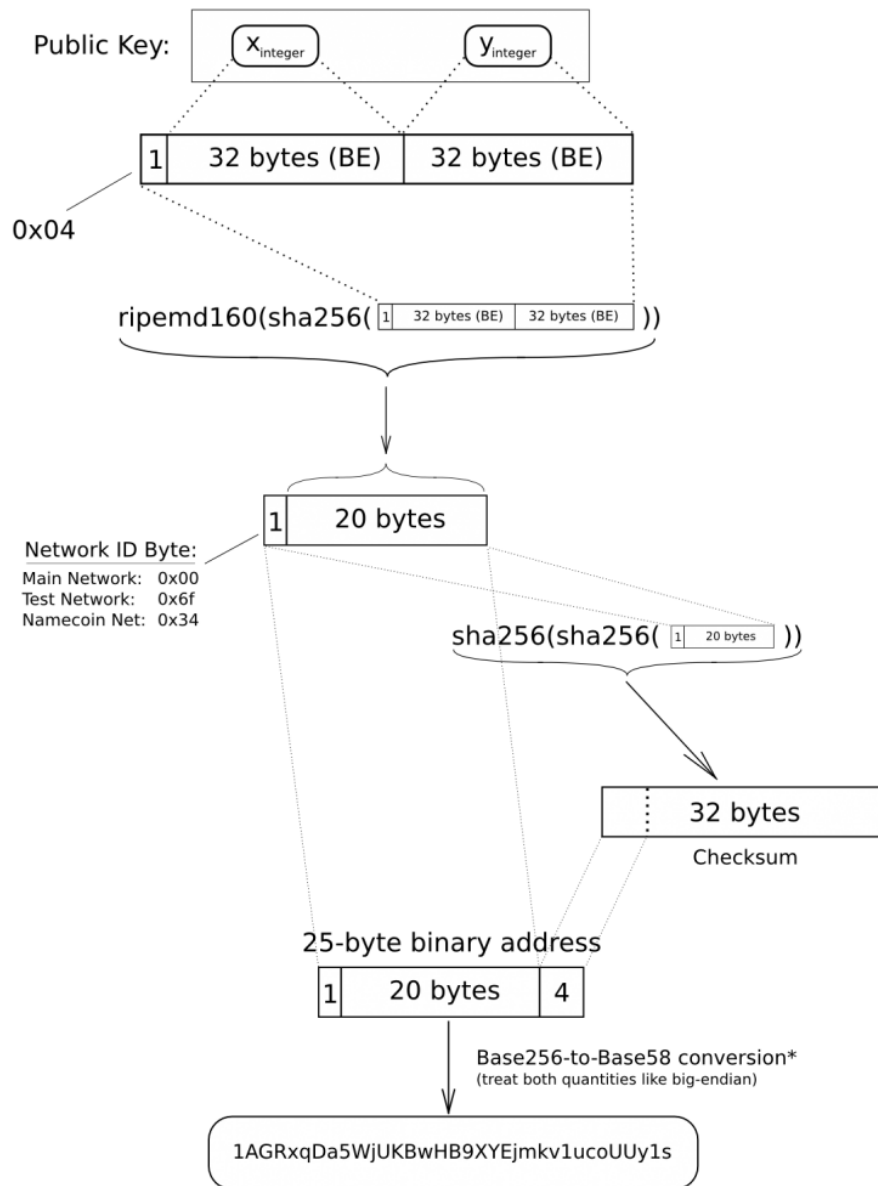


Figure 2.3. Conversion from the public key to Bitcoin Address [28].

Types of Wallets

Bitcoin users have several options for wallets to choose from [29].

- **Software wallets** are wallets that are installed in a device, such as desktop or laptop, that the user has access to. Depending on the source of the software wallet, users may be required to download the entire blockchain, which is at least 90 GB in size and growing.
- **Web wallets** are wallets provided by third-party wallet service companies and are only accessible through the Internet. Web wallets provide flexibility to users as users can access their web wallets anytime, anywhere, and using any device as long as there is an Internet connection. The disadvantages, however, are that users have no access to the wallet without Internet connection and that the service provider is prone to attacks, which can compromise the users' bitcoins. An example of a web wallet accessed in an Internet browser is shown in Figure 2.4.
- **Cold wallets** are wallets that are not connected to the Internet. These cold wallets could be in the form of paper (users print out or write the private keys in a piece of paper) or removable storage devices (e.g., USB sticks and hard disks).
- **Brain wallets** are a kind of cold wallet that are stored in the memory of users. Randomly computer-generated words or pass phrases only the user would know are used to derive private keys and are not stored anywhere

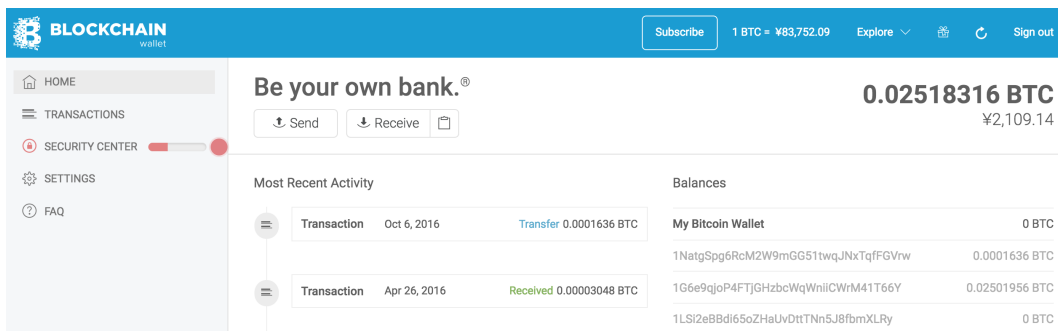


Figure 2.4. A web wallet from Blockchain.info.

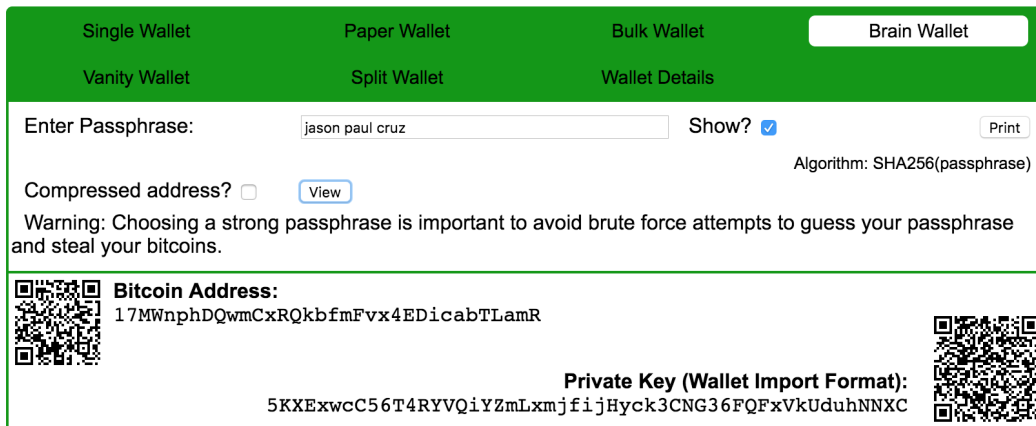


Figure 2.5. A brainwallet with pass phrase of “jason paul cruz” from bitaddress.org.

else (only in the brain or memory of users). Figure 2.5 shows a brain wallet generated from a pass phrase using an online Bitcoin address generator.

- **Hardware wallets** are wallets that can only be accessed with the presence of the hardware device (typically a USB stick) where the wallets are stored.

A new ECDSA keypair is generated for each Bitcoin address, and thus, users typically maintain multiple Bitcoin addresses for different transactions. In fact, it is recommended by the Bitcoin community to use a Bitcoin address only once to ensure anonymity and security. Users can create any number of Bitcoin addresses easily and for free. There are 2^{160} total possible Bitcoin addresses that can be created, and thus, a Bitcoin address is considered to be “unique” as it is extremely unlikely for two users to independently generate the same Bitcoin address.

2.2. Transactions

Transactions are digitally signed data that represent the transfer of bitcoins from a Bitcoin address/es to another. Transactions are broadcast to the Bitcoin network and then included in a block in the blockchain. Transactions can be considered to be the most important part of Bitcoin and the other parts and protocols are used

Field	Description	Size
Version no	currently 1	4 bytes
In-counter	positive integer $VI = \text{VarInt}$	1 - 9 bytes
list of inputs	the first input of the first transaction is also called "coinbase" (its content was ignored in earlier versions)	<in-counter>-many inputs
Out-counter	positive integer $VI = \text{VarInt}$	1 - 9 bytes
list of outputs	the outputs of the first transaction spend the mined bitcoins for the block	<out-counter>-many outputs
lock_time	if non-zero and sequence numbers are < 0xFFFFFFFF: block height or timestamp when transaction is final	4 bytes

Figure 2.6. Structure of a Bitcoin transaction [30].

to ensure that transactions are properly created, propagated to miners, validated, timestamped, and finally included in the blockchain.

2.2.1 Transaction format

A transaction is basically an encoded data structure for the transfer of bitcoins from the *sender* (input address/es) to the *receiver* (output address/es). A transaction contains different fields, as shown in Figure 2.6.

2.2.2 Transaction Inputs and Outputs

Inputs are generally references to outputs from previous transactions. An output from a previous transaction is also called *unspent transaction output* (UTXO). UTXOs are indivisible chunks of bitcoins that correspond to Bitcoin addresses and are recorded in the blockchain. In other words, the bitcoins received by users are recorded as UTXOs in the blockchain, making them easy to track and verify. When a user wants to send bitcoins to another user, he/she creates a transaction using UTXOs locked to his/her Bitcoin address/es as inputs. A transaction represents a transfer of the value of bitcoins from one Bitcoin address to another, and thus the UTXOs used as inputs are still the same UTXOs but locked at a new Bitcoin address. Technically, outputs come first because *coinbase* transactions have no inputs and create outputs from “thin air”. A coinbase transaction [31] is a special kind of transaction that is the first transaction added in a block. The coinbase transaction contains the Bitcoin address of the miner where the reward for mining the block will be sent to**.

**The genesis block famously contains the text “The Times 03/Jan/2009 Chancellor on brink of second bailout for banks”, which is probably intended to be a proof that no bitcoins existed

Outputs contain the instructions for the sending of bitcoins. Outputs are composed of two parts:

- **Amount** of bitcoins, denominated in *Satoshi* (10^{-8}), which is the smallest unit of bitcoins. 1 BTC = 100,000,000 Satoshi.
- **ScriptPubKey** or a locking script, which “locks” the amount of bitcoins with conditions that must be met to spend the UTXOs.

Multiple outputs can be specified as long as they amount to the total or combined value of the inputs. For example, if the input (UTXO) is worth 10 BTC, and the user wants to send 6 BTC to another user, two outputs worth 6 BTC and 4 BTC are created— 6 BTC are sent to the receiving Bitcoin address and 4 BTC will be sent back to one of the sender’s Bitcoin address (this is also known as “change” that a sender sends back to him/herself). Any input bitcoins that are not redeemed in the output will be considered as a *transaction fee*, which is a reward to the miner who validates the transaction.

2.2.3 Transaction Fees

Transaction fees are included in most transactions and serve as compensation to the miners who validate transactions and secure the Bitcoin network. Some transactions can be processed without any transaction fee included if certain conditions are met [32]. Transaction fees serve as incentives for miners to include transactions in the next block. Transaction fees also serve as a disincentive against “dust” transactions, which are transactions with outputs of less than 546 satoshi or 0.00000546 BTC. Dust transactions require a transaction fee of 0.0001 BTC to prevent spamming of transactions and any kind of abuse in the system.

The required transaction fees are calculated based on the size of the transaction in kilobytes and not on the amount of bitcoins being transferred. Additional transaction fees are voluntary on the part of the users creating bitcoin transaction, that is, miners may prioritize transactions with higher transaction fees included in them and even ignore transactions without any transaction fees. Therefore, the lower transaction fees may be delayed and validated on a best-effort basis

before 2009.

(transactions without transaction fees may even never get validated, at the worst case).

In the future, when the total 21 million bitcoins have been created and distributed, the entire Bitcoin network will solely run on transaction fees as compensation and incentive for miners to keep validating transactions and securing the network.

2.2.4 Transaction Details

Bitcoin transactions contain the following information:

- **Transaction ID** is a unique ID used for identifying a transaction in the blockchain.
- **List of input addresses** containing the UTXOs and outputs of previous transactions from which bitcoins will be transferred.
- **Output addresses** containing the Bitcoin addresses that will receive bitcoins.
- **Amount** of bitcoins being transferred.
- **Timestamp** of the blocks, which includes the time the transactions are received in the network and the time they are validated and included in the blockchain.
- **Size** of the transaction in bytes.
- **Block number** of the block where the transaction is included (other transactions will be included in the same block).
- **Script** containing the conditions that need to be satisfied before the bitcoins to be received can be spent, including the details about the coinbase.

Figure 2.7 shows an example of a transaction, as viewed from a blockchain browser.

Transactions are duplicated and broadcast over the P2P network of Bitcoin consisting of Bitcoin users (nodes). The Bitcoin network itself does not have the

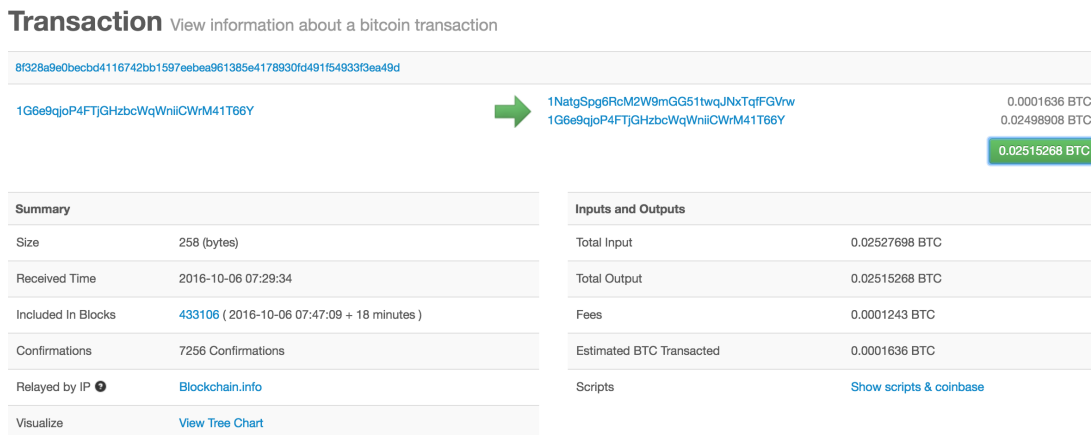


Figure 2.7. Example of a transaction viewed from a blockchain browser.

mechanism to conceal the IP addresses of the source of a transaction, and, in theory, an IP address can happen to be connected to a Bitcoin address. However, such a risk can be mitigated because full node clients relay transactions as if they are the owner of transactions, and thus the source of a particular transaction can be difficult to determine unless all communications logs of all nodes are analyzed and traced. Furthermore, third-party services, such as The Onion Router (Tor) [33], can be used to hide the IP address of a computer used in Bitcoin transactions. Online mixing services, such as BitLaundry [34], can also be used, wherein users send and receive bitcoins to and from such service using independent Bitcoin addresses.

There is a risk that a Bitcoin user is identified if that user uses a Bitcoin address in a naive manner. Bitcoin addresses look like random numbers and letters, and the identity of the owner remains unknown unless the same Bitcoin address is used in other transactions where information about the owner is revealed (e.g., buying a product that is delivered physically wherein the name and home address of the owner are provided in the product's shipping information). This issue can be avoided if good practices are adopted by Bitcoin users [35]; simply, by using a Bitcoin address only once. A user should not publish his/her Bitcoin addresses in such a way that somebody can connect these addresses with other Bitcoin addresses that are intended for private use. For example, a user is not recommended

to move funds from published Bitcoin addresses to another Bitcoin address that he/she owns. This issue will be revisited later in the discussion about the security.

2.3. Blocks and the Blockchain

Transactions are grouped together in blocks and then included in the blockchain. The blockchain is a data structure containing linked blocks of transactions, and it can be stored as a simple database. Each block contains the hash of the previous block to create a chain that connects the genesis block to the current block. Each block is identified by the hash of its header, which is generated using SHA-256. Figure 2.8 shows how three blocks are linked by references in the previous block hash to form a chain.

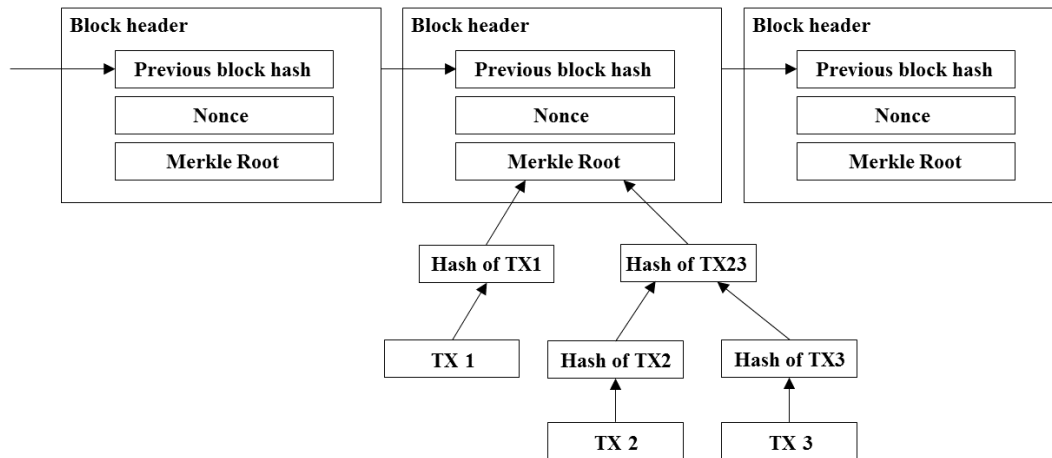


Figure 2.8. Simplified representation of the Bitcoin blockchain.

2.4. Block structure

A block is a container data structure that collects transactions to be included in the blockchain. A block is composed of a block size, block header, transaction counter, and the transactions, as shown in Figure 2.9.

Field	Description	Size
Magic no	value always 0xD9B4BEF9	4 bytes
Blocksize	number of bytes following up to end of block	4 bytes
Blockheader	consists of 6 items	80 bytes
Transaction counter	positive integer $VI = \text{VarInt}$	1 - 9 bytes
transactions	the (non empty) list of transactions	<Transaction counter>-many transactions

Figure 2.9. Structure of a block [36].

The block header contains, among other things, a hash of the previous block, a hash of the Merkle root of valid transactions to be included in this block, a timestamp (creation time of the block), the current difficulty target for the block, and a nonce (a unique solution used for the PoW algorithm). The structure of the block header is shown in 2.10, and the details will be discussed in a latter subsection.

Field	Purpose	Updated when...	Size (Bytes)
Version	Block version number	You upgrade the software and it specifies a new version	4
hashPrevBlock	256-bit hash of the previous block header	A new block comes in	32
hashMerkleRoot	256-bit hash based on all of the transactions in the block	A transaction is accepted	32
Time	Current timestamp as seconds since 1970-01-01T00:00 UTC	Every few seconds	4
Bits	Current <i>target</i> in compact format	The <i>difficulty</i> is adjusted	4
Nonce	32-bit number (starts at 0)	A hash is tried (increments)	4

Figure 2.10. Structure of a block header [37].

The blockchain goes all the way back to the genesis block, which was created in 2009. Each block contains a summary of all transactions that it contains. The transactions are combined using a *Merkle tree*, also known as a *binary hash tree*. A Merkle tree is a data structure that efficiently summarizes and verifies the integrity of large amounts of data. The term “tree” pertains to a computer science term that describes a branching data structure, where the “root” (the hashMerkleRoot field in the block header) is at the top and “leaves” are hashes of pairs of transactions. Figure 2.11 shows how individual transactions form a Merkle tree. Merkle trees are used in the Bitcoin protocol for its efficiency, wherein an element in N data elements can be queried with at most $2 \cdot \log_2(N)$ calculations.

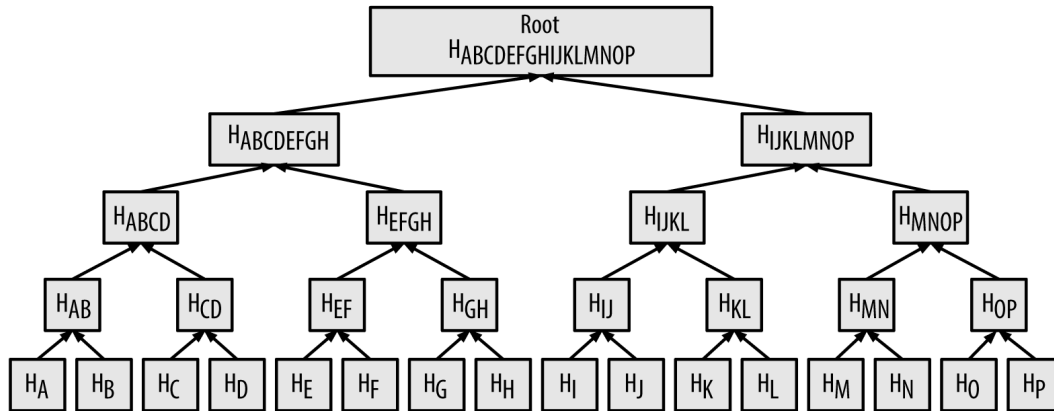


Figure 2.11. Example of how the Merkle tree of Bitcoin transactions is created [38].

2.5. Mining and Proof-of-Work

Blocks are added to the blockchain through the process called *mining*, which uses a proof-of-work system wherein miners participating in the Bitcoin network use customized software and hardware to solve mathematical problems. Mining also governs the issuance of new bitcoins into the bitcoins supply and secures the Bitcoin network against fraudulent transactions, such as double-spending. The miner who solves a block is awarded with newly “minted” bitcoins (currently at 12.5 BTC \approx 16,000 USD) and the transaction fees of the transactions included in the solved block. This process of mining incentivizes miners to keep mining and approving transactions.

The amount of bitcoins being generated by a miner decreases every after 210,000 blocks (approximately every 4 years). This amount started at 50 BTC per block when Bitcoin was deployed in January 2009, and then halved to 25 BTC per block in November 2012, and then again halved to 12.5 BTC per block in July 2016. The reward is estimated to half again some time in July 2020. Based on this rate, the issuance of new bitcoins will stop in the year 2140, which is when all 21 million bitcoins will have been generated. After 2140, the reward for mining will solely come from the transactions fees included in the transactions.

2.5.1 Transaction Verification

After a user creates a transaction, the transaction is forwarded to neighboring nodes in the Bitcoin network until it is propagated across the entire Bitcoin network. As per protocol, each node needs to verify the correctness of a transaction before forwarding it to its neighbors, ensuring that only valid transactions are propagated and included in a block (invalid transactions are discarded by a node that encounters them). Each node verifies every transaction based on a checklist of criteria as follows [38]:

- The transaction's syntax and data structure must be correct.
- Neither list of inputs or outputs is empty.
- The transaction size in bytes is less than `MAX_BLOCK_SIZE`.
- Each output value, as well as the total, must be within the allowed range of values (less than 21 million and more than 0 bitcoins).
- None of the inputs have `hash=0, N=1` (coinbase transactions should not be relayed).
- `nLockTime` is less than or equal to `INT_MAX`.
- The transaction size in bytes is greater than or equal to 100.
- The number of signature operations contained in the transaction is less than the signature operation limit.
- The unlocking script (`scriptSig`) can only push numbers on the stack, and the locking script (`scriptPubkey`) must match `isStandard` forms (this rejects "non standard" transactions).
- A matching transaction in the pool, or in a block in the main branch, must exist.
- For each input, if the referenced output exists in any other transaction in the pool, the transaction must be rejected.

- For each input, look in the main branch and the transaction pool to find the referenced output transaction. If the output transaction is missing for any input, this will be an orphan transaction. Add to the orphan transactions pool, if a matching transaction is not already in the pool.
- For each input, if the referenced output transaction is a coinbase output, it must have at least COINBASE_MATURITY (100) confirmations.
- For each input, the referenced output must exist and cannot already be spent.
- Using the referenced output transactions to get input values, check that each input value, as well as the sum, is in the allowed range of values (less than 21 million and more than 0 bitcoins).
- Reject if the sum of input values is less than sum of output values.
- Reject if transaction fee would be too low to get into an empty block.
- The unlocking scripts for each input must validate against the corresponding output locking scripts.

Miners independently verify all transactions they receive before propagating them to other nodes. Therefore, every miner builds a pool of valid transactions as they come in. At the same time, verified transactions are added to a *memory pool*, which is where transactions are stored and wait until miners include and validate them into a block (mined block).

2.5.2 Transaction Priority and Fees

Nodes select transactions from the memory pool based on a priority criteria. Transactions are prioritized based on the age of the UTXOs in the input, that is, old and high-value UTXOs are prioritized over new and low-value UTXOs.

The priority of a transaction is calculated using the following formula:

$$Priority = \frac{Sum(Value\ of\ input * Input\ age)}{Transaction\ Size} \quad (2.1)$$

where the sum of the value and age of the inputs is divided by the total transaction size. The value is measured in satoshi, the age is the number of blocks that have elapsed since the UTXOs were recorded in the blockchain, and the transaction size is measured in bytes. A transaction is considered “high priority” if its priority is greater than 57,600,000, which corresponds to 1 BTC (100 million satoshi), age of 144 blocks (1 day), and total transaction size of 250 bytes following Equation 2.1.

Any transaction not included in the current block will remain in the memory pool until it is included in a succeeding block. This happens because miners can choose to ignore transactions without transaction fees and therefore these kinds of transactions are validated on a best-effort basis.

2.5.3 Proof-of-Work Algorithm

The process of mining uses a PoW algorithm that involves performing a SHA-256 to the block header repeatedly, changing one parameter (the nonce), until the resulting hash satisfies the difficulty target. Since SHA-256 is a one-way function, the resulting block header hash cannot be determined in advance and no pattern can be discerned as each hash is independent from each other. Therefore, the solution requires “brute force” solution; that is, miners scan and test for a nonce repeatedly, usually the nonce is incremented by one for each solution.

The SHA-256 algorithm takes an input of arbitrary length and outputs a data of fixed length. The resulting hash of a specific input will always be the same, and thus, it can be easily calculated and verified by anyone by using the same hashing algorithm. On the other hand, it is almost impossible to retrieve the input given a hashed output. Moreover, in a cryptographic hashing algorithm, it is almost impossible to find two different inputs that result in the same output, also known as a collision.

In SHA-256, the output is always 256-bits long. To show the randomness of the output, Figure 2.12 shows the results of applying SHA-256 to the phrase “Hello, world!” appended with different integer values [39]. In the mining process, the appended integer is the nonce and is used to vary the resulting hash.

To make the hashing algorithm challenging, a target is set (e.g., the resulting hash in the example should start with at least three zeros). Looking at the

```

"Hello, world!0" => 1312af178c253f84028d480a6adc1e25e81caa44c749ec81976192e2ec934c64
"Hello, world!1" => e9afc424b79e4f6ab42d99c81156d3a17228d6e1eef4139be78e948a9332a7d8
"Hello, world!2" => ae37343a357a8297591625e7134cbea22f5928be8ca2a32aa475cf05fd4266b7
...
"Hello, world!4248" => 6e110d98b388e77e9c6f042ac6b497cec46660deef75a55ebc7cfd65cc0b965
"Hello, world!4249" => c004190b822f1669cac8dc37e761cb73652e7832fb814565702245cf26ebb9e6
"Hello, world!4250" => 0000c3af42fc31103f1fdc0151fa747ff87349a4714df7cc52ea464e12dcd4e9

```

Figure 2.12. Results of applying SHA256 to a phrase with different integers appended to it.

example in Figure 2.12, 4,251 rounds of SHA-256 operations were performed to find a nonce that when concatenated with “Hello, world!” produced an output that starts with at least three zeros. In mining, the resulting hash is governed by a parameter called the difficulty target, which is agreed upon by miners. The difficulty target is expressed as the difficulty on creating the current block compared to generating the genesis block and is determined as follows:

$$\text{difficulty} = \frac{\text{difficulty_1_target}}{\text{current_target}} \tag{2.2}$$

As of writing this manuscript, the difficulty is 460,769,358,091 (the probability of each hash to be a valid solution is 5.053084359146814E-22) [12, 40, 41]. The correct nonce should produce a hash value whose numerical interpretation is lower than the difficulty target, or equivalently the hash should start with a certain number of zeros. When a miner finds the correct nonce, it forwards the solved block to the rest of the miners. After validating the solution for the block, miners move on to determining the correct nonce for the next block. To compensate for increasing hardware speeds, the difficulty target is adjusted every 2,016 blocks so that it takes on average 10 minutes to find a valid nonce.

The security of Bitcoin relies on this PoW system, which inherently means that a block cannot be modified without redoing the work spent on it, including

the work spent on blocks chained after it. Given this design, as long as majority of the overall computing power participating in the Bitcoin network is controlled by honest miners, an attacker will be outpaced by the honest miners, making it almost impossible to modify a published block.

2.6. Bitcoin as Infrastructure

Bitcoin, with blockchain technology as one of its most innovative solutions, has established itself as an effective online payment system. Its security and decentralized nature make it applicable for use other than a payment system.

2.6.1 RBAC

This study develops a practical RBAC system that uses Bitcoin technology to realize the trans-organizational utilization of roles. We investigate a realization of a user-role assignment that is secure (users cannot disguise roles), user-oriented (users can disclose their roles to any organization), and open (anyone can verify if a user has a certain role that is managed and issued by another organization). The key ideas are to define correspondence between the roles issued by organizations and the users and to employ a challenge-response authentication protocol that will be used for verifying if a user really has an asserted role. The relationship between the role issued by organizations and the users will be represented by the transfer of bitcoins using the organizations' Bitcoin address/es as input and the users' Bitcoin address/es as output. We want to show that the relation of payment using bitcoins can translate or represent a relation of trust.

Bitcoin's protocol and cryptography make the proposed system suitable for the trans-organizational utilization and authentication of roles, and furthermore, allow flexible role management operations, such as the endorsement and management of roles, with relatively small realization cost.

2.6.2 E-Voting

This study investigates the use of the Bitcoin protocol as a substitute to a public bulletin board to provide a secure, anonymous, and transparent e-voting system.

In the proposed system, the voters are given the “power” and control over the sensitive parts of the voting process and the trust to other authorities is minimized. In this way, the voters have complete control over their respective votes, and the administrator and counter cannot perform malicious operations because the security of Bitcoin is intact and the transactions are published publicly. In the ideal case, the proposed e-voting system assumes the following:

1. All entities are knowledgeable about Bitcoin, including the protocol and creation of transactions.
2. All entities handle their private keys securely.
3. Voters have initial bitcoins to spend.

For the first assumption: this is currently difficult to imagine as Bitcoin technology is not easy to understand fully. However, just like any e-voting system, the protocols and methods need to be taught to the voters in an effective manner, similar to teaching Bitcoin. Bitcoin is continuously becoming more widespread and it is currently being used for monetary transactions, proving that the general public can learn the proposed system effectively. Furthermore, this issue can be avoided if an excellent wrapper mechanism that hides the details of Bitcoin from the view of users can be created (e.g., an application or digital wallet that can be used solely for the voting process).

For the second assumption: in a general e-voting system, the voters would need to handle keys (one for encrypting the vote and one used as the blinding factor) privately, securely, and secretly. In the proposed system, the voters would need to handle an additional key, the private key for the Bitcoin address that will be used for voting. The Bitcoin community has provided many methods for securing Bitcoin wallets, including performing backups of wallets (local and online), encrypting the wallet by setting passwords, and creating off-line wallets or cold storage [42]. A technology-savvy Bitcoin user can even create multi-signature addresses, e.g., a 2-of-3 multi-signature address, wherein bitcoins can be transferred using 2 out of 3 private keys used in the creation of the said address. In this aspect, using Bitcoin is advantageous because it provides many security tools that are immediately available.

For the third assumption: this is difficult to implement until Bitcoin becomes a widely accepted currency. In general, users can obtain bitcoins through Bitcoin

ATMs, online and off-line exchanges, accepting bitcoins as payment, by buying from friends, and through mining. In the proposed system, it is ideal that the administrator (or government) provides all necessary Bitcoin expenses, so that the voters do not need to spend any money. However, a voter cannot reveal the Bitcoin address that he/she will use for voting to any other entity without compromising his/her anonymity and privacy. The most viable option is for the administrator to provide Prepaid Bitcoin Cards (PBCs) [43] to all eligible voters. PBCs are physical or virtual cards that are pre-loaded with bitcoins. A PBC contains a public Bitcoin address with a pre-loaded amount of bitcoins and the corresponding private key, which is covered and can be scratched off, as shown in 2.13. The PBCs that will be used for the proposed e-voting system can be created by a third-party or by the administrator itself.

2.7. Attacks on the Bitcoin Network

Some strategies, both theoretically and in practice, have been devised to attack the security of the Bitcoin protocol and possibly put it in danger. Some attacks have been designed for dishonest or rogue miners, i.e., those who do not follow the Bitcoin protocol, to get rewards higher than their contribution to the network.

These strategies include the pool hopping attack [44], the mining cartel attack [45], selfish mining [46], block withholding attack [47], and hardware attacks. These attacks are designed to infiltrate factors outside the blockchain, targeting the client side and stealing bitcoins from them (pool and wallet infiltrations). These attacks mainly aim to steal bitcoins and/or gain higher rewards and not to modify transactions or the blockchain. Therefore, Bitcoin's security remains intact and "backed by math". For our purposes, the transactions between organizations and users will remain secure and unmodifiable because they are recorded in the blockchain.



Figure 2.13. Example of a Prepaid Bitcoin card [43].

Chapter 3

Role-Based Access Control (RBAC)

Role-based access control (RBAC), also called role-based security, is a method used in computer systems security to restrict system access based on user authorization. RBAC is currently being employed by various enterprises and companies and it provides solutions to the security needs of commercial and government organizations. RBAC can be deployed to facilitate and handle the security of systems in companies, even large-scale companies with hundreds or even thousands of users and permissions. RBAC started to gain popularity in 1994, when various IT vendors, including IBM, Sybase, and Siemens have developed products based on the RBAC model. In 2000, the Ferraiolo-Kuhn model was integrated with the framework of Sandhu et al. [14] to create a unified model for RBAC, published as the NIST RBAC model [48] and adopted as an ANSI/INCITS standard in 2004 [49].

3.1. Models for RBAC

Among the many technical issues of the RBAC framework, this study mainly focuses on the realization of the user-role assignment in a trans-organizational scenario. Other issues of RBAC may be related to this study, but they are excluded from the scope of our discussion. To clarify the position of our study in the entire framework of RBAC, an abstract model of RBAC and its extension

are discussed first.

In the simplest model of the RBAC [14], the access structure is defined by three sets and two relations; the set U of *users*, the set R of *roles*, the set S of *services*, a *user-role assignment* $UA \subset U \times R$, and a *role-service assignment* $SA \subset R \times S$. A user u is eligible to access a service s if and only if there is a role r such that $(u, r) \in UA$ and $(r, s) \in SA$. In real-world services, roles can be used in a trans-organizational manner. A role that was issued by a *role-providing entity* can be referred by a foreign *service-providing entity* to determine if a service should be given to an unknown guest. An interesting point here is that the role-providing entity is not always concerned about the service-providing entities. This suggests that the service-providing entity is not always allowed access to the user-role assignment, and thus, it needs to devise an alternative means to confirm if an unknown guest has a certain role or not. To deal with this kind of framework, we consider extending the basic model of RBAC by introducing a set of organizations.

The *trans-organizational RBAC* is defined similarly to the usual RBAC, but a set O of *organizations* is defined in addition to the sets of users, roles, and services. Furthermore, the set R of roles is partitioned into several subsets, with each subset of R associated with an element in O , that is, $R = R_{o_1} \cup \dots \cup R_{o_n}$, where $o_1, \dots, o_n \in O$ and $R_{o_i} \cap R_{o_j} = \phi$ if $i \neq j$. To make the relation among roles and organizations explicit, a role r in R_{o_1} is written as $o_1.r$. Similarly, the user-role assignment UA is partitioned into disjoint subsets; $UA = UA_{o_1} \cup \dots \cup UA_{o_n}$, where $UA_{o_i} \subset U \times R_{o_i}$. Obviously, $o_1.r \in R_{o_i}$ means that the role $o_1.r$ is managed by the organization o_i and the assignment of users to $o_1.r$ is controlled by that organization o_i . In the trans-organizational RBAC, a user u demands a service s by asserting his/her role $o_1.r \in R_{o_i}$ that has been provided by a role-providing entity (organization) o_1 . The service-providing organization provides the service to the user if and only if $(u, o_i.r) \in UA_{o_i}$ and $(o_i.r, s) \in SA$. Note that the test of $(o_i.r, s) \in SA$ is easy for the service-providing organization because the assignment SA is defined by the organization itself. On the other hand, the test of $(u, o_i.r) \in UA_{o_i}$, which is sometimes called an *authentication*, is not as obvious as the test of $(o_i.r, s) \in SA$ because the assignment UA_{o_i} is defined by a foreign service-providing organization.

The trans-organizational RBAC will be realistic only if the authentication of roles $(u, o_i.r) \in \text{UA}_{o_i}$ is accomplished. Physical certificates, such as passports and ID-cards, have been widely used for many years to authenticate identities and roles, but these certificates cannot be easily imported to the digitalized world over a computer network. Digital certificates have been studied for the replacement of physical certificates [15], but they are not widely accepted because of the cost issues for acquiring these certificates, keeping related keys secure, and maintaining a public-key infrastructure (PKI) [50, 51] that should be always available. A less sophisticated but simpler approach is to arrange a mutual agreement between role-providing organizations and service-providing organizations. However, such a framework will be semi-closed and only include organizations that share identical benefits.

3.1.1 Related System based on Hierarchical ID-Based Encryption

We have investigated another approach for realizing secure authentication of roles by utilizing a special cryptography known as hierarchical ID-based encryption [52]. In this previous study, a practical mechanism that realizes the trans-organizational utilization of roles was developed. The crucial point of this realization is to make use of hierarchical ID-based encryption (HIBE) [53, 54], which allows an arbitrary string to be used as a public encryption key. The key idea is to define correspondence between the roles and keys of HIBE and to employ a challenge-response authentication protocol that will be used for verifying if a user really has an asserted role. The hierarchical nature of HIBE makes the proposed scheme suitable for the trans-organizational utilization of roles, and furthermore, allows flexible role management operations, such as the endorsement and delegation of roles. A prototype system of the proposed trans-organizational RBAC was also introduced. To make the trans-organizational RBAC system practical, it is highly desirable for a user to be able to carry his/her roles all the time. Therefore, the proposed system was implemented on Android-enabled mobile devices to verify the practicality of the proposed scheme. The implementation contains the realization of cryptographic functions that are essential for

handling cryptographic keys and the development of a scheme that allows two devices to perform the challenge-response authentication by utilizing local and closed communication. The prototype demonstrates that the proposed scheme is simple, lightweight, and completely practical for realizing the trans-organizational RBAC.

The proposed system offers some advantages over other existing approaches, but it necessitates several users with the same role to have an identical secret key, which is not favorable from a security viewpoint. Other comparable systems include decentralized multi-authority systems for attribute-based encryption (MA-ABE) and attribute-based signatures (MA-ABS) [56, 57]. The MA-ABE in [56] is decentralized but requires a trusted setup of common reference parameters. The MA-ABS in [57] is also decentralized and does not necessitate a trusted setup, but it requires the setting of a public parameter for a prime order bilinear group and hash functions. Even though these schemes are decentralized, all users and organizations must agree on the parameters first. Confusion and implementation problems may arise if several communities use different parameters, and therefore, such systems must be initiated by somebody who has strong leadership, grand design, and sufficient financial power for implementation. Consequently, a scheme for a role authentication mechanism that is secure, practical, and easy to set up has yet to be established.

A more comprehensive explanation of the previous study is presented in Appendix B.

3.2. Proposed RBAC System using the Bitcoin Network

The proposed system is a non-conventional authentication mechanism that is suitable for the trans-organizational utilization of roles. The idea is to provide an irrefutable proof of the role of a user issued by an organization by verifying the connection of the user to the organization through the Bitcoin blockchain. Consider for example that A-university would like to manage a “student” role for its students. First, it would perform a Bitcoin transaction using its own public Bitcoin address/es as input/s and the corresponding students public Bitcoin

address/es as output/s. Upon request for a service from an unknown user who asserts that he/she possesses the student role of A-university, a service-providing organization, for example a restaurant, will verify the Bitcoin transaction containing the Bitcoin addresses of A-university and the student, which connects the student role managed by A-university to the output address in the transaction. After establishing the connection, the restaurant can verify (through a challenge-response protocol) if the unknown user has access to the output address in the transaction, which finally connects the student role from A-university to the unknown user.

Note that the restaurant does not have to know anything about the role beforehand, and does not have to make any contract or inquiry to A-university that has assigned the role to the student because the details needed by the restaurant are published publicly and/or possessed by the user (details below). In the proposed system, there is no essential difference between users and role-issuing organizations because they both can be the sender and receiver in the Bitcoin transactions (but for simplicity and explanatory purposes, the role-issuing organizations will be differentiated from the users).

3.2.1 Procedures

Figure 3.1 shows the overall structure of the proposed system. In this model, we assume that the role-issuing organizations are Bitcoin users while the users and service-providing organizations may or may not be Bitcoin users. Bitcoin user means that the entity owns a Bitcoin wallet and performs Bitcoin transactions.

Prerequisites

An organization (o_1) generates n Bitcoin addresses, where n is the number of roles that o_1 wants to manage. The creation of these Bitcoin addresses (and the corresponding private keys) can be accomplished using several options, including Bitcoin wallets and online/offline Bitcoin address generators. After generating the n key pairs, o_1 keeps the private keys secret and secure, and publishes the list of pairs of Bitcoin addresses and corresponding roles using chosen media (e.g., Website, database, etc.) to make them available to the public. We write $o_1.BPK_i$,

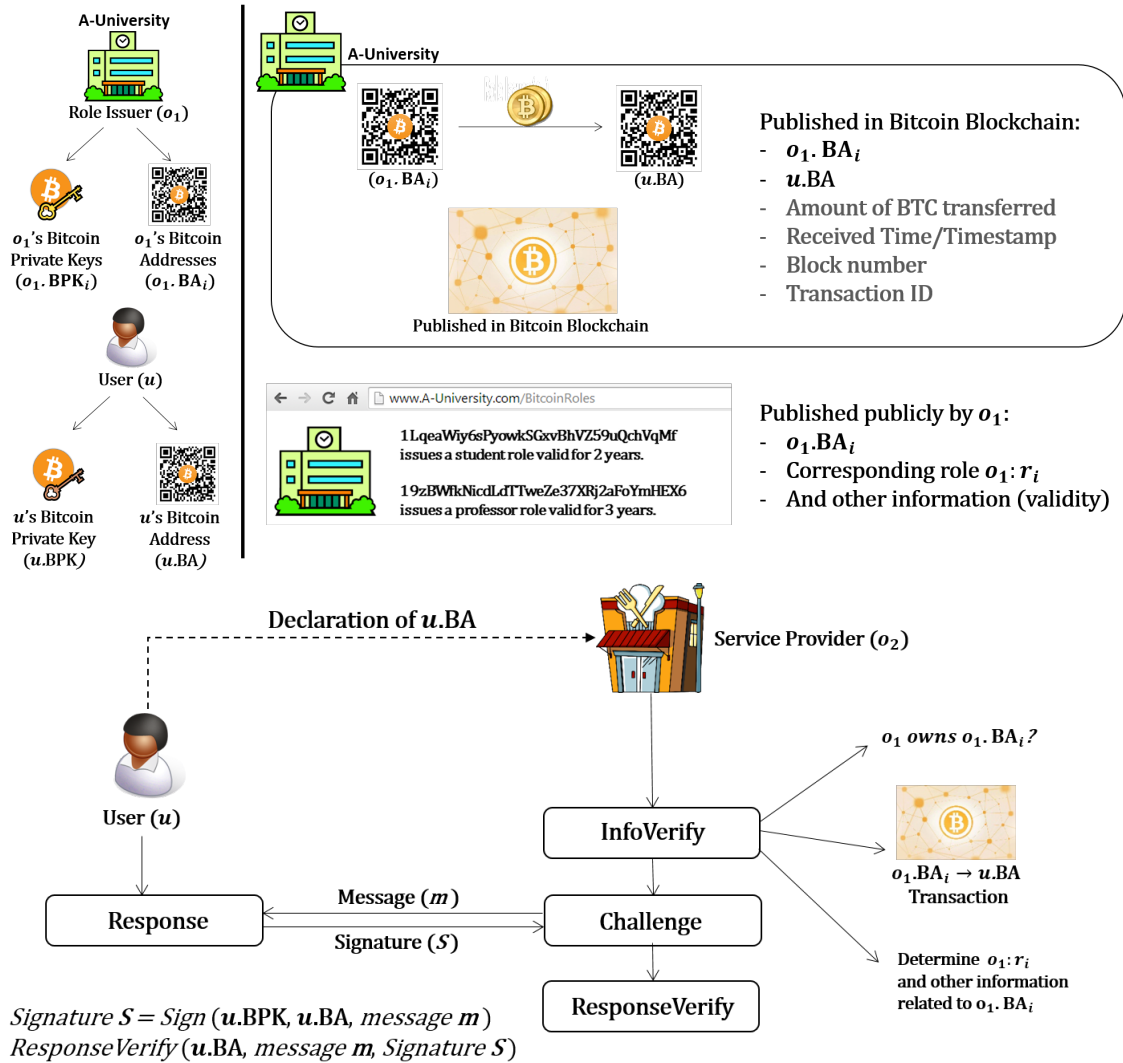


Figure 3.1. Overview of the proposed RBAC using Bitcoin.

$o_1.BA_i$, $o_1:r_i$ for the private key, Bitcoin address, and the role that is associated with the address $o_1.BA_i$, respectively, where $1 \leq i \leq n$. The publication of these Bitcoin addresses will serve as proof that o_1 owns and manages the addresses (it should be noted that o_1 will not gain any benefit from publishing Bitcoin addresses that it does not own, and thus, it is safe to assume that o_1 will only publish Bitcoin addresses it owns).

Similarly, a user (u) generates a pair of a private key $u.BPK$ and a Bitcoin address $u.BA$. Alternatively, o_1 can generate the $(u.BPK, u.BA)$ key pair and send it to u through a secure communication channel. Note however, that it is recommended by the Bitcoin community that only the one who created the key pair should be in possession of the key pair because the private key is used for accessing the bitcoins stored in the corresponding address.

Creating the role-issuer and user connection

The organization o_1 creates a simple Bitcoin transaction using $o_1.BA_i$ as input address and $u.BA$ as output address. In this transaction, o_1 sends an arbitrary amount of bitcoins to u . In the proposed system, the role-issuing organizations are expected to transfer the minimum amount of bitcoins, which is equal to 1 satoshi or 0.00000001 BTC. This is a dust transaction and will incur a required transaction fee of 0.0001 BTC. Therefore, the current minimum amount of BTC that can be used for a transaction to be considered valid is 0.00010001 BTC. Optionally, o_1 can include a higher transaction fee or miners' fee if it wants its transaction to be prioritized in the current round of solving for the block (but for our purposes, the time of confirmation is not vital and the minimum transaction fee is sufficient). After confirming the details of the transaction, o_1 sends the transaction to the Bitcoin network awaiting for confirmations from miners that the transaction is permanently included in a block in the blockchain. Once included in the blockchain, certain details will be publicly available, including $o_1.BA_i$, $u.BA$, amount of bitcoins transferred, transaction ID, block number, received time, and the time it was included in the block.

Verifying a user-role assignment

Assume that user u visits a service-providing organization o_2 and asserts that he/she has the role of $o_1:r_i$ that was issued by o_1 . The organization o_2 inquires u for the Bitcoin address, say $u.BA$, that was granted the asserted role of $o_1:r_i$ from o_1 . Then o_2 will (i) determine the Bitcoin address $o_1.BA_i$ that is associated with the role $o_1:r_i$, (ii) confirm the existence of the transaction from $o_1.BA_i$ to $u.BA$, and (iii) verify if u is the genuine owner of $u.BA$. The Bitcoin address $o_1.BA_i$ can be found in the medium where o_1 published the Bitcoin addresses it owns. The confirmation of the transaction can be done by using a blockchain browser or a program similar to that. Steps (i) and (ii) assure o_2 that the role $o_1:r_i$ and other related information associated with $o_1.BA_i$ are assigned by o_1 to the owner of $u.BA$. The ownership of $u.BA$ is verified by a challenge-response protocol where ECDSA keys that are associated with the Bitcoin address $u.BA$ are utilized.

Challenge-Response Protocol

The organization o_2 chooses an arbitrary data m and requests u to sign it, together with $u.BA$, using the private key $u.BPK$. The signature is defined by $S = \text{Sign}(u.BPK, u.BA, m)$, and thus a correct S will only be created if u has $u.BPK$. User u then sends the signature back to o_2 and o_2 will verify using the function $\text{ResponseVerify}(u.BA, m, S)$. Examples of signing/verifying a message to prove ownership of a Bitcoin address are shown in 3.2.

Remark that o_2 can confirm if u has access to the role $o_1:r_i$ without querying o_1 , and that u has little chance to disguise his/her role.


Comparison of Practicality and Costs

The proposed system provides flexibility for parties who want to join or leave the system anytime (in this subsection, parties refer to role-issuing organizations, service-providing organizations, and users, unless otherwise specified). New parties only need to follow the Bitcoin protocol to participate in the system, while role-issuing organizations who want to leave the system can simply retract or disable the media where they published the Bitcoin addresses they own. The MA-ABE [56] and MA-ABS [57] systems provide the same flexibility as the pro-

Sign Message ?

Address: 1NatgSpg6RcM2W9mGG51twqJNxTqfFGVrw

Message: Challenging User for u.BA

Signature:  HB3wJdStBZrtGvu4+r5zSc8tCps7fYlmXyCUenINua27YFXe7
tTqsDuAyzeKOT7QLc1llmAPtuEV6tZ+81n9M84=

Reset Form

Done

Verify Message ?

Address: 1NatgSpg6RcM2W9mGG51twqJNxTqfFGVrw

Message: Challenging User for u.BA

Signature: HB3wJdStBZrtGvu4+r5zSc8tCps7fYlmXyCUenINua27YFXe7tTasD
uAyzeKOT7QLc1llmAPtuEV6tZ+81n9M84=

Verified 

Done

Figure 3.2. The signing (top) and verifying (bottom) of a message features of a Bitcoin wallet.

posed scheme in terms of parties leaving or joining because they are decentralized, but PKI-based and server-based systems are not as flexible because the issuance and acquisition of certificates are not free of charge and because parties need permission from the central authority to participate.

In the proposed, MA-ABE, and MA-ABS systems, the responsibility of the authentication activities belongs to the parties, while in the PKI-based and server-based systems, the responsibility belongs to the central, single authority. Decentralized and centralized systems have different ways of handling their activities and cannot be compared directly, although it can be argued that decentralized systems are more flexible than centralized systems because centralized systems break down completely if the central authority or server becomes corrupted.

The proposed system is relatively easy to implement given that it adopts the Bitcoin network, which is completely online and is run by a P2P network. Bitcoin has already established a currently secure, always available, and complete system that uses common parameters that have been set and agreed upon by the Bitcoin community based on significant amount of research. It is also noted that the setup cost of the proposed system is reduced to a minimum because the Bitcoin network has already been established and is currently functioning. The MA-ABE [56] and MA-ABS [57] are relatively new technologies, and they can possibly be widely accepted in the future. However, as of writing this manuscript, the practical contribution of these techniques to existing network systems and the cost issues of their deployment are yet to be discussed. PKI-based and server-based systems are well-recognized frameworks and have been utilized in many systems. However, it is often pointed out that, in these frameworks, the setup and service/labor costs become more expensive as the number of users increases. For example, a 5,000-user PKI infrastructure can cost approximately 200,000 USD annually, with the initial setup cost amounting to 10,000 USD [58, 59]. These costs can also apply for other systems that rely on a central authority, including Identity Providers (IdPs) which authenticate users on the Internet by using security tokens.

To avoid confusion, we remark that the computational cost and the participation to the P2P Bitcoin network are not essential for the parties participating in the proposed scheme. Heavy computations are only necessary during the process of mining bitcoins, but the parties do not need to mine bitcoins to participate

in the system. Parties only need to generate Bitcoin addresses, create Bitcoin transactions for role issuance, and perform the challenge-response protocol, all of which have reasonably small computational costs and thus can easily be applied in today’s computer systems and devices.

3.3. Role Management

In the proposed framework, the relation between users and roles is represented by the users’ possession of the private keys that prove they own the corresponding Bitcoin addresses. This approach involves a possible security risks— the leakage, theft, and loss of keys.

3.3.1 Personalization of roles

If a user leaks his/her private key, then the people who happen to know the key can also prove ownership of the corresponding Bitcoin address (which in turn can be used to prove that a role associated with the address was assigned to them), as shown in Figure 3.3. Note that the intended user can still prove ownership of the Bitcoin address even after leakage, although such an inappropriate usage of keys can obstruct fair and reliable access control.

To deter such irresponsible behavior of users, the proposed system offers four possible measures:

1. Given that the proposed system uses Bitcoin technology, it inherently has a “traitor tracing” capability because the Bitcoin addresses are unique (thus, they can be possibly mapped to users) and the transactions are published publicly. Thus, if a user receives a leaked key and maliciously uses the role associated to the corresponding Bitcoin address, a consequent investigation can possibly lead to the original user associated with the Bitcoin address.
2. Role-issuing organizations can “personalize” roles by including some unique identifiers (which can be encrypted as well) to the data it will publish publicly. For example, A-university can publish “ o_1 .BA₁ issues a ‘student’ role to student #123 with 6 months validity.”

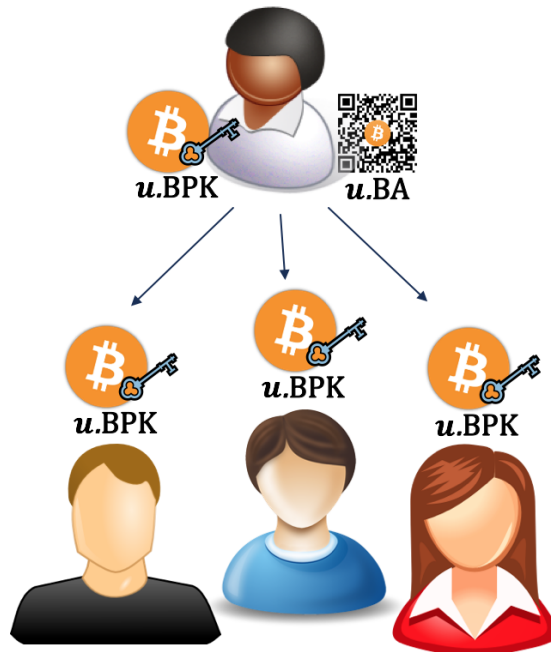


Figure 3.3. A user leaking the private key to other users.

3. Role-issuing organizations can “personalize” roles by making use of the OP_RETURN part of the Bitcoin protocol. The OP_RETURN script is used to mark a transaction output as invalid, i.e., the bitcoins attached to outputs with OP_RETURN are unspendable [60]. Therefore, transactions with modified OP_RETURN are used for other purposes, such as notarization and proof-of-existence of documents, by attaching arbitrary data to it. The OP_RETURN can handle up to 80 bytes of user-defined data. The OP_RETURN script has been increased to 80 bytes from 40 bytes because the Bitcoin blockchain is being used for other purposes that create a lot of dust transaction, consequently bloating the blockchain. The developers hope that the increase in the size of OP_RETURN outputs can be used to burn bitcoins and minimize the creation of dust transactions.
4. Role-issuing organizations can “personalize” roles by making use of a public note that is included in the Bitcoin transaction, as shown in Figure 3.4 . It should be noted that this public note is a relatively new feature offered by an online wallet (blockchain.info) [61] and is not part of the Bitcoin

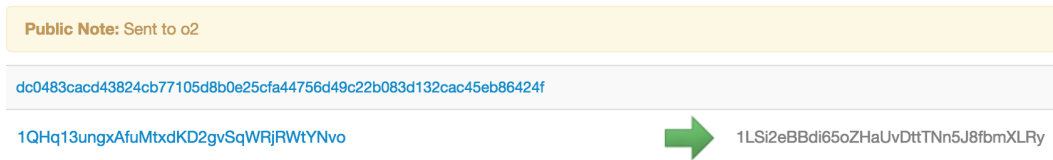


Figure 3.4. A public note attached to a transaction.

protocol.

With these measures, a user will be more conscious of leaking/losing his/her key to another person because he/she will have the risk of being identified and subsequently punished for irresponsible behavior. The theft/loss of keys remains a risk, but such risk also exists for ID-cards used in the real world.

3.3.2 Role Re-issuance

If the private keys are lost or forgotten, or if access to the digital wallet is lost or forgotten, then control over the corresponding Bitcoin addresses is also lost. The ownership of the Bitcoin addresses cannot be verified or proven without the corresponding private keys.

In this case, the role-issuing organizations can easily re-issue the roles by creating another Bitcoin transaction to the new Bitcoin address of the compromised user. The overhead of role re-issuance is relatively low for both the role-issuing organizations and the user.

Moreover, to make sure that the compromised Bitcoin addresses will not be used maliciously, a role-issuing organization can create a revocation list containing these addresses in the media where it publishes the Bitcoin addresses it owns.

3.3.3 Additional Security Measures

Bitcoin wallets are the most common target of attacks, but of course, security measures have been implemented and are recommended to minimize such cases. In the proposed system, the purpose of the Bitcoin transaction is to connect the user to an organization and to a role. If the user is a Bitcoin user, he/she

is recommended to use other wallets or other addresses to store bitcoins. Ultimately, the user only needs to store the private key safely, and even keep it offline. The challenge-response protocol can be performed offline. Moreover, an attacker with no prior knowledge of the proposed system and the role associated with the address will have no motives or incentives to steal the private key (even if an attacker can steal the private keys, the addresses will not contain any large amounts of bitcoins to steal).

3.3.4 Endorsement

The Bitcoin network provides a natural connection between Bitcoin addresses published in the blockchain. This function can be used to realize some personal activities that are not considered in the conventional RBAC approach. One possible example is the endorsement of another person. In the real world, an endorsement among individuals sometimes plays an important role. Semi-closed organizations, such as academic societies and golf clubs, have the tradition or policy that a newcomer must be endorsed or referred by a current member. This mechanism can be realized by extending the proposed system.

Figure 3.5 shows an idea of the endorsement mechanism. Consider for example that Alice (u) is an authorized member of XYZ golf club (o_1). This relationship is realized by the Bitcoin transaction from $o_1.BA$, to $u.BA$. If Alice would like to endorse Bob (u_2) to o_1 , then she can similarly create a Bitcoin transaction from $u.BA$ to $u_2.BA$, linking their addresses. Then, Bob can go to o_1 and declare $u_2.BA$. The club can look up the blockchain and check that $u_2.BA$ was endorsed by $u.BA$, which was originally endorsed by the club, as represented by $o_1.BA$. Once the connection is established, o_1 can verify if u_2 is the owner of $u_2.BA$ by using the challenge-response protocol. By querying the blockchain and through the challenge-response authentication, the club does not have to inquire Alice for the verification of the endorsement.

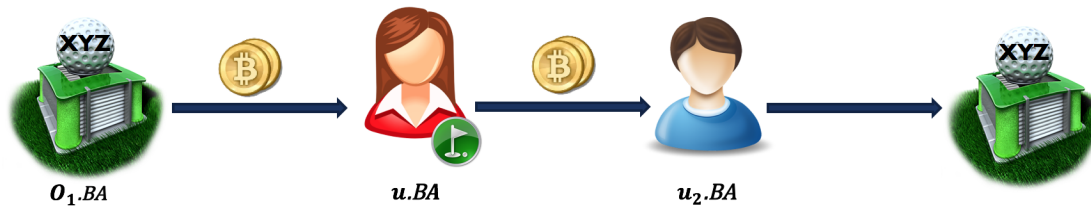


Figure 3.5. Example case where a user endorses another user.

3.3.5 Trusted Timestamping as Proof of Validity or Expiration Dates

Trusted timestamping is the process of securely creating a proof, i.e., timestamp, of the creation or modification time of a document. It is used for proving that certain information or document existed at some point in time and has not been tampered or modified since.

Traditional timestamping processes follow the RFC 3161 standard, wherein the timestamp is issued by a trusted third party acting as a Time Stamping Authority (TSA) [62].

The Bitcoin network features a timestamp server used in the blockchain to link the blocks together in a chronological manner. This timestamp server has been used, outside the main purpose of Bitcoin, as a trusted timestamping mechanism for digital documents given that it is secure (extremely difficult to attack and modify), robust (DoS resistant), and a trustworthy source of time (i.e., the time a transaction is included in the blockchain) [63, 64]. Put simply, a hash of the data that a user wants to timestamp is converted into a Bitcoin address. The timestamping service (or the user him/herself) then creates a Bitcoin transaction and makes a small payment to the converted Bitcoin address. This transaction is then stored in the public blockchain. Anyone who wants to verify the point in time a data (i.e., the hash of the data) existed can be connected to the time the transaction concerning the corresponding converted Bitcoin address was included in the blockchain.

This timestamping scheme is innovative and provides additional features as compared to the traditional trusted timestamps issued by TSAs, which are prone

to data corruption and tampering. This timestamping scheme is decentralized and is not controlled by a single authority; it can easily be accessed over the Internet and is always available; it is anonymous (one's identity, the data being given a timestamp, or even the fact that one wants something to be given a timestamp are kept secret); it is relatively low cost; the timestamp is accurate; and it is secure.

The timestamp server of Bitcoin provides a natural solution to the inclusion of expiration dates or validity of the roles in the proposed system. A role-issuing organization can include expiration dates or validity of the roles it manages in the information it publishes publicly. In this way, a service-providing organization can verify the validity of a role simply by investigating the timestamp of the block where the transaction was included in the blockchain and comparing it with the details published by the role-issuing organization.

Chapter 4

E-Voting

In this section, we first discuss some widely recognized e-voting protocols, which are the bases of other more complex secret voting schemes that have been proposed by other researchers. Many e-voting systems are based on the blind signature scheme. In this kind of scheme, voters obtain a token or signature, which is created by an authority “blindly” for a data (the vote) that is known only to the voter. The blind signature scheme is simple to understand and is easily adaptable in complicated schemes. However, a common problem with this kind of scheme is its potential for single points of failure because the authorities involved are given too much power, e.g., authorities can introduce dummy votes for voters who abstained from voting. To address this problem, variations of the blind signature have been created. For example, a threshold blind signature scheme (or simply requiring blind signatures from multiple authorities) reduces the power of authorities by creating replicates of these authorities and keeps the protocol from failing as long as a certain number of the replicates are honest. However, this kind of threshold scheme is vulnerable to the problem of colluding voters. To solve the problem of colluding voters, a public bulletin board, which is accessed through anonymous channels, has been used on top of systems based on threshold blind signature schemes to provide transparency in the voting process.

4.1. Blind Signature Scheme

A blind signature, as presented by Chaum [5], is a cryptography that is a digital version of using carbon paper-lined envelopes. In this analogy, the signature written outside the envelope leaves a carbon copy of the signature on the document inside the envelope, which the signer cannot see. In e-voting systems based on the blind signature protocol, the signer (typically an authority) signs an unknown message (blinded message) for a known requester (typically a voter). This blinding process is necessary because the signer should not know the vote of the voter. In this scheme, a data or message to be signed by a signer (authority) is disguised (randomized) first by a provider (voter) using the following blinding function:

$$m' = \text{blind}_e(m, r),$$

where m' is the blinded message, e is the public key of the signer, m is the message, and r is a random number. In an RSA blind signature scheme, the blinding function is defined as $\text{blind}_e(m, r) = mr^e \bmod n$, for example. The provider sends m' to the signer, then the signer signs m' to generate the following:

$$s' = \text{sign}_d(m')$$

where s' is the blind signature and d is the private key of the signer. Then, the signer returns s' to the provider. The provider obtains the digital signature s for m using a corresponding unblinding function given by:

$$s = \text{unblind}(s', r).$$

In the RSA blind signature, $s' = (mr^e)^d = m^d r \bmod n$, and $\text{unblind}(s', r) = r^{-1}s' = m^d \bmod n$, as shown in Figure 4.1.

4.2. Secret Voting for Large Scale Elections

Fujioka et al. [17] proposed a voting scheme for large scale elections. Their model consists of three entity types: voters V_i ($i = 1, 2, \dots, n$), an administrator A , and a counter C . In this model, V_i and C are assumed to communicate through an anonymous communication channel, which is a virtually assumed

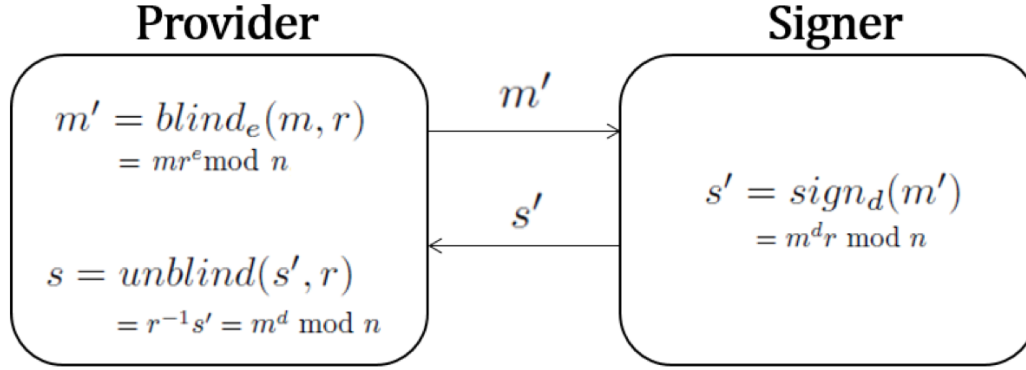


Figure 4.1. Representation of the blind signature protocol using RSA.

special channel where nobody can identify the sender of transmitted data. The model also requires the bit-commitment, digital signature, and blind signatures schemes. Voters V_i use their own digital signature scheme, A uses its own blind signature scheme and is responsible for verifying the eligibility of voters, and C only collects the ballots and publishes the results. An informal description of their scheme is as follows:

In the Preparation stage, voter V_i selects a vote v_i and completes the commitment $x_i = enc(v_i, k)$ using the bit-commitment scheme, where k is a randomly chosen key. Then, V_i computes the blinded message $x_i' = blind_e(x_i, r)$, where e is the public key of A and r is a random blinding factor. Then, V_i signs x_i' to generate $s_i = sign_{V_i}(x_i')$, where $sign_{V_i}(x_i')$ is V_i 's signature scheme (hence $sign_{V_i}$ is the signature of V_i itself for the blinded messages), and sends this s_i , together with his/her identification and x_i' , to A .

In the Administration stage, administrator A verifies the eligibility of V_i . If V_i is eligible to vote, then A signs x_i' to generate a digital signature $d_i = sign_A(x_i')$ for x_i , where $sign_A(x_i')$ is A 's signature scheme. Then, A sends d_i back to V_i . At the end of this stage, A publishes a list that contains the identities, blinded message, and signed blinded message (i.e., ID_i , x_i' , and d_i) of the voters that received d_i .

In the Voting stage, voter V_i retrieves the signature $y_i = unblind(d_i, r)$ for the commitment x_i . Voter V_i verifies the correctness of y_i , and if it succeeds, sends x_i and y_i to the counter C through an anonymous channel.

In the Collecting stage, counter C verifies that y_i is the signature for x_i . If the test succeeds, C publishes a list of (l, x_i, y_i) , where l is the entry number of the corresponding x_i and y_i .

In the Opening stage, voter V_i checks that the number of ballots that C published is equal to the number of voters that A published, and that his/her vote is listed in the list that C published. If the check succeeds, V_i sends the key k (that was used to make the commitment x_i) with the corresponding number l to C through an anonymous channel.

In the Counting stage, counter C opens the l -th ballot using the corresponding k to retrieve the vote v_i . Finally, C counts the votes and announces the results.

As pointed out in [65], this scheme has the potential for a single point of failure, wherein the authority can provide votes for the voters who did not cast their votes.

4.3. Blind Multisignatures

Multisignature schemes are used to avoid single points of failure wherein any subgroup of a group of players jointly sign a document to convince a verifier that each member of the subgroup participated in signing [66]. A multisignature scheme can be combined with a blind signature scheme to create a blind multisignature scheme that can be applied to e-voting systems. In this kind of system, a user runs the blind signature protocol with each signer (the signers are considered to be a subgroup of a group of signers) to obtain signatures for a message, each generated using the corresponding signer's private key. An entity is replicated N times and it is assumed that at least t replicates are kept from failing even in the most pessimistic scenarios. The number of t should be greater than 1 and less than N . The value of t should be selected in such a way that it is unlikely for $N - t$ or more replicates to collude or fail simultaneously (i.e., it is likely that t replicates stay safe and secure).

A general e-voting system based on blind multisignatures consists of five entity types: voters V_i ($i = 1, 2, \dots, n$), an administrator A , registration authorities R_j ($j = 1, 2, \dots, N$), a key authority K , and a counter C . The stages are as

follows:

In the Initialization stage, administrator A distributes the set of identities of legitimate voters together with their corresponding public keys to concerned entities, i.e., the registration authorities R_j with $j = 1, 2, \dots, N$ and key authority K . In this scheme, K generates the private/public keypair that will be used for the encryption process during the voting process. Note that K can also be replicated to dissolve its power, but for simplicity, only one K is discussed here.

In the Preparation stage, voter V_i selects a vote v_i and completes the commitment $x_i = enc(v_i, k)$, where k is the public key provided by K .

In the Administration/Registration stage, voter V_i randomly selects t registration authorities. In this explanation, without loss of generality, let R_1, \dots, R_t be the authorities selected by V_i . Then, V_i computes the blinded message $x_{i,j}' = blind_{e_j}(x_i, r)$ for $j = 1, \dots, t$, where r is a random blinding factor and e_j the public key of the corresponding R_j . Note that $x_{i,j}'$ is computed separately for different R_j because it is generated with the e_j of the corresponding R_j . Then, V_i requests a signature for each $x_{i,j}'$ from R_1, \dots, R_t . Then, R_j verifies the eligibility of V_i . If V_i is eligible, R_j signs $x_{i,j}'$ to generate $d_{i,j} = sign_{R_j} x_{i,j}'$. Then, R_j sends $d_{i,j}$ back to V_i . Then, V_i retrieves t signatures $y_{i,j} = unblind(d_{i,j}, r)$ for the commitment x_i . If V_i retrieves the required t signatures, then V_i is ready to vote.

In the Voting stage, voter V_i sends x_i and the t signatures $y_{i,j}$ to C through an anonymous channel, then C verifies that the required number of t signatures has been satisfied.

In the Opening/Counting stage, key authority K opens the collected commitment x_i publicly using the private key. Then, C counts the votes and announces the results.

In this kind of scheme, K holds the private key for the decryption of x_i , and thus, has the power to reveal the votes as soon as it receives x_i ; this can violate the fairness property. Moreover, the different R_j are assumed to not collaborate with each other and that they communicate with V_i independently. Therefore, colluding eligible voters can introduce extra vote/s using the extra signatures obtained beyond t .

As pointed out in [65], a colluding group of voters can introduce extra votes as follows:

$$extra\ votes = \left\lfloor \frac{N-t}{t} V_{col} \right\rfloor \quad (4.1)$$

where V_{col} is the number of colluding voters. Thus, given $N = 4$ registration authorities and $t = 3$, three colluding voters, $V_{col} = 3$, can generate 1 extra vote.

4.4. Public Registration Boards and E-Voting

Koenig et al. [65] pointed out that a public registration board is required to avoid the problem of colluding voters in e-voting systems based on threshold blind signatures. In this system, they use a public board as a knowledge base for synchronization among the registration authorities. A public board is an append-only broadcast channel with memory or storage device. Data published on the board can be read but cannot be modified.

In their protocol, the voters need not communicate with the registration authorities directly, and vice versa. Following the stages in the system discussed in the previous section, they made changes in the Registration and Voting stages. In the Registration stage, voters broadcast a blinded hash of the commitment anonymously to the public registration board. Then, the R_j check the board entries, get the blinded messages, sign the messages, and then broadcast the corresponding blind signatures back to the board. In the Voting stage, voters send the commitment, the hash of the commitment, and the signatures of the R_j anonymously to the counter C . All other stages remain the same. Figure 4.2 shows an overview of an e-voting system that makes use of a public registration board.

This kind of scheme solves the problems of colluding voters and single points of failure. However, by introducing the public registration board, it can be prone to some additional problems, such as denial of service attack and anonymity issues. A natural improvement to this kind of scheme is to create a collective of public boards or a distributed web bulletin board [67]. This collective is based on N peers of identical public boards, which, ideally, have a synchronized history of the

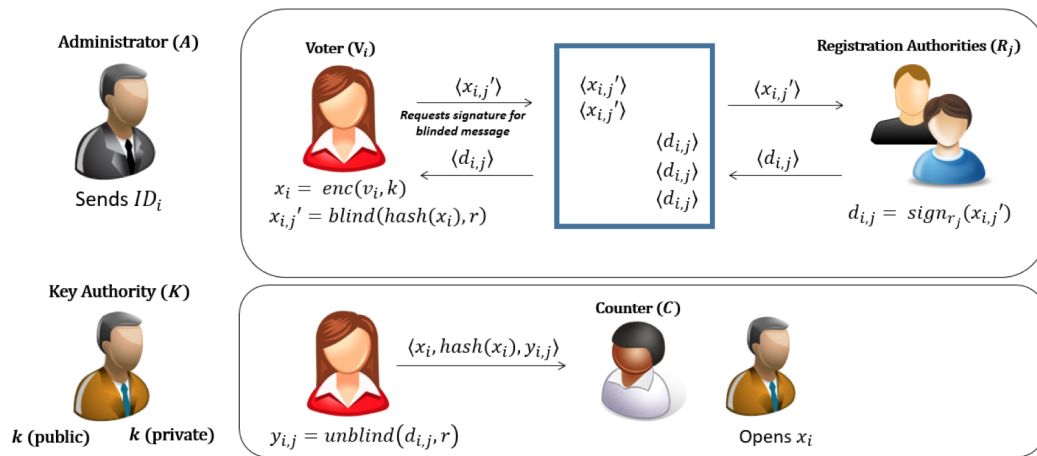


Figure 4.2. An e-voting scheme that uses a public bulletin board system.

entries. Furthermore, the anonymous nature of the bulletin board is still needed, and this approach does not mitigate the practicality issue of previous schemes.

4.5. Using the Bitcoin Blockchain for secure, independently verifiable, electronic votes

Noizat [68] uploaded a whitepaper that describes a system that leverages the Bitcoin blockchain as a secure transaction database for logging votes and auditing results. He created a demo voting application which can be found in [69]. His system revolves around a centralized voting application server, which is a point of failure. Consequently, the privacy and anonymity of the voters are violated because the application knows the votes of the voters. Moreover, the application knows the results of the votes as the votes come in, and can know the pattern and behavior of how voters are voting. The application is given almost all the power, such that it handles all the private keys, and it could act maliciously and dishonestly by introducing dummy votes and it can even change the votes.

4.6. Proposed E-Voting Protocol

The proposed system is an unconventional e-voting system based on the Bitcoin Protocol and Blind Signature Protocol. The idea is to use the Bitcoin blockchain as an alternative to an anonymous bulletin board or public registration board system, as shown in 4.3. The proposed system involves three entity types: voters V_i ($i = 1, \dots, n$), an administrator A , and a counter C .

4.6.1 Initialization and Preparation

Administrator A initiates the voting process by publishing empty ballots. Voter V_i selects a vote v_i , completes the ballot, and creates the commitment $x_i = enc(v_i, k)$, where k is a randomly chosen key. Then, V_i generates the blinded message $x_i' = blind_e(x_i, r)$, where r is a random blinding factor and e is the public key of A .

4.6.2 Administration

This stage is performed in face-to-face communication. Voter V_i requests a signature from A for x_i' . Administrator A checks and verifies if V_i is an eligible voter and has the right to vote and if V_i has not yet requested for a signature. If both conditions are met, A generates the signature $d_i = sign_A(x_i')$, where $sign_A(x_i')$ is A 's signature scheme, and then A sends d_i back to V_i . At the end of this stage, when all voters have requested the signature from A , A publishes a list that contains the identities of all the voters who received the signature from A and their corresponding commitment given by $\langle ID_i, x_i' \rangle$.

Voter V_i , who now holds the signature d_i , will retrieve the signature $y_i = unblind(d_i, r)$ for the commitment x_i . Voter V_i verifies if y_i is A 's signature for the commitment x_i . If the verification fails, V_i can claim the invalid signature by showing that $\langle x_i, y_i \rangle$ is invalid.

If PBCs will be issued by A , then V_i is given one of these PBCs (which can be put inside an envelope to ensure that it cannot be traced back to V_i). The voter V_i can check the blockchain to ensure that the Bitcoin address included in the PBC contains bitcoins.

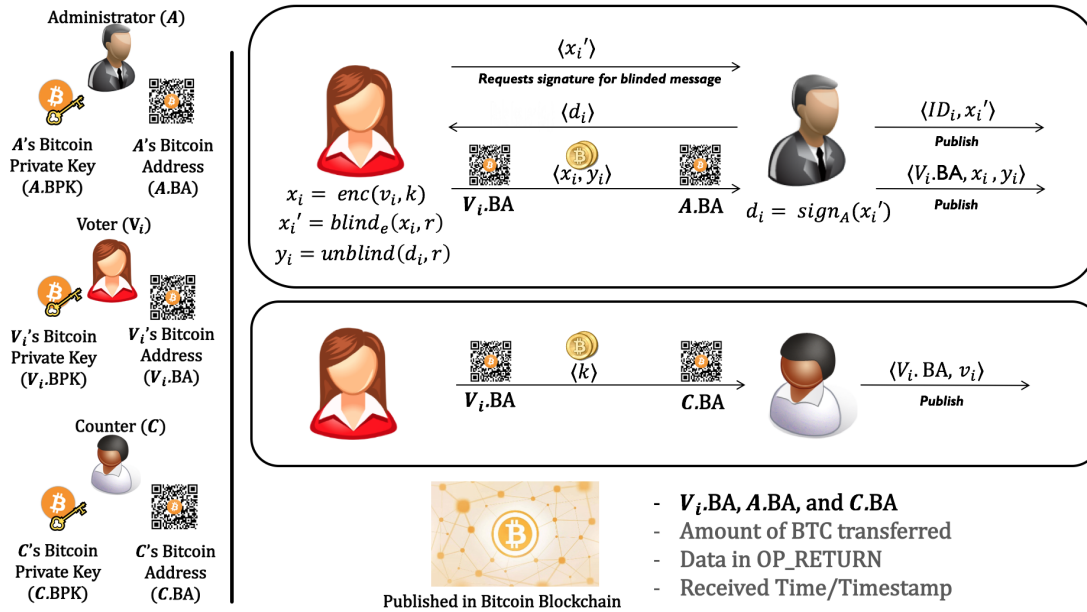


Figure 4.3. Overview of the proposed structure.

Then, privately, (i.e., at home or some secure place) V_i scratches off the covered private key, allowing him/her to have access to the corresponding Bitcoin address in the PBC. The voter V_i generates a pair of a private key V_i .BPK and a Bitcoin address V_i .BA that will be used for voting. To ensure V_i 's privacy and anonymity, V_i transfers the bitcoins from the Bitcoin address in the PBC to V_i .BA. This transaction is performed solely by V_i , and thus, no one, aside from V_i , can link the identity of the owner of V_i .BA to the Bitcoin address included in the PBC. The use of PBCs also introduces an advantage to the security and reliability of the voting process, wherein A publishes all Bitcoin addresses used in the PBCs, and only the Bitcoin addresses that are linked to these PBCs can be used in the voting process.

4.6.3 Bitcoin Address Registration

This stage is an extension of the Administration stage wherein the eligible voters who received a signature from A will register the Bitcoin Addresses that they

will use for anonymous e-voting. From the start of the process, A has generated a pair of a private key $A.BPK$ and a Bitcoin address $A.BA$. $A.BA$ is published publicly.

The voter V_i creates a simple Bitcoin transaction using $V_i.BA$ as input address and $A.BA$ as output address. In this transaction, V_i includes $\langle x_i, y_i \rangle$ in the OP_RETURN part of the protocol as proof that $V_i.BA$ is owned by a legitimate voter, but the identity of the voter is not revealed. In this transaction, V_i sends an arbitrary amount of bitcoins to A ; currently the minimum amount that can be used for a transaction to be considered valid is 0.00010001 BTC (1 satoshi = 0.00000001 BTC plus 0.0001 BTC required transaction fee). Optionally, V_i can include a higher transaction fee or miners' fee if he/she wants the transaction to be prioritized in the current round of solving for the block (but for our purposes, if the time of confirmation is not vital, the minimum transaction fee is sufficient). After confirming the details of the transaction, V_i sends the transaction to the Bitcoin network awaiting for confirmations from miners that it is permanently included in a block in the blockchain.

Once included in the blockchain, certain details will be publicly available, including $V_i.BA$, $A.BA$, the amount of bitcoins transferred, the commitment and signature $\langle x_i, y_i \rangle$ in the OP_RETURN, transaction ID, block number, received time, and the time it was included in the block. Administrator A can verify if the signature y_i of the commitment x_i is valid using its verification key. If the validation is successful, A publishes a list that contains all of the Bitcoin addresses that sent the correct signature y_i of the commitment x_i given by $\langle V_i.BA, x_i, y_i \rangle$. At the end of this stage, the number of entries in the list that contains $\langle ID_i, x_i' \rangle$ should be equal to the number of entries in the list that contains $\langle V_i.BA, x_i, y_i \rangle$.

4.6.4 Voting

Since x_i contains the vote v_i , C can just check and collect the list that contains $\langle V_i.BA, x_i, y_i \rangle$. Counter C can double-check and verify the content of the list by looking up the transactions in the blockchain using a blockchain browser or a program similar to that.

4.6.5 Opening and Counting

Voter V_i creates a simple Bitcoin transaction using V_i .BA as input address and C .BA as output address. In this transaction, V_i includes $\langle k \rangle$ in the OP_RETURN. Then, C opens the commitment x_i using the key k to retrieve v_i . Finally, C counts the votes and announces the results by publishing a list $\langle V_i$.BA, $v_i \rangle$.

4.7. Security Analysis

The security of the proposed system is analyzed based on the following properties that make e-voting systems secure:

Completeness: If all entities are honest, the results of the voting can be trusted.

Robustness/Soundness: The entities hold their own private keys, thus no other entity can perform a function or transaction on their behalf. Possible cases that can disrupt the voting process are as follows:

During the Initialization and Preparation stage, a voter V_i can keep sending invalid ballots, either with an invalid vote v_i or commitment x_i . However, this can be detected in the Counting stage, and the invalid vote will not be counted. Moreover, V_i can receive only one signature y_i from A for one commitment x_i , and x_i (hence the contents of the ballot) cannot be changed.

During the Counting stage, if V_i sends an illegal key k that cannot open x_i and obtain v_i , the fault can only come from a dishonest V_i because all Bitcoin transactions are published publicly and only V_i has access to the Bitcoin address used for the voting and only V_i possesses k that opens x_i .

After the Bitcoin Address Registration stage, if V_i leaks the private key V_i .BPK to others, a corrupt third-party cannot introduce a new v_i because it cannot change x_i . If V_i loses V_i .BPK before sending k to C , then V_i can use another Bitcoin address and state that he/she is the owner of the compromised Bitcoin address by providing the correct k that opens x_i to obtain v_i (This can be safely assumed to be valid because only the eligible voter possesses the correct k and x_i cannot be changed). If V_i loses V_i .BPK after sending k to C , then there is no problem as the purpose of the Bitcoin address was already fulfilled; i.e., x_i has been cast and k that opens x_i to obtain v_i has already been sent.

The Administrator A cannot introduce additional votes by using its own Bitcoin addresses and creating dummy signatures because the entries in the list $\langle V_i.BA, x_i, y_i \rangle$ will overflow and it will not match the list $\langle ID_i, x_i' \rangle$. A mismatch or overflowing of the lists can only happen because of a corrupt A . Moreover, A cannot dummy vote for a voter that did not cast a vote because it cannot reproduce the voter's commitment x_i and generate x_i' , which is published in $\langle ID_i, x_i' \rangle$.

Anonymity/Privacy: The relation between V_i 's identity (ID_i) and x_i is hidden by the blind signature scheme. The information that V_i sends through Bitcoin transactions (x_i and k) maintain V_i 's anonymity but can be considered valid because of the signature y_i , which can only be obtained by an eligible V_i . The Bitcoin address used by V_i in the voting process, i.e., $V_i.BA$, cannot be linked to the identity of V_i if proper management is taken and if V_i uses this Bitcoin address solely for the voting process. Moreover, the vote v_i remains secret until the Opening and Counting stage when V_i sends the key k that opens x_i .

Unreusability: It is assumed that the blind signature and cryptographic schemes cannot be broken. Voter V_i can be given only one signature y_i for one commitment x_i . Therefore, V_i can only register one valid $V_i.BA$ in the Bitcoin Address Registration stage. If V_i uses other Bitcoin address/es to send the same pair of x_i and y_i , this redundancy can easily be detected because all Bitcoin transactions are published publicly in the blockchain, and thus, V_i 's vote, which is connected to only one Bitcoin address, can only be counted once.

Fairness: The Opening and Counting stage is performed after the Voting stage; i.e., V_i sends k that opens x_i to obtain v_i only during the Counting stage. Moreover, the votes are encrypted and disguised using the encryption scheme, and thus, they cannot affect the voting during the Voting stage.

Eligibility: Assume that the digital signature scheme used by A cannot be broken, and thus, V_i cannot generate a correct signature y_i for x_i on his/her own. The verification of the eligibility of voters is performed in the Administration stage.

Individual verifiability: Given that Bitcoin transactions are published publicly, V_i can easily verify that v_i should be included in the counting by checking the blockchain using a blockchain browser. This also means that a corrupt A cannot exclude a Bitcoin address that sent the pair of $\langle x_i, y_i \rangle$. A corrupt C cannot

exclude v_i because k is sent publicly and is easily verifiable. A corrupt C cannot give a false tally of the results as all votes are publicly available.

Universal verifiability: The published outcome cannot be falsified because all votes are public.

Chapter 5

Conclusion and Future Work

In this study, the Bitcoin network was utilized as an infrastructure to realize a trans-organizational RBAC system and an e-voting system.

The proposed RBAC system provides a secure mechanism for verifying the user-role assignments of organizations in a trans-organizational manner. Compared to other similar approaches, the proposed scheme provides more flexibility and autonomy while maintaining security. Moreover, the proposed system allows the realization of many collaborative right managements that are common in physical communication but are difficult to implement over computer networks. These role management features include personalization of roles, role re-issuance, and endorsement of roles. Finally, the timestamping mechanism provided in the Bitcoin protocol provides a natural solution to the inclusion of expiration dates or validities in the created roles. Future research will focus on the realization and quantitative analysis of a more comprehensive hierarchical, multi-authority, and attribute-based system, which can offer additional role management features, such as role transfer and access policies in terms of Boolean formulas, and on the development of a prototype for easier use and interoperability.

The proposed e-voting system provides secure, anonymous, and transparent voting process. The Bitcoin protocol is complemented with well-known protocols of existing e-voting systems to provide an alternative for public bulletin boards and void the issue of anonymous communication channels that introduces problems in many existing schemes. Compared to other e-voting systems, the proposed system provides power and control to the voters while minimizing the trust on

other entities. Future research will focus on the development of a prototype that can demonstrate possible scalability and to make the proposed system easier to understand and use.

This thesis dissertation provides a couple of ways on how the Bitcoin network can be used outside the main purpose of Bitcoin (i.e., to provide a digital cryptocurrency platform where users can transfer bitcoins to each other) to create innovative systems. Blockchain technology, which was pioneered by Bitcoin, is a groundbreaking technology that has provided a solution to the centralization issue in the digital money field. However, it should also be noted that blockchain technology can also be used to provide solutions in other systems and applications that require security, decentralization, and anonymity, among others. It is believed that decentralization will play a key role in the next generation of the Internet world. The Bitcoin blockchain has been the inspiration of the concept called *blockchain 2.0*, which is the next-generation cryptolegder space (See Appendix C for a detailed explanation of blockchain 2.0.). In blockchain 2.0, blockchain technology can be used not only for transferring digital currency but also for registering smart contracts and assets. Smart contracts are applications that run exactly as programmed without any possibility of downtime, censorship, fraud or third party interference. For example, this technology is being used by Ethereum [70], which is a blockchain app platform that can run and execute programmable assets, also known as decentralized applications (dapps). Indeed the future of blockchain technology is bright, and the best applications of blockchain technology may not have been realized yet.

References

- [1] D. Chaum, “Untraceable Electronic Mail, Return addresses, and Digital Pseudonyms,” *Commun. ACM* 24, 2, 1981.
- [2] A. Back, “Hashcash - A Denial of Service Counter-Measure,” 2002, URL: <http://www.hashcash.org/papers/hashcash.pdf> [accessed: 2016-12-01].
- [3] B. Cohen, “Incentives Build Robustness in BitTorrent,” *Proc. of the 1st Workshop on Economics of Peer-to-Peer Systems*, 2003, pp. 68–72.
- [4] G. Kreitz and F. Niemel, “Spotify – Large Scale, Low Latency, P2P Music-on-Demand Streaming,” In: *P2P*, 2010, pp. 1–10.
- [5] D. Chaum, “Blind Signatures for Untraceable Payments,” *CRYPTO82*, 1982, pp. 199–203.
- [6] L. Law, S. Sabett, and J. Solinas, “How to Make a Mint: The Cryptography of Anonymous Electronic Cash,” *American University Law Review* 46, 4, 1996, pp. 1131–1162.
- [7] W. Dai, “B-money,” 1998, URL: <http://www.weidai.com/bmoney.txt> [accessed: 2016-12-01].
- [8] N. Szabo, “Bit Gold,” 2005, URL: <http://nakamotoinstitute.org/bit-gold/#selection-7.6-7.14> [accessed: 2016-12-01].

- [9] H. Finney, “RPOW – Reusable Proofs of Work,” 1998, URL: <http://nakamotoinstitute.org/finney/rpow/index.html> [accessed: 2016-12-01].
- [10] Bitcoin.org “Frequently Asked Questions: What is Bitcoin,” URL: <https://bitcoin.org/en/faq#what-is-bitcoin> [accessed: 2016-12-01].
- [11] S. Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System,” 2008, URL: <https://bitcoin.org/bitcoin.pdf>[accessed: 2016-12-01].
- [12] Blockchain.Info “Currency Stats,” URL: <https://blockchain.info/stats> [accessed: 2016-12-01].
- [13] Bitcoin.org “Bitcoin for Individuals,” URL: <https://bitcoin.org/en/bitcoin-for-individuals> [accessed: 2016-12-01].
- [14] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, “Role-based access control models,” *IEEE Computer*, 29, 2, 1996, pp. 38–47.
- [15] S. Farrell and R. Housley, “An Internet attribute certificate profile or authorization,” RFC 3281, 2002.
- [16] Internet2, “The Shibboleth System,” URL: <http://shibboleth.internet2.edu> [accessed: 2016-12-01].
- [17] A. Fujioka, T. Okamoto, and K. Ohta, “A Practical Secret Voting Scheme for Large Scale Elections,” *AUSCRYPT92*, 1992, pp. 244–251.
- [18] J. C. Benaloh and M. Yung, “Distributing the Power of a Government to Enhance the Privacy of Voters,” *5th ACM Symposium on Principles of Distributed Computing*, 1986, pp. 52–62.
- [19] T. Okamoto, “Receipt-free electronic voting schemes for large scale elections,” *Security Protocols*, 2005, pp. 25–35.
- [20] T. Okamoto, “An Electronic Voting Scheme,” *IFIP’96 Proceedings*, 1996, pp. 21–30.

- [21] S. Nakamoto, “Bitcoin P2P e-cash paper,” post in The Cryptography Mailing List URL: <http://www.metzdowd.com/pipermail/cryptography/2008-October/014810.html> [accessed: 2016-12-01].
- [22] Bitcoin Wiki, “Protocol Specification,” URL: https://en.bitcoin.it/wiki/Protocol_specification [accessed: 2016-12-01].
- [23] Bitcoin.org, “Bitcon Developer Guide,” URL: <https://bitcoin.org/en/developer-guide#block-chain> [accessed: 2016-12-01].
- [24] BestBitcoinExchange.io, “The Best Bitcoin Exchanges,” URL: <https://www.bestbitcoinexchange.io/> [accessed: 2016-12-01].
- [25] Coin ATM Radar, “New bitcoin ATM locations as of 21.07.2015,” URL: <https://coinatmradar.com/blog/tag/bitcoin-atm-map-2/> [accessed: 2016-12-01].
- [26] D. Johnson, A. Menezes, and S. Vanstone, “The Elliptic Curve Digital Signature Algorithm (ECDSA),” Technical report CORR 99–34, 1999, URL: <http://cs.ucsb.edu/~koc/ccs130h/notes/ecdsa-cert.pdf> [accessed: 2016-12-01].
- [27] H. Dobbertin, A. Bosselaers, and B. Preneel, “The hash function RIPEMD-160,” URL: <https://homes.esat.kuleuven.be/~bosselae/ripemd160.html> [accessed: 2016-12-01].
- [28] Bitcoin Wiki, “Technical background of version 1 Bitcoin addresses,” URL: https://en.bitcoin.it/wiki/Technical_background_of_version_1_Bitcoin_addresses [accessed: 2016-12-01].
- [29] I. DeMartino, “The Many Types and Functions of Bitcoin Wallets,” The Cointelegraph URL: <https://cointelegraph.com/news/the-many-types-and-functions-of-bitcoin-wallets> [accessed: 2016-12-01].
- [30] BitcoinWiki, “Transaction,” URL: <https://en.bitcoin.it/wiki/Transaction> [accessed: 2016-12-01].

- [31] BitcoinWiki, “Coinbase,” URL: <https://en.bitcoin.it/wiki/Coinbase> [accessed: 2016-12-01].
- [32] BitcoinWiki, “Transaction fees,” URL: https://en.bitcoin.it/wiki/Transaction_fees [accessed: 2016-12-01].
- [33] The Onion Network, “Tor,” URL: <https://www.torproject.org/> [accessed: 2016-12-01].
- [34] BitLaundry, “BitLaundry,” URL: <http://bitlaundry.appspot.com/> [accessed: 2016-12-01].
- [35] Bitcoin.org, “Protect your privacy,” URL: <https://bitcoin.org/en/protect-your-privacy> [accessed: 2016-12-01].
- [36] BitcoinWiki, “Block,” URL: <https://en.bitcoin.it/wiki/Block> [accessed: 2016-12-01].
- [37] BitcoinWiki, “Block hashing algorithm,” URL: https://en.bitcoin.it/wiki/Block_hashing_algorithm [accessed: 2016-12-01].
- [38] A. Antonopoulos, “Mastering Bitcoin: Unlocking Digital Cryptocurrencies,” O’Reilly Media, Inc., 2015, pp. 177–179.
- [39] C. Pacia, “Bitcoin Mining Explained Like Youre Five: Part 2 Mechanics,” URL: <https://chrispacia.wordpress.com/2013/09/02/bitcoin-mining-explained-like-youre-five-part-2-mechanics/> [accessed: 2016-12-01].
- [40] “Blockexplorer,” URL: <https://blockchain.info/q/getdifficulty> [accessed: 2016-12-01].
- [41] Blockchain Info, “Probability,” URL: <https://blockchain.info/q/probability> [accessed: 2016-12-01].
- [42] Bitcoin.org, “Securing your wallet,” URL: <https://bitcoin.org/en/secure-your-wallet> [accessed: 2016-12-01].
- [43] “Prepaid Bitcoin,” URL: <https://prepaidbitco.in/> [accessed: 2016-12-01].

- [44] M. Rosenfeld, “Analysis of Bitcoin Pooled Mining Reward Systems,” URL: https://bitcoil.co.il/pool_analysis.pdf [accessed: 2016-12-01].
- [45] RHorning, “Mining cartel attack,” Bitcoin Forum, 2010 URL: <https://bitcointalk.org/index.php?topic=2227.0> [accessed: 2016-12-01].
- [46] I. Eyal and E. G. Sirer, “Majority is not Enough: Bitcoin Mining is Vulnerable,” Financial Cryptography and Data Security, 2014.
- [47] N. T. Courtois and L. Bahack, “On Subversive Miner Strategies and Block Withholding Attack in Bitcoin Digital Currency,” CoRR, 2014 URL: <http://arxiv.org/pdf/1402.1718v5.pdf> [accessed: 2016-12-01].
- [48] R. S. Sandhu, D. Ferraiolo, and R. Kuhn, “The NIST Model for Role-Based Access Control: Toward A Unified Standard,” 5th ACM Workshop Role-Based Access Control, July 2000, pp. 47–63.
- [49] National Institute of Standards and Technology, “ROLE BASED ACCESS CONTROL (RBAC) AND ROLE BASED SECURITY,” February 2004 URL: <http://csrc.nist.gov/groups/SNS/rbac/> [accessed: 2016-10-01].
- [50] C. M. Ellison et al., “SPKI certificate theory,” RFC 2693, 1999.
- [51] P. Gutmann, “Simplifying public key management,” IEEE Computer, 37, 2, 2004, pp. 101–103.
- [52] J. P. Cruz and Y. Kaji, “Trans-Organizational Role-Based Access Control in Android,” INFOCOMP 2014, pp. 114–119.
- [53] D. Boneh and M. Franklin, “Identity-based encryption from the Weil pairing,” Proc. of the Advances in Cryptology (CRYPTO) 2001, pp. 213–229.
- [54] C. Gentry and A. Silverberg, “Hierarchical ID-based cryptography,” Proc. of the Advances in Cryptology (ASIACRYPT) 2002, pp. 548–566.
- [55] R. Rivest, A. Shamir, and L. Adleman, “A method for obtaining digital signatures and public-key cryptosystems,” Communications of the ACM, 21, 2, 1978, pp. 120–126.

- [56] A. B. Lewko and B. Waters, “Decentralizing Attribute-Based Encryption,” EUROCRYPT 2011, pp. 568–588.
- [57] T. Okamoto and K. Takashima, “Decentralized Attribute-Based Signatures,” Public-Key Cryptography-PKC 2013, pp. 125–142.
- [58] VeriSign, “Total Cost of Ownership for Public Key Infrastructure,” URL: http://www.verisign.com/stellent/groups/public/documents/white_paper/005321.pdf [accessed: 2016-12-01].
- [59] TC TrustCenter, “The Costs of Managed PKI,” URL: http://www.safebiopharma.org/VendorPartnerFiles/ChosenSecurity/The%20Costs%20of%20Managed%20PKI%20Whitepaper_Web%20and%20QuickStart_Feb08.pdf [accessed: 2016-12-01].
- [60] Bitcoin Wiki, “OP_RETURN,” URL: https://en.bitcoin.it/wiki/OP_RETURN [accessed: 2016-12-01].
- [61] Blockchain.info, “Blockchain Website FAQ: What are public notes?,” URL: <https://blockchain.info/wallet/website-faq> [accessed: 2016-12-01].
- [62] SSL4NET, “Trusted Timestamping,” URL: <http://www.ssl4net.com/technology/trusted-timestamping> [accessed: 2016-12-01].
- [63] J. Kluscek, “Trusted Timestamp in Cryptocurrency Block Chain,” Student EEICT, 2014.
- [64] BTProof, “Trusted timestamping on the Bitcoin blockchain,” URL: <https://www.btproof.com> [accessed: 2016-12-01].
- [65] R. Koenig, E. Dubuis, and R. Haenni, “Why Public Registration Boards are Required in E-Voting Systems Based on Threshold Blind Signature Protocols,” Electronic Voting, 2010, pp. 255–266.
- [66] A. Boldyreva, “Threshold Signatures, Multisignatures and Blind Signatures Based on the Gap-Diffie-Hellman-Group Signature Scheme,” PKC, 2003, pp. 31–46.

- [67] J. Heather and D. Lundin, “The append-only web bulletin board,” *Formal Aspects in Security and Trust*, 2009, pp. 242–256.
- [68] P. Noizat, “Using the Bitcoin Blockchain for secure, independently verifiable, electronic votes,” (whitepaper), 2014, URL: https://bitcoin.fr/public/divers/docs/Blockchain_Electronic_Votes_-_White_paper.pdf [accessed: 2016-12-01].
- [69] P. Noizat, “unchain.voting,” URL: <http://www.unchain.voting/bootstrap/index?l=en&locale=fr> [accessed: 2016-12-01].
- [70] “Ethereum Homestead Release,” URL: <https://www.ethereum.org/> [accessed: 2016-12-01].
- [71] Nomura Research Institute, “Survey on Blockchain Technologies and Related Services: FY2015 Report,” URL: http://www.meti.go.jp/english/press/2016/pdf/0531_01f.pdf [accessed: 2016-12-01].
- [72] D. J Bernstein, J. Buchmann, and E. Dahmen, “Post-Quantum Cryptography,” Springer, 2009.
- [73] D. J. Bernstein, D. Hopwood, A. Hülsing, et al., “SPHINCS: Practical Stateless Hash-Based Signatures,” *EUROCRYPT 15*, LNCS 9056, 2015, pp. 368–397.
- [74] D. Bleichenbacher and U. Maurer, “Optimal Tree-Based One-Time Digital Signature Schemes,” *Symp. on Theoretical Aspects of Computer Science*, LNCS 1046, 1996, pp. 363–374.
- [75] D. Bleichenbacher and U. Maurer, “On the Efficiency of One-Time Digital Signature Schemes,” *ASIACRYPT 96*, LNCS 1163, 1996, pp.145–158.
- [76] D. Boneh, E. Shen, and B. Waters, “Strongly Unforgeable Signatures Based on Computational Diffie-Hellman,” *Intl. Conf. on Theory and Practice of Public-Key Cryptography*, LNCS 3958, 2006, pp. 229–240.

- [77] J. Buchmann, E. Dahmen, S. Ereth, A. Hülsing, and M. Rückert, “On the Security of the Winternitz One-Time Signature Scheme,” AFRICACRYPT 11, LNCS 6737, 2011, pp. 363–378.
- [78] J. Buchmann, E. Dahmen, and A. Hülsing, “XMSS—A Practical Forward Secure Signature Scheme Based on Minimal Security Assumptions,” Intl. Conf. on Post-Quantum Cryptography, LNCS 7071, 2011, pp. 117–129.
- [79] C. Dodds, N. Smart, and M. Stam, “Hash Based Digital Signature Schemes,” Intl. Conf. on Cryptography and Coding 2005, LNCS 3796, 2005, pp. 96–115.
- [80] A. Hülsing, “W-OTS⁺ — Shorter Signatures for Hash-Based Signature Schemes,” AFRICACRYPT 13, LNCS 7918, 2013, pp.173–188.
- [81] L. Lamport, “Constructing Digital Signatures from a One-Way Function,” Technical Report SRI-CSL-98, SRI International Computer Science Laboratory, 1979.
- [82] R. Merkle, “A Certified Digital Signature,” CRYPTO 89, LNCS 435, 1990, pp. 218–238.
- [83] A. Perrig, “The BiBa One-Time Signature and Broadcast Authentication Protocol,” ACM Conf. on Computer and Communications Security, 2011, pp.28–37.
- [84] L. Reyzin and N. Reyzin, “Better than BiBa: Short One-Time Signatures with Fast Signing and Verifying,” Intl. Information Security and Privacy Conf., LNCS 2384, 2002, pp. 1–47.
- [85] P. W. Shor, “Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer,” SIAM J. Computing, **26**, 5, 1997, pp. 1484–1509.
- [86] A. De Caro and V. Iovino, “jPBC: Java pairing based cryptography”, Proc. Of the IEEE Symposium on Computers and Communications (ISCC) 2011, pp. 850–855.

- [87] R. Meier, “Professional Android 4 Application Development”, John Wiley & Sons, Inc., Indianapolis, 2012, pp. 693–700.
- [88] S. Komatineni and D. MacLean, “Pro Android 4”, Apress, 2012, pp. 858–870.
- [89] QRCode.com, “QRCode.com”, URL: <http://www.qrcode.com/en/> [accessed: 2016-12-01].
- [90] zxing, “Official ZXing (Zebra Crossing) project home”, URL: <https://github.com/zxing/zxing> [accessed: 2016-12-01].
- [91] M. Scott, N. Costigan, and W. Abdulwahab, “Implementing cryptographic pairings on smartcards”, Proc. of the Cryptographic Hardware and Embedded Systems, 2006, pp. 134–147.
- [92] BitcoinWiki, “Script,” URL: <https://en.bitcoin.it/wiki/Script> [accessed: 2016-12-01].
- [93] Wikipedia, “Turing completeness”, URL: https://en.wikipedia.org/wiki/Turing_completeness [accessed: 2016-12-01].
- [94] Blockchain.info, “Blockchain Size”, URL: <https://blockchain.info/charts/blocks-size> [accessed: 2016-12-01].
- [95] CoinMarketCap, “Crypto-Currency Market Capitalizations”, URL: <https://coinmarketcap.com/currencies/views/all/> [accessed: 2016-12-01].
- [96] Ethereum.org, “Ethereum Homestead Documentation”, URL: <http://www.ethdocs.org/en/latest/> [accessed: 2016-12-01].
- [97] Solidity, “Solidity Documentation”, URL: <https://solidity.readthedocs.io/en/develop/> [accessed: 2016-12-01].
- [98] EtherCasts, “State of the Dapps”, URL: <http://dapps.ethercasts.com/> [accessed: 2016-12-01].

Acknowledgements

My four years in Japan have been the most fun and most challenging years of my professional life. Looking back, this journey was the product of hard work, luck, fate, and destiny. Japan has been the setting of one of my greatest achievements—receiving a Doctoral Degree in Engineering. Of course, this milestone would not have been possible without the guidance, support, and help of the wonderful people I have met throughout this journey. I would like to express my sincerest gratitude to the following:

To Professor Yuichi Kaji, my supervisor and mentor, for his guidance, advice, and support. He was always available throughout my entire Ph.D. program, and he is the best supervisor anyone would be lucky to have (other students under his supervision would attest to this). He challenged me to grow as a good researcher, and he has provided opportunities for me to start my career.

To Professor Hiroyuki Seki for recruiting me and urging me to pursue my doctorate in NAIST. He was the first person to encourage me to come to Japan, and if not for him, I would not even have any idea about this amazing opportunity.

To Professor Minoru Ito and Professor Kenichi Matsumoto for accepting me in their laboratories.

To Assistant Professor Hideaki Hata for the opportunities that allowed me to continue my research on blockchain technology.

To the Japanese Government and MEXT for the financial support. I hope they would continue providing opportunities to others like me, who want to pursue their graduate studies in Japan.

To the staff of the International Affairs Section in NAIST for their help with all important documents that made my student life in Japan less difficult.

To the Filipino Community in NAIST for making life in a foreign country

easier and enjoyable. They made NAIST feel like a second home.

To my international friends in NAIST for letting me know different cultures, for the conversations in English, and for the parties.

To my Japanese friends for helping me with matters in Japanese that I do not understand.

To the basketball club for letting me enjoy the sport I love the most, and making me better at it. After all, having a good physical condition contributes to a good mental condition.

To my family– Jesse (my dad), Marcy (my mom), KC (my sister) and Arnel (my brother-in-law), Lucas (my godson), and Kevin and Alfred (twin nephews)– for their unconditional love and support. They gave me the freedom to discover the world and pursue my dreams. My success and achievements are for you.

To Chanikarn Chantarasrivong– my fiancé, my love, my life– for being by my side every step of the way and for bringing out the best in me.

To those who I forgot to mention, thank you.

Finally, to God, creator of everything. Without him, none of these would have been possible.

Appendix

A. Constant-Sum Fingerprinting for Winternitz One-Time Signature

A fingerprinting function for hash-based Winternitz one-time signature is proposed. A hash-based one-time signature is a light-weight and quantum-immune alternative to digital signature schemes that typically make use of number theoretic problems. The Winternitz scheme, which is one of the most promising hash-based one-time signatures, realizes the function of digital signatures by utilizing several hash chains. In computing a signature, a fingerprinting function is used to specify certain positions in the hash chains. This study proposes a fingerprinting function that reduces the size of a signature and the operational cost for the verification of a signature. The proposed fingerprinting function can be combined with another improvement technique of the Winternitz scheme, and contributes to the discussion that promotes practical use of the Winternitz scheme.

A.1 Introduction

Commonly used digital signature schemes often make use of number theoretic problems to achieve security. Such an approach, however, can introduce two problems. The first and now arising problem is the complexity issue; digital signature schemes require computation of quite large numbers. It is difficult, or not favorable, to implement such schemes on small battery-operated devices for which energy consumption is the greatest concern. The second and still tentative problem is the threat of quantum computers; the security of conventional digital signatures collapses if practical quantum computers are realized because they can efficiently solve number theoretic problems, including the factoring and the discrete logarithm [85].

A *hash-based digital signature* is a light-weight and quantum-immune alternative to conventional digital signature schemes. The functions of a hash-based digital signature scheme are performed using cryptographic hash functions. The scheme is lightweight because the computation of a hash function can be performed with much less time and energy than the computation of modular exponentiation. The scheme is regarded as quantum immune because a hash function

involves tricky mechanisms which make the function mathematically disordered and thus difficult to tackle even by quantum algorithms. Refer to [72] for the contribution of hash functions in the security against quantum computers.

In a typical hash-based digital signature, the secret *signing key* is a set of randomly selected hash values, and the public *verification key* is another set of hash values that are derived from the signing key. A *signature* is a set of intermediate hash values that are determined from the message to be signed. A verifier confirms the integrity of a signature by checking if the verification key that is reconstructed from the signature coincides with the verification key that has been distributed in advance. In this framework, an issuance of a signature can be regarded as a disclosure of secret, and a once used key should not be reused to ensure security. For this reason, a hash-based digital signature is sometimes called a *one-time signature (OTS)* in literature, although there is a technique that consolidates several verification keys into a single key [82].

The first hash-based OTS was proposed by Lamport [81]. Merkle improved Lamport OTS, and reduced the signature size by about half [82]. As described in [82], Merkle OTS can be generalized to Winternitz OTS that makes use of longer hash chains than Merkle OTS. Several OTS have been proposed thereafter [74, 75, 79, 83, 84], but Winternitz OTS seems most promising because it is simple and has the potential for improvements.

Modifications of Winternitz OTS are discussed in [77, 80], in which hash chains are constructed by using a family of keyed hash functions rather than a single specific hash function. These extensions of Winternitz OTS are employed in more practical investigations [78, 73].

A possible problem of Winternitz scheme is the complexity of the signature verification; the number of hash computations for signature verification can be quite large.

As described previously, one major target of OTS is the use in a computer network that consists of resource-restricted devices. In such a network, it is typical that a centralized server and network clients have different capabilities and different roles; signed messages are always constructed and issued by the sever, and clients just receive and verify the signature. In this environment, the complexity of verification is more substantial than the complexity of signing,

because the verification is performed by resource-restricted clients.

To reduce the complexity of the signature verification in Winternitz OTS, we consider to introduce a novel fingerprinting function that is called a *constant-sum* fingerprinting. A constant-sum fingerprinting function contributes to control the complexity of the signature verification, and thus makes Winternitz OTS more favorable for resource-restricted devices.

The constant-sum fingerprinting can be used over arbitrarily constructed hash chains, which means that our proposal can be combined with the investigations in [77, 80].

A.2 Winternitz OTS

A hash-based OTS consists of three algorithms KeyGen, Sign, and Verify.

- KeyGen(1^n) is a probabilistic algorithm that generates a secret *signing key* SK and a public *verification key* VK for a security parameter n . The keys must be constructed such that they allow 2^n different signatures.
- Sign(SK, m) is a probabilistic or a deterministic algorithm that computes a *signature* σ for a given message m using SK as a signing key.
- Verify(VK, m, σ) is a deterministic algorithm that checks if σ is a valid signature for m where VK is used as a verification key. The algorithm accepts m and σ if and only if $(\text{SK}, \text{VK}) \leftarrow \text{KeyGen}(1^n)$ and $\sigma \leftarrow \text{Sign}(\text{SK}, m)$.

Winternitz OTS gives implementations of the above three algorithms by using several hash chains that are constructed by a cryptographic hash function with the one-way property. In constructing Winternitz OTS, we first determine a *fingerprinting function* f that maps a message of an arbitrary length to a binary sequence of length n , where n is the security parameter that is given to KeyGen algorithm. We also determine a positive integer parameter l , and define $w = w_1 + w_2$ with

$$w_1 = \left\lceil \frac{n}{\log_2 l} \right\rceil, \quad w_2 = \left\lceil \frac{\log_2(w_1(l-1))}{\log_2 l} \right\rceil + 1.$$

Notice that an n -bit binary sequence can be regarded as an integer that is 0 or greater and $2^n - 1$ or less, and hence is written with w_1 -digits in the base- l

representation. Based on this observation, we say that (a_1, \dots, a_{w_1}) with $0 \leq a_i \leq l - 1$, $1 \leq i \leq w_1$ is an l -ary fingerprint of a message m if

$$\sum_{i=1}^{w_1} a_i l^{w_1-i} = \sum_{i=1}^n b_i 2^{n-i},$$

for $f(m) = b_1 \cdots b_n$. The *check-sum* of the l -ary fingerprint (a_1, \dots, a_{w_1}) is defined as an integer $C = w_1(l - 1) - \sum_{i=1}^{w_1} a_i$, which is 0 or greater and $w_1(l - 1)$ or less, and hence is written with w_2 -digits in the base- l representation. Let (c_1, \dots, c_{w_2}) be the base- l representation of the above check-sum C , and call $(a_1, \dots, a_{w_1}, c_1, \dots, c_{w_2})$ a *check-summed fingerprint* of the message m . Note that there is a one-to-one correspondence between fingerprints in $\{0, 1\}^n$ and consistent check-summed fingerprints, and f can be regarded as a function that directly computes a check-summed fingerprint for a given message.

Besides the fingerprinting function f , Winternitz OTS uses a one-way hash function which we denote by h . We assume that the domain and the co-domain of h are both $\{0, 1\}^L$.

The algorithms of Winternitz OTS are defined as follows.

- $\text{KeyGen}(1^n)$: The signing key SK is defined as $\text{SK} = (s_1, \dots, s_w)$, where s_i with $1 \leq i \leq w$ is uniformly selected from $\{0, 1\}^L$ at random. The verification key VK is defined as $\text{VK} = (h^{l-1}(s_1), \dots, h^{l-1}(s_w))$.
- $\text{Sign}(\text{SK}, m)$: The signature for the message m is defined as $\sigma = (h^{f_1}(s_1), \dots, f^{f_w}(s_w))$, where (f_1, \dots, f_w) is the check-summed fingerprint of m .
- $\text{Verify}(\text{VK}, m, \sigma)$: The pair of the message m and the signature $\sigma = (\sigma_1, \dots, \sigma_w)$ is accepted if and only if $(h^{l-1-f_1}(\sigma_1), \dots, h^{l-1-f_w}(\sigma_w))$ coincides with the verification key VK, where (f_1, \dots, f_w) is the check-summed fingerprint of m .

The check-sum is essential in the security of Winternitz OTS: Without the check-sum, an attacker who obtains a pair of a message and a signature is able to forge a signature for other messages with the probability that is much higher than the collision probability of the fingerprinting function.

The signing key, verification key, and signature for Winternitz OTS are all wL -bits in size because they are tuples of w hash values having L -bits each.

Table 5.1. Parameters, signature sizes and costs of Winternitz OTS

l	w_1	w_2	w	$ \sigma $	KeyGen	Sign	Verify
2	160	8	168	$168L$	168	≤ 168	≤ 168
3	101	5	106	$106L$	212	≤ 212	≤ 212
4	80	4	84	$84L$	252	≤ 252	≤ 252
5	69	4	73	$73L$	292	≤ 292	≤ 292
6	62	4	66	$66L$	330	≤ 330	≤ 330
7	57	3	60	$60L$	360	≤ 360	≤ 360
8	54	3	57	$57L$	399	≤ 399	≤ 399
16	40	3	43	$43L$	645	≤ 645	≤ 645
32	32	2	34	$34L$	1,054	$\leq 1,054$	$\leq 1,054$
64	27	2	29	$29L$	1,827	$\leq 1,827$	$\leq 1,827$
128	23	2	25	$25L$	3,175	$\leq 3,175$	$\leq 3,175$
256	20	2	22	$22L$	5,610	$\leq 5,610$	$\leq 5,610$

The *cost* of an operation is defined as the number of computations of the hash function h . Therefore, the cost of KeyGen algorithm is always $w(l - 1)$, and the costs of Sign and Verify algorithms are both $w(l - 1)$ at the maximum. Table 5.1 shows, for some values of l , corresponding values of w_1 , w_2 , w , the size of the signature (denoted by $|\sigma|$), and the cost of each algorithm to accommodate a fingerprint of 160 bits (i.e., $n = 160$). From the table, we can observe a trade-off relation between the size of a signature and the costs of operations, although the decrease in the size of a signature seems to “saturate” with the increase in l . In an asymptotic order notation, the size of the signature is in $O(1/\log l)$, and the costs of the algorithms are in $O(l/\log l)$. This suggests that having large l noticeably increases the costs of operations, while the reduction of the signature size is quite limited.

Table 5.2. Relation among l , w and $N_{l,w} = \log_2 |\mathcal{T}_{l,w}|$

	$w = 16$	32	48	64
$l = 100$	61.06	99.80	129.08	152.93
300	83.76	144.74	194.55	237.50
500	94.59	166.68	227.19	280.44
700	101.77	181.34	249.12	309.47
900	107.16	192.35	265.66	331.43

A.3 Constant-Sum Fingerprinting

Specification of the Function

For positive integers l and w , define $\mathcal{T}_{l,w} = \{(t_1, \dots, t_w) : t_i \in \{0, \dots, l\}, t_1 + \dots + t_w = l\}$, which is the set of tuples of w non-negative integers that sum to l . An (l, w) -constant-sum fingerprinting is a fingerprinting function that maps a message of an arbitrary length to a tuple in $\mathcal{T}_{l,w}$. In this paper, an (l, w) -constant-sum fingerprinting function is denoted by $f_{l,w}$. The co-domain of $f_{l,w}$ contains $|\mathcal{T}_{l,w}| = (l+w-1)!/(l!(w-1)!)$ tuples in total. To illustrate the relation among the size of the co-domain and the parameters l and w , we define $N_{l,w} = \log_2 |\mathcal{T}_{l,w}| = \log_2(l+w-1)!/(l!(w-1)!)$. Use Stirling's formula $x! \approx \sqrt{2\pi x}(x/e)^x$, and $N_{l,w}$ can be approximated as

$$N_{l,w} \approx \left(l + w - \frac{1}{2}\right) \log_2(l + w - 1) - \left(l + \frac{1}{2}\right) \log_2 l - \left(w - \frac{1}{2}\right) \log_2(w - 1) - \frac{1}{2} \log_2 2\pi.$$

Table 5.2 shows the approximations of $N_{l,w}$ for various combinations of l and w . We can see, for example, $N_{l,w} = 61.06$ for $l = 100$ and $w = 16$, and $\mathcal{T}_{100,16}$ contains $2^{61.06}$ tuples. For the sake of security, the co-domain of $f_{l,w}$ should have enough number of tuples, say, $N_{l,w} \geq 160$ (equivalently $|\mathcal{T}_{l,w}| \geq 2^{160}$). A numerical computation determines the smallest value of l that makes $N_{l,w} \geq 160$ for a given w , and the results of the computation are shown in the first two columns of Table 5.3 (read the bottom line of the table, for example, as $N_{56,128} \leq 160 \leq N_{57,128}$). By selecting parameters l and w from this table, we can make the constant-sum fingerprinting function to have a co-domain of size 2^{160} or greater.

Table 5.3. Smallest l that makes $N_{l,w} \geq 160$, and the efficiency of the scheme

w	l	$ \sigma $	KeyGen	Sign	Verify
16	10,430	$16L$	166,880	156,450	10,430
32	429	$32L$	13,728	13,299	429
48	171	$48L$	8,208	8,037	171
64	111	$64L$	7,104	6,993	111
80	86	$80L$	6,880	6,794	86
96	72	$96L$	6,912	6,840	72
112	63	$112L$	7,056	6,993	63
128	57	$128L$	7,296	7,329	57

Construction of the Function

A constant-sum fingerprinting $f_{l,w}$ can be constructed by combining an arbitrary cryptographic fingerprinting function f and a *mapping* from integers to tuples in $\mathcal{T}_{l,w}$. The base fingerprinting function f must be selected in such a way that its co-domain contains exactly $|\mathcal{T}_{l,w}|$ fingerprints. We do not discuss the specific construction of f because there are several techniques to construct a fingerprinting function with a specified size of co-domain. The fingerprint by the function f is regarded as an integer, and we let $\mathcal{I}_{l,w} = \{0, \dots, |\mathcal{T}_{l,w}| - 1\}$ be the co-domain of f without loss of generality.

A focal point in our construction is the mapping that translates an integer in $\mathcal{I}_{l,w}$ to a tuple in $\mathcal{T}_{l,w}$. We would like to construct a bijective function $M_{l,w}(i) : \mathcal{I}_{l,w} \rightarrow \mathcal{T}_{l,w}$ for given parameters l and w , and define the constant-sum fingerprinting function $f_{l,w}$ as $f_{l,w}(m) = M_{l,w}(f(m))$. In the rest of this section, we show that the computation of $M_{l,w}(i)$ can be done in a recursive manner. Notice that the argument i is an integer in $\mathcal{I}_{l,w}$ and $M_{l,w}(i)$ results in a tuple (t_1, \dots, t_w) .

First, remark that

$$|\mathcal{T}_{l,w}| = |\mathcal{T}_{0,w-1}| + \dots + |\mathcal{T}_{l,w-1}|.$$

This equation can be understood as $\mathcal{T}_{l,w}$ being partitioned into $l + 1$ subsets

according to the value of the first component of the tuple, and $|\mathcal{T}_{k,w-1}|$ with $0 \leq k \leq l$ gives the number of tuples $(l-k, t_2, \dots, t_w)$ in the k -th subset of $\mathcal{T}_{l,w}$. Based on this observation, we consider partitioning the domain $\mathcal{I}_{l,w}$ of $M_{l,w}$ into $l+1$ sections, where the k -th section with $0 \leq k \leq l$ contains $|\mathcal{T}_{k,w-1}|$ integers. If the argument i of the mapping $M_{l,w}$ belongs to the k -th section of the domain, then the first component of the output is set to $l-k$. The remaining $w-1$ components should satisfy $t_2 + \dots + t_w = k$, and they can be selected by $M_{k,w-1}(i')$ where i' is an integer that is obtained by “offsetting” the original argument i .

To implement the above construction, we need to know $|\mathcal{T}_{k,w-1}|$ for all $0 \leq k \leq l$. Notice that

$$|\mathcal{T}_{k,w-1}| = \frac{(k+w-2)!}{k!(w-2)!},$$

which derives a recursion

$$|\mathcal{T}_{0,w-1}| = 1, \quad |\mathcal{T}_{k,w-1}| = \frac{k+w-2}{k} |\mathcal{T}_{k-1,w-1}| \quad (k > 0).$$

By using this recursion, we can compute all of $|\mathcal{T}_{k,w-1}|$ for $0 \leq k \leq l$. These values determine the “borderlines” of the sections of $\mathcal{I}_{l,w}$. Define

$$b_k = \sum_{i=0}^{k-1} |\mathcal{T}_{i,w-1}| \quad (0 \leq k \leq l+1).$$

Confirm that $0 = b_0 < \dots < b_{l+1} = |\mathcal{T}_{l,w}|$, and the k -th section $[b_k, b_{k+1}) = \{b_k, \dots, b_{k+1} - 1\}$ contains exactly $|\mathcal{T}_{k,w-1}|$ integers. Summarizing the discussion above, an algorithm to compute the mapping $M_{l,w}(i)$ is given as follows:

1. If $w = 1$, then terminate the procedure with $(t_1) = (l)$ as an output. If $w > 1$, then continue to the next step;
2. $k \leftarrow 0, a \leftarrow 1, b_L = 0, b_R \leftarrow a; \quad /* a = |\mathcal{T}_{k,w-1}| */$
3. while $i \notin [b_L, b_R)$ do
4. $k \leftarrow k + 1, a \leftarrow a(k+w-2)/k, b_L \leftarrow b_R, b_R \leftarrow b_R + a;$
5. $t_1 \leftarrow l - k; \quad /* because i \in [b_L, b_R) = [b_k, b_{k+1}) */$

6. $(t_2, \dots, t_w) \leftarrow M_{k,w-1}(i - b_L)$;
7. Output (t_1, \dots, t_w) ;

We remark that the constant-sum fingerprinting function $f_{l,w}(m) = M_{l,w}(f(m))$ has exactly the same statistical property as the base fingerprinting function f . If f is collision-resistant, then so is $f_{l,w}$, for example.

A.4 Winternitz OTS with Constant-Sum Fingerprinting

Description of the Scheme

The constant-sum fingerprinting function can be integrated with Winternitz OTS.

Assume that there are a collision-resistant constant-sum fingerprinting function $f_{l,w}$ and a hash function $h : \{0, 1\}^L \rightarrow \{0, 1\}^L$ with the one-way property. Three algorithms of the OTS are given as follows.

- **KeyGen**(1^n): Choose parameters l and w so that $N_{l,w} \geq n$, and select an (l, w) -constant-sum fingerprinting function $f_{l,w}$. The signing key SK is defined as $\text{SK} = (s_1, \dots, s_w)$, where s_i with $1 \leq i \leq w$ is uniformly selected from $\{0, 1\}^L$ at random. The verification key VK is defined as $\text{VK} = (h^l(s_1), \dots, h^l(s_w))$.
- **Sign**(SK, m): The signature for the message m is defined as $\sigma = (h^{l-f_1}(s_1), \dots, h^{l-f_w}(s_w))$, where $(f_1, \dots, f_w) = f_{l,w}(m)$ is the constant-sum fingerprint of m .
- **Verify**(VK, m, σ): The pair of the message m and the signature $\sigma = (\sigma_1, \dots, \sigma_w)$ is accepted if and only if $(h^{f_1}(\sigma_1), \dots, h^{f_w}(\sigma_w))$ coincides with the verification key VK, where $(f_1, \dots, f_w) = f_{l,w}(m)$ is the constant-sum fingerprint of m .

Signature Size and Operation Costs

In the proposed scheme, the size of the signature is wL -bits because the signature consists of w hash values of L -bits each. The cost of KeyGen, which is measured by the number of computations of the hash function h , is always lw . The costs

of Sign and Verify are $lw - l$ and l , respectively, which are independent of the message m and the signature σ .

The last four columns of Table 5.3 show the size of the signature and the costs of the three operations of the modified Winternitz OTS. The size of the signature is controllable by the parameter w , and it can be arbitrarily reduced at the expense of the increase in operation costs. This is advantageous compared to the original Winternitz OTS, in which there is difficulty in reducing the size of a signature (see Table 5.1). The size of a signature is especially important in a wireless communication over battery-operated devices because wireless communication consumes much more energy than in-device computation, and the reduction of data size (signature size) contributes more in saving energy than the reduction of computational costs.

Notice also that the cost of Verify is extremely small in the proposed scheme. Unfortunately, the costs of KeyGen and Sign are much larger than the cost of Verify, and they can be larger than those costs of the original Winternitz OTS with similar signature size. The large costs of KeyGen and Sign are certainly not preferable, but they are not so problematic in some kinds of network applications. Consider for example a wireless sensor network that consists of a centralized server and many sensor nodes. Typically, the server is powered and equipped with sufficient computational resources so that it can manage a large number of nodes and the data that are collected through those nodes. Sensor nodes are, on the other hand, small battery-operated devices that are not given enough computational resources. In this type of network systems, data communications are not equally significant. For example, directive messages from the server are quite important and they should be authorized and thus signed by an OTS scheme. On the other hand, upstream raw data from a node are less significant and do not require signatures. In this case, KeyGen and Sign algorithms are performed only by the server, which has sufficient computational power and thus can perform thousands of hash computations easily. On the other hand, sensor nodes only need to perform Verify algorithm, which is quite light-weight in the proposed scheme.

A.5 Conclusion

A modification of Winternitz OTS is proposed and shown to have provable security. The modification increases the operation costs for generating a key and for signing a message, but effectively reduces the operation cost of the verification and the size of a signature. In certain environments including a wireless sensor network, the downside of the proposed scheme is not so problematic while the benefits are highly favorable. Our approach can be combined with the investigations in [78, 80] to further improve Winternitz OTS. The authors conjecture that the combined scheme is also provably secure in the sense of being strongly existentially unforgeable under chosen-message attacks, but the details are yet to be discussed.

B. Trans-Organizational Role-Based Access Control in Android

B.1 Introduction

The role-based access control (RBAC) [14] is a widely accepted framework that describes the access control relation among users and services. In RBAC, users are associated with roles, and roles are associated with services. This framework is compatible with the access control requirements of real-world organizations and is employed in the computer systems of many organizations/companies. However, it must be noted that RBAC is a versatile framework, and roles are often used in a trans-organizational manner. For example, students are often allowed to be admitted to a museum with discounted admission fee. In this example, the “student” role that is issued by an organization (school) is used by another organization (museum) to determine if a guest is eligible to receive a certain service (discounted admission). This kind of trans-organizational use of roles is, unfortunately, not common in computer networks. Even if one has a certain role that is issued by an organization, there is no way to convince a third-party organization that he/she really has that role.

To realize a trans-organizational RBAC mechanism in a computer network, two issues should be considered; the security and the flexibility. With regard to security, the mechanism should prevent malicious users from disguising their roles. This requirement is naturally accomplished in real-world services with the use of physical certificates, such as passports and ID-cards, which are difficult to forge or alter. This problem, however, is not obvious in a computer system. Digital certificates [15] can be utilized as an analogue of physical certificates, but the use of digital certificates is not favorable from the viewpoint of realization cost, which can discourage small companies and non-profit organizations from participating in the framework. Another less sophisticated approach to the security problem is to let a service-providing organization (the museum in the above example) inquire a role-issuing organization (school) about the user-role assignment. This approach works fine in some cases [16], but a focal point of this approach is the necessity for the agreed beneficial relationship among organizations. Consequently, it is

difficult for a new organization to join the partnership, severely restricting the trans-organizational utilization of roles.

This study aims to develop a practical mechanism that realizes the trans-organizational utilization of roles. First, we extend the model of RBAC to represent the trans-organizational usage of roles. This simple extension clarifies the components and requirements that are needed in the framework of trans-organizational RBAC. Then, we investigate a realization of a user-role assignment that is secure (users cannot disguise roles), user-oriented (users can disclose their roles to any organization), and open (anyone can verify if a user has a certain role that is managed and issued by another organization). The crucial point of this realization is to make use of hierarchical ID-based encryption (HIBE) [53, 54], which allows an arbitrary string to be used as a public encryption key. Our key idea is to define correspondence between the roles and keys of HIBE and to employ a challenge-response authentication protocol that will be used for verifying if a user really has an asserted role. The hierarchical nature of HIBE makes our scheme suitable for the trans-organizational utilization of roles, and furthermore, allows flexible role management operations, such as the endorsement and delegation of roles. A prototype system of the proposed trans-organizational RBAC will also be introduced. For the usability of the trans-organizational RBAC, it is highly desirable for a user to be able to carry his/her roles all the time. To verify the practicality of the proposed scheme, we implemented the proposed system on Android-enabled mobile devices. The implementation contains the realization of cryptographic functions that are essential for handling cryptographic keys, and the development of a scheme that allows two devices to perform the challenge-response authentication by utilizing local and closed communication. The prototype demonstrates that the proposed scheme is simple, lightweight, and completely practical for realizing the trans-organizational RBAC. The rest of this paper is organized as follows. Section B.2 introduces the RBAC and the different models associated with it. Section B.3 discusses the HIBE and its functions. Section B.4 presents the structure, procedures, and features of the proposed framework. Section B.5 discusses the realization and implementation of the proposed system in Android-enabled devices. Section B.6 provides the conclusion and future work.

B.2 Models for the Role-Based Access Control

In the simplest model of the RBAC [14], the access structure is defined by three sets and two relations; the set U of *users*, the set R of *roles*, the set S of *services*, a *user-role assignment*, $UA \subset U \times R$, and a *role-service assignment* $SA \subset R \times S$. A user u is eligible to access a service s if and only if there is a role r such that $(u, r) \in UA$ and $(r, s) \in SA$. The access control should be made in such a way that services are provided to eligible users only. In general, the user-role assignment UA is defined by an entity that issues roles in R , and the role-service assignment SA is defined by an entity that provides the services in S . In this paper, the former is called a *role-issuing entity*, and the latter is called a *service-providing entity*. If RBAC is utilized in a single organization, then we can regard that the role-issuing entity and the service-providing entity are the same identical organization, and that the service-providing entity should have no difficulty referring to the user-role assignment. In this case, the eligibility of a user u to a service s can be easily determined.

On the other hand, in the real world, there are many cases wherein a service-providing entity is a different organization from a role-issuing entity. As stated in the previous section, a school issues the “student” role to its students, and an external organization, such as a museum, provides services to users who hold the “student” role. In this case, the service-providing organization (museum) is a completely independent organization from the role-issuing organization (school), and the service-providing organization is not expected to refer to the user-role assignment that was defined by the role-issuing organization. To discuss such a situation, we first consider an extended model of RBAC.

The *trans-organizational RBAC* is defined similarly to the usual RBAC, but a set O of *organizations* is defined in addition to the sets of users, roles, and services. Furthermore, the set R of roles is partitioned into several subsets, with each subset of R associated with an element in O , that is, $R = R_{o_1} \cup \dots \cup R_{o_n}$, where $o_1, \dots, o_n \in O$ and $R_{o_i} \cap R_{o_j} = \phi$ if $i \neq j$. To make the relation among roles and organizations explicit, a role r in R_{o_1} is written as $o_1.r$. Similarly, the user-role assignment UA is partitioned into disjoint subsets; $UA = UA_{o_1} \cup \dots \cup UA_{o_n}$, where $UA_{o_i} \subset U \times R_{o_i}$. Obviously, $o_1.r \in R_{o_i}$ means that the role $o_1.r$ is managed by the organization o_i and the assignment of users to $o_1.r$ is controlled

by that organization o_i . In the trans-organizational RBAC, a user u demands a service s by asserting his/her role $o_1.r \in R_{o_i}$ that has been provided by a role-providing entity (organization) o_1 . The service-providing organization provides the service to the user if and only if $(u, o_i.r) \in UA_{o_i}$ and $(o_i.r, s) \in SA$. Note that the test of $(o_i.r, s) \in SA$ is easy for the service-providing organization because the assignment SA is defined by the organization itself. On the other hand, the test of $(u, o_i.r) \in UA_{o_i}$, which is sometimes called an *authentication*, is not as obvious for the service-providing organization as in the single-organization case. If the confirmation cannot be established, then a malicious user may try to access a service by asserting a role that the user does not actually have.

It is essential in the trans-organizational RBAC to realize a secure authentication mechanism, and this problem can be solved using two approaches. The first approach is to utilize digital certificates that are protected by the digital signatures of the role-issuing entities. This kind of certificate is sometimes called an attribute certificate [15] and is regarded as a digitalized version of physical certificates, such as ID-cards. The problem in this approach is the maintenance cost of the public-key infrastructure (PKI) [50, 51]. Different from written signatures, continuous efforts are essential to keep digital signatures secure and functional. PKI is widely recognized as expensive, and this cost issue prevents small organizations from participating in a PKI-based framework. The second, rather political, approach to the authentication problem is to arrange a mutual agreement between role-issuing organizations and service-providing organizations. If several organizations share an identical benefit, then they can set up a partnership and mutually disclose their user-role assignments. A good example of this approach can be found in the Shibboleth project [16], but we need to remark that this framework is essentially a semi-closed one. An organization will not be allowed to join the partnership if that organization cannot offer recognizable benefits to the organizations involved, consequently limiting the trans-organizational utilization of roles.

B.3 Hierarchical ID-Based Encryption

A public-key encryption is a cryptography that utilizes two different keys for encryption and decryption. In a typical public-key encryption, such as RSA [55],

a user sets up his/her key pair by himself/herself. One of the keys in the key pair is called an encryption key and is disclosed to the public. The other key is called a decryption key and is kept secretly by the user. In many cases, the keys are constructed from randomly selected information, which means that the keys look like random data. A separate mechanism, such as PKI, is needed to associate public encryption keys with users. However, PKI makes the system complicated and costly [50, 51].

An ID-based encryption [53] is a special public-key encryption. Different from the usual public-key encryption, a user first chooses his/her encryption key. An interesting point in the ID-based encryption is that, under an appropriate setting, one's identity, such as an e-mail address, can be used as an encryption key. After choosing the encryption key, the user submits the encryption key to a trusted authority which we call a *key generator*. The key generator examines the eligibility of the user and then, upon confirmation of the user's eligibility, computes the decryption key that corresponds to the submitted encryption key. Different from the usual public-key encryption, the correspondence between users and encryption keys becomes obvious if the identities of users are chosen as the encryption keys. Consequently, the costly mechanism of PKI is not needed in the framework of ID-based encryptions [53].

The HIBE [54] is an extension of the ID-based encryption wherein identities and functions of the key generator are realized in a hierarchical manner. In this paper, we write a hierarchical identity (abbreviated simply as ID) by a sequence of strings $S = s_1.s_2.\dots.s_n$, where n is a non-negative integer called a *level* of S , and s_i with $1 \leq i \leq n$ is a string. If an ID $S = s_1.s_2.\dots.s_n$ is a prefix of $S' = s'_1.s'_2.\dots.s'_{n'}$, then we say that S is a *super-ID* of S' and S' is a *sub-ID* of S . In HIBE, an ID can be regarded as an encryption key by itself, although, it is sometimes convenient to distinguish IDs from encryption keys explicitly. In the following discussion, we write ek_S and dk_S for the encryption and decryption keys that correspond to the identity S , respectively. In the original ID-based encryption, all decryption keys are solely generated by a trusted key generator. In HIBE, however, the generation of decryption keys is made in a hierarchical manner; the key pair (ek_S, dk_S) for a level-one ID $S = s_1$ is generated by a designated key generator which we call a *root key generator* and is issued to an

appropriate user who is eligible to hold the key pair. A user who has a key pair (ek_S, dk_S) for an ID S can generate a key pair $(ek_{S'}, dk_{S'})$ for an ID S' that is a sub-ID of S . The functions used in HIBE are described below. There is complicated mathematics behind these functions, but we omit them because they are not the subject of this study. The main body of HIBE consists of the following four procedures:

Initialize() is a procedure that is executed by the root key generator in the initialization of the HIBE system. This procedure determines the public and secret parameters used in the system. The secret parameter is kept secretly by the root key generator, and the public parameter is disclosed to the public. The value of the public parameter is used in the following three procedures, although we do not write them explicitly in the notation for simplicity.

KeyGenerate $(S, (ek_{S'}, dk_{S'}))$ is a procedure that computes the decryption key for the given ID S . More precisely, the procedure generates dk_S if S is a sub-ID of S' and $dk_{S'}$ is a correct decryption key of S' . If not, the procedure fails to compute dk_S . We remark that dk_S cannot be computed if one does not know a correct decryption key of a super-ID of S .

Encrypt (k, m) encrypts data m by using k as an encryption key.

Decrypt (k, c) decrypts data c by using k as a decryption key.

If (ek_S, dk_S) is a key pair that is correctly generated by KeyGenerate and $c = \text{Encrypt}(ek_S, m)$ is an encryption of m constructed by using the encryption key ek_S , then $\text{Decrypt}(dk_S, c)$ returns m .

HIBE is useful when used in a challenge-response authentication protocol. Consider a scenario that involves two people, a *prover* and a *verifier*. The prover asserts himself/herself as a genuine user with an identity S , but the verifier is not sure if this assertion is true or not. In this case, the verifier can determine if the prover is genuine or not by executing the following steps. First, the verifier chooses a random message m , and then encrypts m by using the encryption key ek_S for the asserted ID S . The obtained ciphertext $c = \text{Encrypt}(ek_S, m)$, which is called a *challenge*, is passed to the prover. The prover is requested to decrypt c by using the decryption key dk_S that a genuine user should possess. If the prover returns m as the result of the decryption, then he/she succeeds to make a correct response and is authorized as a user with the identity S . If the response is wrong,

then the prover is rejected. Through this challenge-response protocol, the verifier is able to determine if the prover has the ID S . At this point, it should be noted that the ID S is indeed a hierarchical identity; the fact that the prover possesses the decryption key dk_S means that somebody who had a certain super-ID of S authorizes the prover to have the identity S . In other words, with the challenge-response protocol, the prover confirms a “chain of trust” that originates from the root key generator. This scenario is favorable in an open system with many unspecified users.

B.4 Proposed Scheme

Overview

We now consider an authentication mechanism that is suitable for the trans-organizational utilization of roles. Our idea is to represent roles by hierarchical identities that work as encryption keys of HIBE. For example, if we would like to define a “student” role of A-university, then a hierarchical identity, such as “A-univ.student”, is introduced and used as an encryption key of HIBE. Decryption keys are managed so that users with a role r possess the correct decryption key dk_r of r . A service-providing organization can verify if a user has the role r by examining the user by using the challenge-response protocol with r used as an encryption key of HIBE. Note that the service-providing organization does not have to know anything about r beforehand and does not have to make any contract or inquiry to the role-issuing organization that has assigned r to the user because r itself is used as an encryption key. This feature makes it easy to verify the user-role assignment of users even if the role is issued by another organization. In the proposed framework, there is no essential difference between users and role-issuing organizations because they both receive valid key pairs from superordinate entities and they both have the ability to generate new roles and corresponding key pairs by utilizing their keys in possession. However, for an easy understanding of the proposed framework and for simplicity, we will distinguish users from role-issuing organizations and introduce three component procedures that are needed for defining a user-role assignment. In the following, we extend the hierarchical notions of IDs to roles. If r_1 is a super-ID of another ID r_2 , then

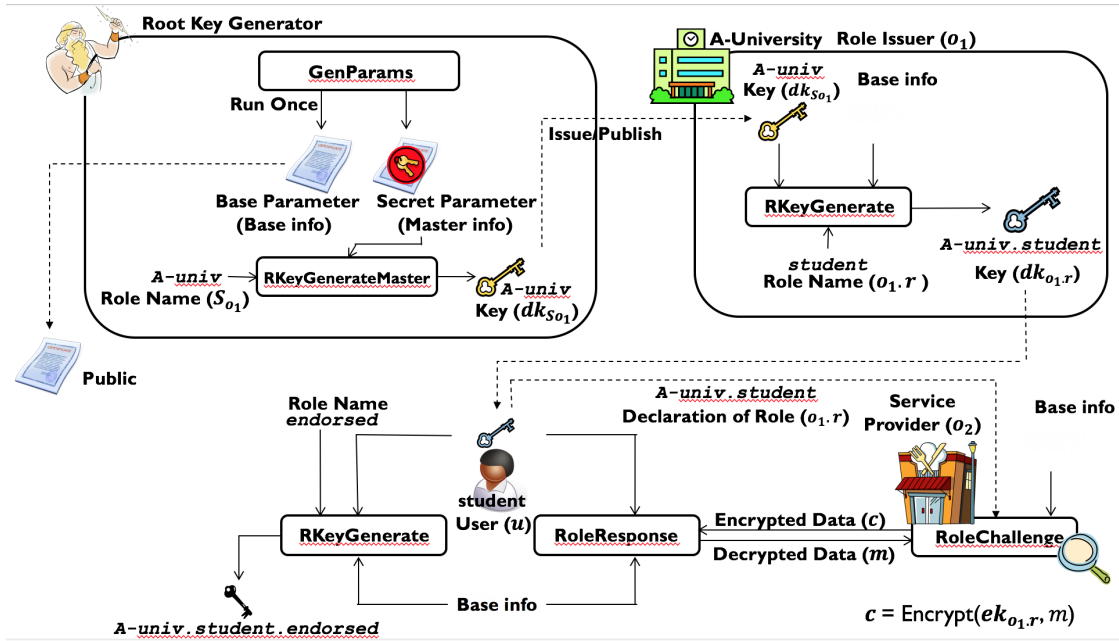


Figure 5.1. Overview of the proposed RBAC using HIBE.

the role represented by r_1 is a *super-role* of the role represented by r_2 . The term *sub-role* is defined in the same way.

Procedures

Figure 5.1 shows the overall structure of the proposed model. In this model, we assume the existence of a designated root key generator that is trusted by all users and organizations. The root key generator executes `Initialize()` of HIBE and determines the public and secret parameters. The public parameter is disclosed to the public and should be accessible to all users and organizations. The root key generator secures the secret parameter and uses it to generate key pairs for level-one roles.

1) Setting up an organization

An organization o_1 chooses its identity string, say S_{o_1} , and requests the root key generator to approve that the organization uses S_{o_1} as its identity. If the root key generator approves the request, it computes the decryption key $dk_{S_{o_1}}$

and sends this key to oo_1 by using a secure communication channel. As a result, o_1 possesses a correct key pair $(ek_{S_{o_1}}, dk_{S_{o_1}})$ of its identity S_{o_1} . The organization o_1 then defines the set R_{o_1} of roles it should manage. All roles in R_{o_1} must be sub-roles of S_{o_1} , where the identity S_{o_1} is regarded as a “role”. Note that o_1 can compute the decryption key of any role $o_1.r \in R_{o_1}$ by utilizing the function of $\text{KeyGenerate}(o_1.r, (ek_{S_{o_1}}, dk_{S_{o_1}}))$, because o_1 knows the correct key pair $(ek_{S_{o_1}}, dk_{S_{o_1}})$ and $o_1.r$ is a sub-ID of S_{o_1} .

On the other hand, organizations other than o_1 cannot compute the decryption key of the role $o_1.r$ because the trusty key generator does not disclose $dk_{S_{o_1}}$ to other organizations. Identities of roles in R_{o_1} can be open to the public, but the corresponding decryption keys must be kept secret by organization o_1 .

2) Defining a user-role assignment

To assign a role $o_1.r$ to a user u , the organization o_1 gives the key $dk_{o_1.r}$ to user u by using a secure communication channel. User u records $dk_{o_1.r}$ as the decryption key of the role $o_1.r$, and keeps the key secure.

3) Verifying a user-role assignment

Assume that a user u visits a service-providing organization o_2 and asserts that he/she has the role $o_1.r$ that was assigned by the role-issuing organization o_1 . Organization o_2 needs to verify if the assertion of u is true or not. The verification can be done by using the challenge-response protocol; o_2 chooses a random data m and requests u to decrypt $c = \text{Encrypt}(ek_{o_1.r}, m)$. Note that we are using HIBE, and the encryption key $ek_{o_1.r}$ is the same as (or easily derived from) the hierarchical identity $o_1.r$ of the asserted role. If u really has the role $o_1.r$, then he/she must possess the decryption key $dk_{o_1.r}$ that is provided by o_1 and should be able to decrypt the challenge c . Remark that o_2 can verify if u holds the role $o_1.r$ without querying o_1 and that u has little chance to disguise his/her role.

Managing Roles

1) Personalization of roles

In the proposed framework, the relation between users and roles is represented by the possession of cryptographic keys by users. This approach involves a possible security risk; a leakage of keys. If, for example, a role r is assigned to several

users, then all those users have the same key dk_r . If one of those users is unconscious of security, then he/she may let other persons use the key dk_r . Such an inappropriate usage of keys can obstruct fair and reliable access control. To deter such irresponsible behavior of users, a role-issuing organization can “personalize” a role by appending an additional string to the identity of roles. Assume for example that there are several students in A-university. In this case, instead of using a general role, such as “A-univ.student”, the university can define personalized roles, such as A-univ.student.Alice and A-univ.student.Bob, and provide decryption keys of these roles to Alice and Bob, respectively. With this kind of personalization, a student will be more conscious of leaking/losing his/her key to another person because he/she will have the risk of being identified and subsequently punished for irresponsible behavior. The theft/loss of keys remains a risk, but such risk also exists for ID-cards used in the real world. We cannot say that the proposed framework is “more secure than” but we may say it is “as secure as” the real-world role management.

2) *Hierarchical issuance of roles*

In the proposed scheme, there is no essential difference between organizations and users. A user can compute decryption keys from the key that he/she already has and issue a new sub-role of the role that he/she already has. This function can be used to realize some personal activities that are not considered in the conventional RBAC approach. One possible example is the endorsement of another person. In the real world, an endorsement among individuals sometimes plays an important role. Semi-closed organizations, such as academic societies and golf clubs, have the tradition or policy that a newcomer must be endorsed or referred by a current member. This mechanism can be realized using the proposed scheme. Consider for example that Alice is an authorized member of XYZ golf club and is given a personalized role “XYZ-golf.member.alice” and its corresponding decryption key. If Alice would like to endorse Bob to the club, then she can generate a new sub-role “XYZ-golf.member.alice.endorsed” and its corresponding decryption key. By providing the decryption key to Bob, Bob can demonstrate that he is really endorsed by Alice. Using the HIBE and the challenge-response authentication, the club does not have to inquire Alice for the verification of the endorsement. Besides personal endorsement, we conjecture that a broad range of

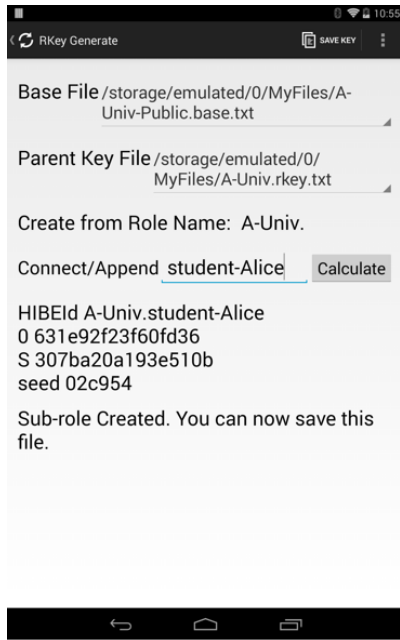
personal relations can be implemented by utilizing the hierarchical roles.

B.5 Realization

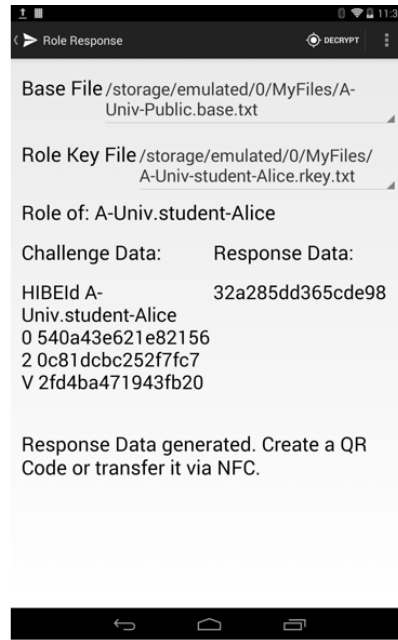
The proposed scheme was implemented in Android-enabled mobile devices. In the proposed scheme, the user-role assignments are represented by possession of decryption keys by users. A role-issuing organization does not have to construct and maintain large databases for recording the user-role assignments, and it does not have to be bothered by inquiries of other organizations with regard to user-role assignments. The created Android application implements all the functions of the HIBE and the proposed scheme for ease of use and accessibility.

The prototype contains several functions, as shown in Figure 5.2. The functions of the root key generator mainly consist of two operations: GenParams and RKeyGenerateMaster. GenParams utilizes the Initialize() function of HIBE and generates the public and secret parameters, where the public parameter is disclosed to the public. An organization o_1 that would like to participate in this system chooses its identity, say S_{o_1} , and asks the root key generator to compute the decryption key $dk_{S_{o_1}}$. The root key generator utilizes RKeyGenerateMaster to compute $dk_{S_{o_1}}$, which needs the information of the secret parameter and hence this function is accessible to the root key generator only. The generated key $dk_{S_{o_1}}$ is transferred to the organization o_1 through general communication means, such as Wi-Fi, Bluetooth, Android Beam, and NFC. The role-issuing organization o_1 now has the key pair $(ek_{S_{o_1}}, dk_{S_{o_1}})$ for the ID S_{o_1} . By using the function of RKeyGenerate, this key pair, and the public parameter that has been disclosed, the organization o_1 can compute valid key pairs of sub-IDs of S_{o_1} . A user receives, possibly many, keys from organizations, each of which corresponds to a role in an organization. The user safely stores these keys in his/her device and accesses them for the RKeyGenerate and RoleResponse functions. RoleResponse provides the function of the prover for the challenge-response authentication and interacts with the RoleChallenge function that is invoked by a service-providing organization.

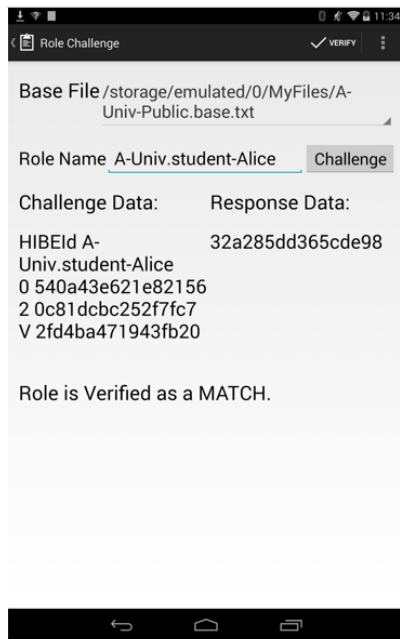
The cryptographic operations used in these functions are performed using the Java Pairing-Based Cryptography (JPBC) library [86], which is a collection of classes and methods for handling underlying pairing-based cryptosystems. Over



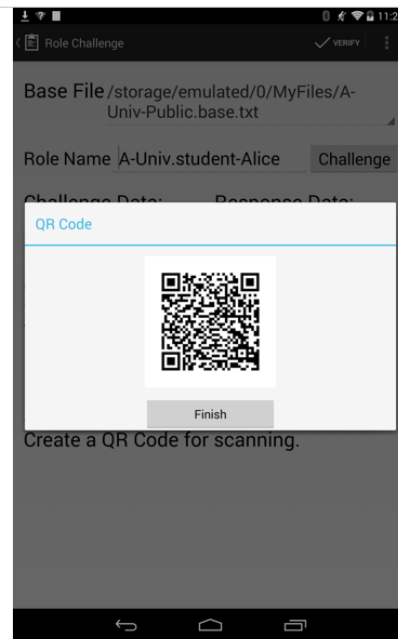
(a)



(b)



(c)



(d)

Figure 5.2. (a) RkeyGenerate, (b) Role Response, (c) Role Challenge, and (d) QR Code functions of the application.

JPBC, we implemented the HIBE that was proposed by Gentry [54].

The most complicated but important communication in the proposed scheme is the challenge-response authentication between a user and a service-providing organization. Several messages must be exchanged between two devices, and we provide two different schemes to realize this communication, namely, the use of near-field communication (NFC) and quick-response (QR) codes.

NFC is a contactless technology used to transmit small amounts of data across short distance. NFC has three modes of operation, and this study tackles only P2P mode. NFC messages in Android are handled using the NFC Exchange Format (NDEF). In the proposed Android application, the Intent Filters that listen to the intent action of `NfcAdapter.ACTION_NDEF_DISCOVERED` were added to the `RoleChallenge` and `RoleResponse` activities to be able to receive NFC data [87, 88]. Only the MIME type of `text/plain` was included in the application as we are only concerned with passing and receiving data of type string. Given that at least two activities have the same intent filter that responds to an NFC tap, users are, by default, prompted to select which application in the mobile device to use, making the application tedious to use. To solve this problem, the foreground dispatch system was utilized. The foreground dispatch system is used to make a particular activity have priority over other activities. This allows a particular activity to become the default receiver when it is on the foreground.

QR Code is a type of 2D barcode that is capable of handling different types of data [89]. This code can accommodate high capacity of data in a small area, which is sufficient to include the challenge-response data in one code symbol. The camera hardware of mobile phones can be used as scanners for QR codes generated for the challenge/response actions. For this implementation, the camera hardware of the device was programmed and the ZXing (“zebra crossing”) library, which is an open-source library that supports the decoding and generation of barcodes, was used to obtain the data [90]. This feature allows the interaction of two users without NFC-capable devices and without the Internet.

Typically, the challenge-response authentication is carried out as follows:

1. A user, the prover, opens the application and goes to the “Role Response” option. Then, the prover selects the base file that contains the public parameter and the role file that contains the role he/she wants to use.

2. A service-providing organization, the verifier, opens the application and goes to the “Role Challenge” option. To start the verification process, the verifier selects the same base file and either types the role indicated by the prover or obtains the role automatically via NFC (first tap). Once the role is received, a random challenge data is created.

3. The verifier then sends this challenge data to the prover via NFC (second tap) or by generating a QR code for the prover to scan.

4. After receiving the challenge data via NFC or scanning of the Challenge QR Code, a random response data is calculated and created in the prover’s device based on the role file selected.

5. The prover then sends this response data to the verifier, again, via NFC (third tap) or by generating a QR code for the verifier to scan.

6. After receiving the response data via NFC or scanning of the Response QR Code, the Role Challenge indicates if the assumed role is verified or is a mismatch.

Several screen shots of the prototype are shown in Figure 5.2. Another possibility for the realization of the user-side system is through the use of smartcards that are compatible with the NFC technology [91].

B.6 Conclusion and Future Work

A trans-organizational RBAC is considered and extended to represent the trans-organizational usage of roles. The proposed scheme provides a secure mechanism for verifying the user-role assignments of organizations. The proposed scheme was developed on Android-enabled mobile devices for ease of use and accessibility. Compared to other similar approaches, the proposed scheme provides more flexibility and autonomy while maintaining security. This mechanism allows the realization of many collaborative right managements that are common in physical communication but are difficult to implement over computer networks. Even with the given advantages, the proposed scheme remains subject to the classical issue of compromised secret keys; the proposed scheme is based on the assumption that keys are managed appropriately and protected well. If dk_S is compromised for an unfortunate reason, the keys of the sub-roles of S should be redeployed. This problem can be mitigated by utilizing the personalized and fixed-term roles,

but it is encouraged in general to provide more protection for the keys of roles of higher-level. Taking such issue into consideration, future research will focus on the inclusion and integration of expiration dates on the roles. Moreover, the prototype will be expanded to non-Android devices, such as iPhones and Windows mobile devices, for interoperability.

C. Blockchain 2.0

The Bitcoin blockchain has been successful in providing a decentralized mechanism by combining proven technologies. Since the successful deployment of Bitcoin, many alternative blockchain-based systems (i.e., they use blockchain technology but with modified properties and core uses) have followed suit. These systems, with their own implementation of blockchain technology, have been developed to solve the challenges and limitations of the Bitcoin blockchain and diversify the application of the powerful blockchain technology.

C.1 Problems with the Bitcoin blockchain

With the success and increasing use of the Bitcoin blockchain, certain issues and problems have been recognized. Some of the issues that limit the full potential of the Bitcoin blockchain are as follows:

- **Bitcoin script.** The Bitcoin script is a list of instructions recorded with each transaction that describe the parameters or conditions that the next user need to satisfy to be able to spend the bitcoins being transferred [92]. The Bitcoin script is purposefully not Turing complete*, with no loops. This property of the Bitcoin script allows it to be metered, ensuring that every node follows executes the same format strictly. On the other hand, this also restricts the Bitcoin script to be used in procedures that may need loop processing.
- **Confirmation Time.** In Bitcoin, a block that contains transactions is validated and confirmed every 10 minutes, on average. In general, Bitcoin suggests that a block be confirmed by at least 60 minutes (i.e., six blocks have been mined after it) to ensure finality. This particularly applies for high-value transactions, while even 1 confirmation is enough for low-value transactions. This confirmation time is not practical for many business applications that require promptness in their transactions or functions. More-

*A system of data-manipulation rules, such as programming language or instruction set, is considered to be Turing complete if it can be used to simulate any single-taped Turing machine [93]. The concept is named after English mathematician Alan Turing.

over, this confirmation time produces a variable in the timestamp of a transaction because the timestamp affixed on the block (and therefore on the transactions) pertains to the time when the block is mined and included on the blockchain, not to the time when the transaction was created and transmitted.

- **P2P System.** Each node participating in the Bitcoin network should keep a copy of the entire blockchain, which includes all past transactions (as of November 2016, the Bitcoin blockchain is approximately 90 GB in size) [94]. This requirement limits the participation of smaller nodes with smaller capacity, and therefore may not be applicable for Internet of Things (IoT) systems. Moreover, the mining process in Bitcoin requires a significant amount of computing power, where miners are professional businesses that built structures and systems specifically for mining bitcoins. These mining systems use considerable amount of resources, i.e., electricity, and may become a problem in the future.
- **Transaction Finality.** Bitcoin is highly resistant to falsification, and thus, transactions are almost impossible to rewrite or modify once they are recorded in a block in the blockchain. Although this is favorable in a security perspective, it is not favorable for applications that need to correct transaction details retrospectively. In relation to this, transaction details in the blockchain can be verified by anyone, and thus, confidential information (unless encrypted) cannot be disclosed in the transactions.

C.2 Uses of Blockchain Technology

Wanting to solve specific or various problems associated with the Bitcoin blockchain, various blockchain-based systems have been created to expand and diversify the use of blockchain technology. Some of these systems are as follows:

- **Altcoins.** Bitcoin is open-source, and thus, many alternative virtual currency, also known as altcoins, have been created following the success of Bitcoin. More than 600 altcoins are currently considered as virtual currency with actual market value [95].

- **Transaction Record.** Given the decentralized nature of blockchain technology, many systems use it for managing goods and services transactions. This capability allows the managing of the transfer of assets and services, such as real estate and sales (e.g., tickets and coupons).
- **Proof of Rights.** Given that transactions on the blockchain are recorded indefinitely, blockchain technology provides a guarantee of authenticity, which can be applied for proving the existence of documents or proving the casting of votes.
- **Automated Processes.** Automated processes can be implemented using blockchain technology by deploying programs that have expanded scripts and that can handle future procedures and functions. One of the most famous implementation of this idea is Ethereum, which is a blockchain app platform that can run and execute programmable assets.

C.3 Ethereum

Ethereum is an open blockchain platform that can run decentralized applications based on blockchain technology [96]. Similar to Bitcoin, Ethereum is open-source and decentralized, i.e., no one entity controls Ethereum. Ethereum was designed to be adaptable and flexible, which are properties Bitcoin was lacking. In 2014, Ethereum conducted a presale of ether tokens (ETH), which is the currency unit of Ethereum, to kickstart a large network of developers, miners, investors, and other stakeholders. Beginning in July 2014, Ethereum was able to distribute approximately 60 million ether, which is approximately 31,500 BTC worth 18.5 million USD, over a 42-day public ether sale. Ethereum has almost the same functions as Bitcoin, but with the addition of being able to run and deploy smart contracts.

Smart Contracts

In Ethereum, there are two kinds of accounts:

- **Externally owned accounts (EOAs).** These are accounts that have an ether balance, can send transactions (either to transfer ether to other



Figure 5.3. The Ethereum Logo.

accounts or trigger contract code), are controlled by private keys, and have no associated code.

- **Contract accounts.** Contract accounts are created when smart contracts are deployed. They contain an ether balance (which can also be empty) and associated code.

The code in contract accounts is executed when triggered by transactions or messages from EOAs. When executed, contract accounts perform operations of varying complexity depending on the deployed code. The code is instructed by the input parameters included in the transaction sent by the EOAs. In other words, smart contracts act like “autonomous agents” embedded on the blockchain that always execute a specific piece of code when prompted or “poked” by a message or transaction.

How Smart Contracts Work

Figure 5.4 shows a snippet of a smart contract code to be deployed using the Ethereum wallet. The wallet serves as the editor and the compiler for the code written using the Solidity programming language [97], which is a contract-oriented, high-level language designed for creating smart contracts in Ethereum. If there are no errors, the smart contract is compiled and deployed onto the

```
1 pragma solidity ^0.4.2;
2
3 contract owned {
4     address public admin;
5
6     function owned() {
7         admin = msg.sender;
8     }
9
10    modifier onlyadmin {
11        if (msg.sender != admin) throw;
12        -;
13    }
14
15    function transferadminship(address newadmin) onlyadmin {
16        admin = newadmin;
17    }
18 }
19
20 contract AuctionSystem is owned {
21
22    /* Contract Variables and events */
23    //uint public minimumNumberOfBids;
24    auction[] public auctions;
25    uint public numberOfAuctions;
26    uint public numberOfMembers;
27    mapping (address => uint) public memberId;
28    Member[] public members;
29
30    struct auction {
```

SOLIDITY CONTRACT SOURCE CODE

CONTRACT BYTE CODE

SELECT CONTRACT TO DEPLOY

Auction System

Figure 5.4. Example of a smart contract deployed using the Ethereum wallet.

Ethereum network, awaiting for miners to validate the transaction and convert it into a contract account.

After the code has been validated and the contract account has been created, the functions written on the code can now be executed by anyone through transactions. Figure 5.5 shows an the different functions written in the sample code that can be executed. Details of the codes that are successfully executed are permanently recorded in the Ethereum blockchain, and they can be verified by anyone using a blockchain browser.

Smart contracts allow systems that can be translated into code to be deployed onto the Ethereum blockchain. Figure 5.6 shows some of the Dapps that have been deployed using blockchain technology.

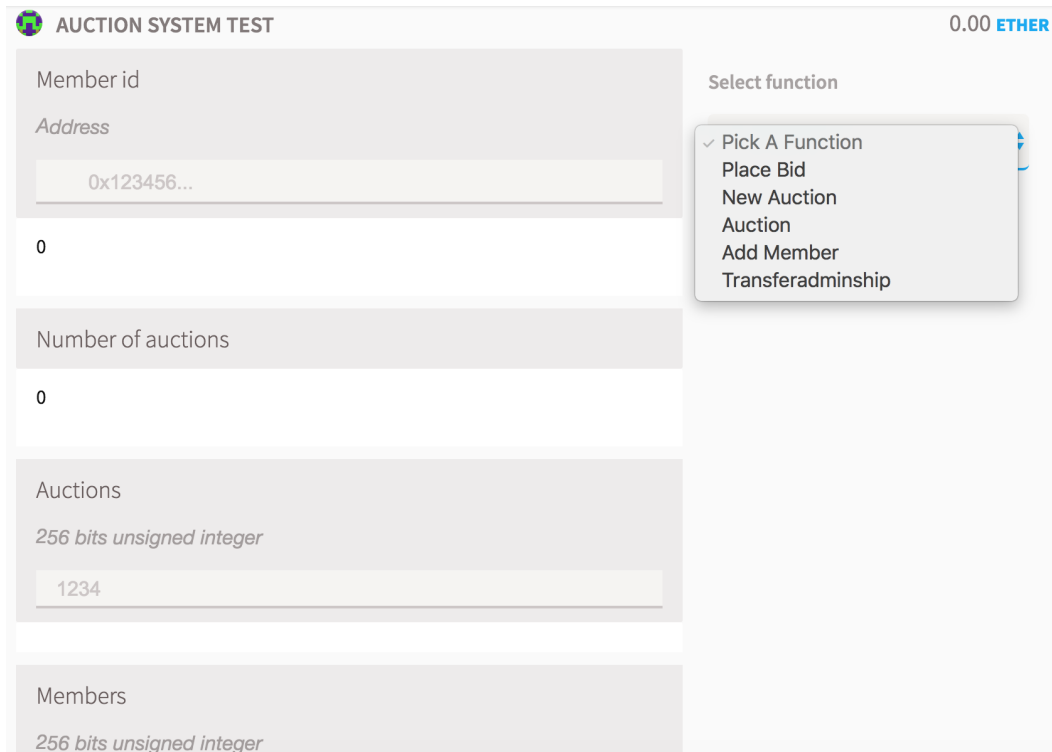


Figure 5.5. Functions that pertain to specific pieces of the code in the smart contract.



Figure 5.6. Some of the Dapps using blockchain technology [98].

Publication List

Peer-reviewed Journal Paper

1. Jason Paul Cruz and Yuichi Kaji, “The Bitcoin Network as Platform for Trans-Organizational Attribute Authentication,” *IPSJ Transactions on Mathematical Modeling and Its Applications*, Vol.9, No.2, August 2016, pp. 41–48. (related to Chapters 2 and 3).
2. Jason Paul Cruz and Yuichi Kaji, “E-voting System Based on the Bitcoin Protocol and Blind Signatures,” *IPSJ Transactions on Mathematical Modeling and Its Applications* (accepted and awaiting publication) (related to Chapters 2 and 4).

Peer-reviewed International Conference Papers

1. Jason Paul Cruz and Yuichi Kaji, “Trans-Organizational Role-Based Access Control in Android,” *The Fourth International Conference on Advanced Communications and Computation (INFOCOMP 2014)*, July 2014, pp. 114–119. (related to Chapter 3).
2. Jason Paul Cruz and Yuichi Kaji, “The Bitcoin Network as Platform for Trans-Organizational Attribute Authentication,” *The Third International Conference on Building and Exploring Web Based Environments (WEB 2015)*, May 2015, pp. 29–36. (related to Chapter 3).
3. Jason Paul Cruz, Yuchi Kaji, and Yoshio Yatani, “Constant-Sum Fingerprinting for Winternitz One-Time Signature,” *ISITA 2016, International*

Symposium on Information Theory and its Applications, October 2016.
Chapter 2 (Appendix).

Non-reviewed Local Conference, Posters, and Symposium

1. Jason Paul Cruz and Yuichi Kaji, “The Bitcoin Network as Platform for Trans-Organizational Attribute Authentication,” Modeling and Problem Solving (MPS) 102 Study Group, March 2015. (related to Chapters 2 and 3).
2. Jason Paul Cruz and Yuichi Kaji, “E-voting System Based on the Bitcoin Protocol and Blind Signatures,” Symposium on Cryptography and Information Security (SCIS), January 2016. (related to Chapters and 4).
3. Jason Paul Cruz and Yuichi Kaji, “E-voting System Based on the Bitcoin Protocol and Blind Signatures,” Modeling and Problem Solving (MPS) 107 Study Group, March 2016. (related to Chapters 2 and 4).
4. Yoshio Yatani, Jason Paul Cruz, and Yuchi Kaji, “Improvement of Winternitz One-Time Signature, Poster Presentation,” ISITA 2016, International Symposium on Information Theory and its Applications, October 2016. Chapter 2 (Appendix).