HOCHSCHULE MITTWEIDA
UNIVERSITY OF APPLIED SCIENCES

MASTER THESIS

Ms.
Shaghik Amirian

Probabilistic Micropayments

2022

Faculty of **Applied Computer Sciences and Biosciences**

MASTER THESIS

Probabilistic Micropayments

Author:
Shaghik Amirian

Study Programme:
Applied Mathematics for Network & Data Sciences

Seminar Group:
MA19w1-M

First Referee:
Prof. Dr. Klaus Dohmen

Second Referee:
Dipl. Volkswirt Mario Oettler

Mittweida, May 2022

Acknowledgement

I would like to acknowledge my supervisor, Professor Dr. Klaus Dohmen, for his guidance, kind support, and Mr. Mario Oettler for feedback throughout this thesis research. I am also thankful to my colleague Mr. Bafna for helpful suggestions. Lastly, my parents deserve endless gratitude for their loving support in this journey.

Bibliographic Information

Amirian, Shaghik: Probabilistic Micropayments, 63 pages, 10 figures, Hochschule Mittweida, University of Applied Sciences, Faculty of Applied Computer Sciences and Biosciences

Master Thesis, 2022

Abstract

Probabilistic micropayments are important cryptography research topics in electronic commerce. The Probabilistic micropayments have the potential to be researched in order to obtain efficient algorithms with low transaction costs and high speeding computer power. To delve into the topic, it is vital to scrutinize the cryptographic preliminaries such as hash functions and digital signatures. This thesis investigates the important probabilistic methods based on a centralized or decentralized network. Firstly, centralized networks such as lottery-based tickets, Payword, coin-flipping, and MR2 are described, and an approach based on blind signatures is also discussed. Then, decentralized network methods such as MICROPAY3, a transferable scheme on the blockchain network, along with an efficient model for cryptocurrencies, are explained. Then we compare the different probabilistic micropayment methods by improving their drawback with a new technique. To set the results from the theoretical analysis of different methods into some context, we analyze the attacks that reduce the security and, therefore, the system's efficiency. Particularly, we discuss various methods for detecting double-spending and eclipse attacks occurrence.

I. Contents

Contents ... I

List of Figures ... II

List of Tables ... III

1 Introduction ... 1

1.1 Thesis Research Goal ... 1

1.2 Probabilistic micropayments through time ... 2

1.3 History of Micropayment System ... 2

1.4 Thesis Structure ... 3

2 Cryptographic Preliminaries ... 4

2.1 Hash Function ... 4

2.2 Data Structure ... 4

2.2.1 Linked List ... 5

2.2.2 Merkle Trees ... 5

2.3 Consensus Mechanism in blockchain ... 6

Proof of work model ... 7

Properties for Proof of Work schemes ... 7

Hashcash ... 9

2.4 Digital signature ... 11

Elliptic Curve ... 11

Elliptic curve discrete logarithm problem ... 12

2.4.1 Elliptic curve digital signature scheme ... 14

2.4.2 ElGamal digital signature scheme ... 15

Key generation ... 15

Signature generation ... 15

Verification process ... 15

Correctness ... 16

2.4.3 Schnorr digital signature scheme ... 16

Key generation ... 16

Signing process 17

Verifying process 17

3 Analysis of various methods in probabilistic micropayments 18

3.1 Micropayment 18

3.2 Probabilistic Micropayment 19

Overview of Probabilistic Micropayment Protocol 19

Blockchain scalability 20

3.3 Analysis of various methods for probabilistic micropayments 21

3.3.1 Payword 21

3.3.2 Electronic Lottery Tickets scheme 23

3.3.3 Efficient Coin-Flipping scheme 24

3.3.4 MR2 Scheme 26

3.3.5 Blind signature scheme based on discrete logarithm 28

Blind signature for modification of DSA 29

3.4 Analysis of different methods for probabilistic micropayments based on decentralized network 30

Decentralized Network 30

Decentralized anonymous micropayments (DAM) 31

3.4.1 MICROPAY: Abstract model in cryptocurrency system for probabilistic micropayments 31

MICROPAY3 32

3.4.2 Probabilistic Micropayments with a transferable scheme in blockchain 34

Transferable scheme design steps 34

Proportional fee scheme 37

3.4.3 Efficient and double-spending resistance micropayment method for cryptocurrencies 37

The efficient micropayment scheme 38

4 Attack analysis in probabilistic micropayment methods 41

4.1 Detection method for double-spending attacks 43

4.2 Double-spend attack with concurrent of eclipse attack in blockchain 44

4.3 Adversary attacks 46

5 Conclusion 49

Summary of different methods for micropayments and probabilistic micropayments 50

Future Work 50

Bibliography 51

A Implementation: Python Code 55

A.1 Probabilistic Micropayment scheme's: Payword 55

A.2 Merkle Tree 56

A.3 Elliptic Curve Addition operation 57

A.4 Decentralized vs.Centralized visualization 59

B Mathematics 61

B.1 Elliptic Curves: Weierstrass Form for Singular & Non-singular Curves 61

II. List of Figures

1.1 Probabilistic Micropayments History 2
2.1 Merkle Tree Example 6
2.2 Timeline of consensus algorithm for evolution [1] 7
2.3 Illustration of digital signature and verification [2] 11
3.1 Example of Probabilistic micropayment 19
3.2 Visualization of centralized versus decentralized networks. 31
3.3 Outline of probabilistic micropayment with transferable scheme [3] 35
4.1 Double-spending outline [3] 42
4.2 Double-spending Attack scheme [3] 42
5.1 Summary of various methods 50

III. List of Tables

2.1 Example of finding number of points on Elliptic curve 12
3.1 Example of Transaction Fees in different structures in PayPal 18
3.2 Summary of lottery ticket scheme 24
3.3 Summary of output sequence of Coin-flipping scheme 26
3.4 Summary of MR2 scheme 27
3.5 Traffic costs of different schemes 27
3.6 Features summary for probabilistic micropayments 28

1 Introduction

Micropayment structures are based on electronic payment methods, particularly small money for virtual goods and services. One of the significant benefits of micropayment is handling an arbitrarily small amount of payments over the internet by exchanging the large advance payments with the small fraction of original fees, such as a small number of purchases involving a single-day subscription to a newspaper and other online services. Micropayments protocols consist of basic parties as a user, recipient, and trusted third party. Despite the high potential of micropayment systems, very few techniques have been successful. Continual surveying of successful methods is necessary for the case of real applications, and therefore, it is essential to allocate cryptography research to analyze various micropayment approaches, particularly probabilistic micropayments. Probabilistic micropayment schemes are introduced by Rivest and Wheeler to overcome issues related to high transaction costs in micropayments [4, 5]. The key insight is that lottery tickets are considered as tickets' expected value in the micropayment system. Participants control bank settlements by configuring the winning probability of tickets and payouts. Besides that, fair payment is secured as long as many tickets are processed and received in the long run, as per the law of large numbers in probability. Finally, the concept of a bank is replaced by blockchain so that instead of depending on the bank, each new node is analogous to a new member in the blockchain, supporting a smart contract deployment on a blockchain that is a digital agreement programmed to handle winning tickets and maintain the cryptocurrencies. Hence, probabilistic micropayments have a huge potential for researching their mathematical mechanism in blockchains.

1.1 Thesis Research Goal

There is a growing need for an effective and efficient probabilistic micropayment technology for e-commerce products and services. This thesis studies detailed probabilistic micropayments methods with the existing proposed solutions in micropayment technologies to overcome security, lack of anonymity, and performance issues.
This thesis is aimed at analyzing underlying mathematical and cryptographic aspects of probabilistic micropayments with respect to various methods in the centralized and decentralized networks along with studying possible attacks such as double spending in the network.

1.2 Probabilistic micropayments through time

Fair amount of history related to probabilistic micropayments protocols are illustrated in figure 1.1.

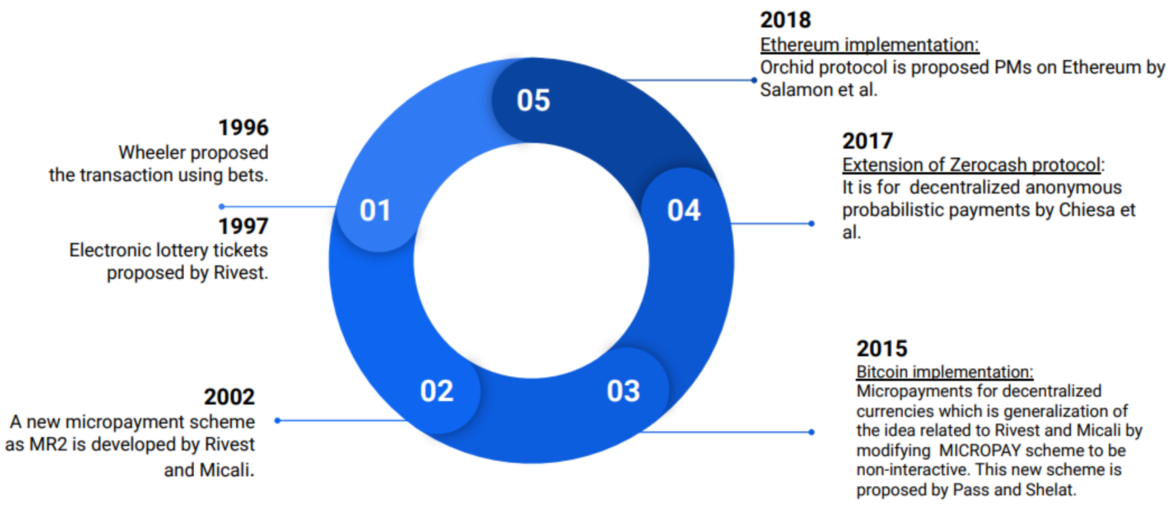
2018
Ethereum implementation:
Orchid protocol is proposed PMs on Ethereum by Salamon et al.
1996
Wheeler proposed the transaction using bets.
1997
Electronic lottery tickets proposed by Rivest.
05
01
04
02
03
2017
Extension of Zerocash protocol:
It is for decentralized anonymous probabilistic payments by Chiesa et al.
2002
A new micropayment scheme as MR2 is developed by Rivest and Micali.
2015
Bitcoin implementation:
Micropayments for decentralized currencies which is generalization of the idea related to Rivest and Micali by modifying MICROPAY scheme to be non-interactive. This new scheme is proposed by Pass and Shelat.

Figure 1.1: Probabilistic Micropayments History

1.3 History of Micropayment System

The history of micropayment dates back to 1960, when the micropayment term was initiated by Theodor Holm Nelson [6]. Nelson invented hypertext systems and has a significant role in network growth and web development concepts. During 1990 there was a huge interest in making micropayments a standard transaction method. In 1989 Chaum presented the electronic cash concept. Based on Chaum's electronic cash, developers of the micropayment system initiated the cash in electronic form, denoted as tokens, digital currency, E-Coins, E-cash, and operating it on Internet [7].
Micropayment systems are classified based on either being an account-based system or not. Until the end of the 1990s, the illustrated micropayment systems were account-based. The account-based method is equivalent to a banking system that transfers money from the user account to the merchant account. Nonetheless, the account-based methods, denoted as the first generation of micropayments, are broken down due to security, speed, and transaction cost.
The next generation of micropayments systems unfolded in 2000. The new approaches improve the issues related to security, attacks, transaction cost, and double-spending in the network. For this reason, new techniques based on probability for micropayments evolved as a suitable solution to enhance the methods regarding small payments.
Probabilistic micropayments have been an interesting section for the cryptography scientist community, including Rivest and Wheeler, along with others who are actively working in this area [4,5]. In 1996, Wheeler proposed the transaction using bets [5]. Then, in 1997 one of the most significant probabilistic micropayment algorithms was presented

by Rivest as ”Electronic lottery tickets” [4]. Pass and Shelat’s research on anonymous, decentralized systems for micropayments is combined with probabilistic micropayments to secure anonymity in the blockchain protocols [8]. Furthermore, in 2018 there is an ongoing Orchid payment project that works on cost and time efficiency for probabilistic micropayments in Ethereum blockchain [9].

1.4 Thesis Structure

This thesis research is structured as follows: Chapter 1 provides the introduction and historical background of probabilistic micropayment systems. In chapter 2 the cryptographic preliminaries are explained, including hash functions, Merkle trees, proof of work model, and digital signatures. In Chapter 3, various probabilistic micropayment methods are compared. Chapter 4 presents the attack analysis in probabilistic micropayments such as double-spending attacks. Finally, chapter 5 provides conclusions based on comparing various probabilistic methods and outlines the possible future research.

2 Cryptographic Preliminaries

2.1 Hash Function

Hash functions have a significant role in the security protocol of blockchains, mining operations, and identity recognition in the Bitcoin and Ethereum networks. Principally in the blockchain, the hash function connects the blocks firmly with high security in the blockchain. The most common hash functions implemented in different protocols are secure hashing algorithms such as SHA1, SHA2, SHA3, and message digest as MD2, MD4, and MD5 with the output of 128-bit as message-digest value. Depending on the security level required, different hashing methods are applied.
Hashing is an operation undertaken to map any arbitrary size of input data to a fixed length by applying a hash function. The hash function is a mathematical method that calculates hashed value for the given data in the form of strings or digits [10]. Moreover, the hash function is a one-way function meaning that the hash value is computed for the given input, but finding the inverse is not feasible.

Definition 2.1 (Hash Function) Let $H(x)$ denotes a hash function, where x is an input of arbitrary finite length bit strings, and it returns fixed size hash value Y [10].

$$Hash\text{-}Function = \begin{cases} \{0,1\}^* \rightarrow \{0,1\}^n \\ Y = H(x) \end{cases}$$

where $*$ is termed as Kleene star. Calculation of hash function is fast as it renders outputs in linear time. In contrast, the inversion of a hash function is computationally infeasible as it renders the input of the given output in exponential time. Moreover, it is important that the hash function is collision-resistant. To put it another way, if there are two different inputs as x_1 and x_2, the computed hashed value are two distinct output strings [11].
In cryptographic applications, hash functions are implemented in digital signature schemes to deliver data integrity to recognize the altering of the message. Moreover, hashing is useful in both Bitcoin and Ethereum protocols by securing the transactions in different data structures.

2.2 Data Structure

It is vital to understand how the data is organized and stored in blockchain technology. Blockchain transactions operate faster with high security. In blockchains, data are stored in blocks and are immutable. To obtain immutability, it utilizes consensus and cryptographic methods. The data is stored in distributed nodes, and this technology is denoted as “Distributed Ledger Technology (DLT)” [12]. In order to figure out more clearly, we explain how the data structures are utilized and combined to organize blockchains. Data storing is categorized as linear or non-linear. In the linear data structure, the data is collected sequentially on top of each other such that each block is linked linearly. For instance, linked list, arrays, and stacks. In the non-linear data structures, each data block is connected to more than one data unit. Example of this data structure is graphs, trees, and the Merkle tree.

2.2.1 Linked List

The linked list scheme is a sequence of blocks that includes data body and pointer in each block, where the pointer directs to the previous or next block so that data blocks are tight to each other by chains of pointers. There are two blocks exception in the linked list structure, the first block (genesis block), as there is no block before it, and the last block where a pointer is allocated to a NULL value [12].
The differences between blockchain and linked list data structure are as follows. A hash function is used in the blockchain to define the previous block, whereas a linked list utilizes a pointer function to determine the preceding node. Another difference is that tampering and data manipulation is almost impossible in a blockchain, making it a secure network. While in the linked list, data manipulation happens easily. As a structure, blockchain is far more complex than a linked list.

2.2.2 Merkle Trees

The theory of Merkle trees is initiated by Ralph Merkle and patented in 1979 [13, 14].

Definition 2.2 (Merkle Trees) In cryptography, a Merkle tree, also considered a hash tree, is a data structure implemented for securely verifying the data. The Merkle tree is inverted down where the leaf nodes are at the lowest part. While every leaf is labeled by its hashed value of data blocks and non-leaf nodes contain the hashed value of its corresponding child nodes [13].

In order to solve the technical hindrance related to storing and accessing the data in the decentralized networks, Merkle trees are utilized. It resolves by securely sharing and verifying transaction data among peers in the network. Satoshi Nakamoto has implemented the Merkle trees in the Bitcoin blockchain system.

Example 2.3 (Blockchain transaction based on Merkle tree) In figure 2.1 we have a block that contains 8 transactions denoted as $(Tl, Tm, Tn, To, Tp, Tq, Tr, Ti)$ in the form of a Merkle tree.
On each transition, the hash is applied. Consequently, Tl is hashed as $h(l)$ and it

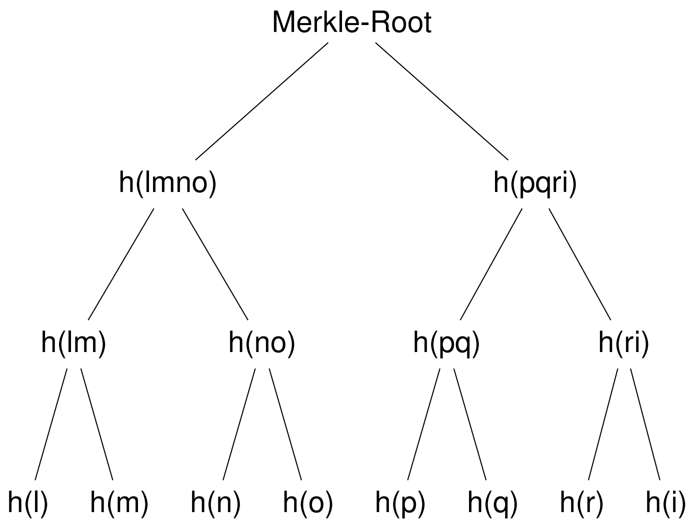

Figure 2.1: Merkle Tree Example

continues till the 8^{th} transaction. We have even nodes; however, it entails appending a node to the end of the block, copying the last node if it is not an even node.
Take transaction hashes into pairs and perform the hash on that pair of hashes as $h(lm)$. Keep doing this process until all transactions meet in the single hash. This single hash is called **Merkle Root**.

Properties of the Merkle tree are as follows:
Verification: Merkle tree includes the property of data integrity. Implementing a Merkle tree indicates that there is no need to get through the whole transaction to verify a particular transaction. Instead, it is only necessary to look through the branch on which the transaction is on.
Merkle root: It is the root of the entire Merkle tree when the Merkle root gets plugged into the block header, which is the part of the Bitcoin block and gets hashed in mining.
Merkle root for block invalidation: By accessing the Merkle root in the block header, if we modify old transactions, that change reflects in the hash of transactions. Then the modification cascades up to the Merkle root and causes changes in the value of the Merkle root, which invalidates the block by tampering with it.
In the next section, we explain proof-of-work (PoW), which is a decentralized consensus mechanism.

2.3 Consensus Mechanism in blockchain

Consensus mechanism is an essential foundation of blockchain with the aim of assisting all the nodes in the network for verification of each transaction. There is a process to validate the real transactions cryptographically. If the transaction is validated for other nodes, they add the new block to the chain of the existing blocks. To guarantee all of this, it is necessary to maintain a consensus mechanism in the network.

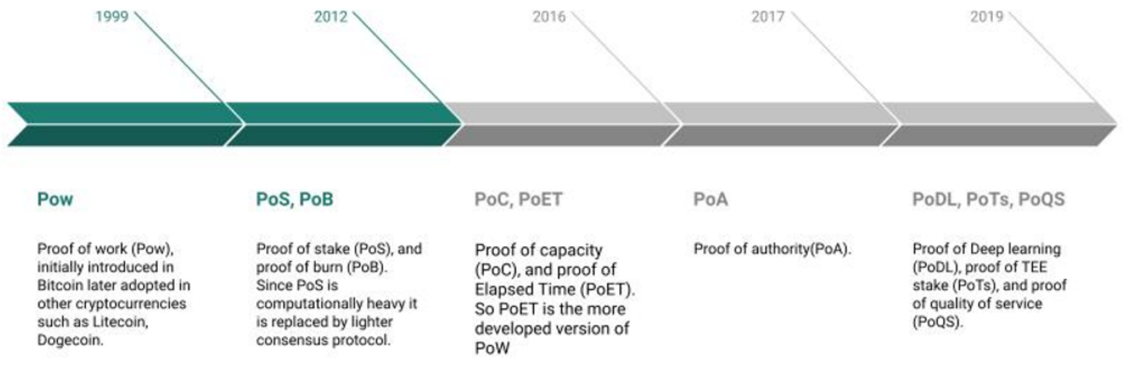
1999
2012
2016
2017
2019
Pow
Proof of work (Pow), initially introduced in Bitcoin later adopted in other cryptocurrencies such as Litecoin, Dogecoin.
PoS, PoB
Proof of stake (PoS), and proof of burn (PoB). Since PoS is computationally heavy it is replaced by lighter consensus protocol.
PoC, PoET
Proof of capacity (PoC), and proof of Elapsed Time (PoET). So PoET is the more developed version of PoW
PoA
Proof of authority(PoA).
PoDL, PoTs, PoQS
Proof of Deep learning (PoDL), proof of TEE stake (PoTs), and proof of quality of service (PoQS).

Figure 2.2: Timeline of consensus algorithm for evolution [1]

In figure 2.2, we illustrate the evolution of the consensus algorithm in the timeline. For instance, some well-known consensus mechanisms are Proof of Work (PoW) utilized in Bitcoin and former Ethereum. Proof of Stake (PoS), Proof of Authority (PoA), Byzantine Fault Tolerance (BFT) are presented in the figure 2.2 [1].

Proof of work model

Proof of work protocol initially is proposed by Dwork and Noar [15]. The project goal is to compute complex puzzles to provide authentication. There are several applications for PoW, for instance, in the bitcoins transaction system, it leverages the PoW scheme. Aside from bitcoin, the PoW scheme is used for identifying denial of a service attacks and also being able to deter spam email. This concept is also termed client puzzle protocol (CPP). In addition, Dwork and Naor proposed the utilization of PoWs function in the computation of square roots modulo a large prime in Fiat Shamir Scheme [16]. Proof of Work (PoW) is a cryptographic algorithm, in which for the verification, the prover delivers the particular level of computational process in a specific timeline [17].

Properties for Proof of Work schemes

To implement the proof of work schemes a pricing function is required for system evaluation. Computing pricing function f is relatively simple while finding inverse of f is mathematically not feasible. Let $F = \{f_s | s \in S\}$ be a family of pricing function. For the PoW scheme the collection of pricing function $F = \{f_k | k \geq 1\}$ is selected where k is a prime number in the length based on difference parameter. The difference parameter is the same as the security parameter in cryptosystems.

We define the shortcuts in the proof of work scheme. A shortcut is analogous to the one-way trapdoor permutation proposed by Diffie and Hellman [18]. By permutation we mean that there exists a bijective mapping $f : S \to S$ where $S \in Z_n$ and by giving x along with public key it is easy to compute the $f(x)$ while given y with public key it is difficult to compute $f^{-1}(y)$.

The square roots modulo a prime p is implemented as given below.

- **Indexing step** Select a prime p of the length of 2^{10} bits.
- **Function defining step** The function $f_p \in Z_p$ and $f_p(x) = \sqrt{x}\, mod\ p$.
- **Verification step** For every $x, y,\ y^2 \equiv x \mod p$ is satisfied.

Another pricing function is based on Fiat Shamir's scheme. A Hash function where the range is the security parameter. The pricing function based on the Fiat Shamir scheme is as follows:

- **Indexing step:**

  1. Suppose $N = pq$ based on two large prime integers as p and q
  2. $y_1 = x_1^2$, $y_2 = x_2^2, \cdots$, $y_k = x_k^2$ be the k^2 modulo N where k is based on difference parameter
  3. h is the hash function $h : Z_N^* \times Z_N^* \rightarrow \{0,1\}^k$
  4. Index s is $(N, y_1, y_2, \cdots, y_k, h)$ of length $k+2$

  The square roots of $x_1, x_2, \cdots, x_k$ are considered as shortcuts

- **Function definition step:** Z_N^* is the domain of f_s. To find z and r^2 the below property is required to be satisfied.

  1. $$z^2 = r^2 x^2 \prod_{i=1}^{k} y_i^{b_i}\ (mod\ N) \tag{2.1}$$

     where each b_i is a single bit in the hash function $h(x, r^2) = b_1, \cdots, b_k$, and $f_s(x) = (z, r^2)$
  2. **Verification step:** For every x, z, r^2 the $h(x, r^2) = b_1, \cdots, b_k$ is calculated and check that equation 2.1 is satisfied.
  3. **Evaluation step** Evaluation of f_s without help of shortcut information is as follows:

     – To find $f_s(x) = (z, r^2)$ we select $b_1, \cdots, b_k \in \{0,1\}^k$ and then compute as follows:

     $$B = \prod_{i=1}^{k} y_i^{b_i} (mod\ N)$$

     – select randomly $z \in Z_N^*$

- Suppose $r^2 = \frac{z^2}{Bx^2}\ mod\ N$ is defined. Repeat this process until the equation of $h(x, r^2) = b_1, \cdots, b_k$ is satisfied

Hence, for evaluating f_s, if there is no shortcut, the number of iterations is 2^k. The verification procedure requires k multiplication and one evaluation of the hash functions. As suggested the best choice for k is 10 [15]. The proposed scheme by Dwork and Noar in 1992 was based on a centralized network, including a secret key. However, for today's applications, decentralized systems are more practical.
One of the well-known PoWs algorithms is hashcash, introduced by Adam Back in May 1997 [19]. In the hashcash method, the sender provides a string whose cryptographic hash begins with zeros. Therefore, a hashcash puzzle (proof of work puzzle) is quite costly to solve, while verifying it has a relatively low cost.

Hashcash

The hashcash cost function is created to compute the cost related to the cryptographic puzzle. Suppose a customer pays electronically for utilizing services on the server.
To verify this transaction, the client calculates a cryptographic puzzle. If the customer is able to solve the determined puzzle, the client allows to generate the token as a confirmation. The token is verified by users in a peer-to-peer network denoted as publicly auditable [19].
Time is an important concept required for the hashcash cost function to check if it is solvable in an acceptable amount of time.
The probabilistic cost functions are :

- Bounded: The number of trials to solve the puzzle to generate valid tokens and the domain of detecting tokens is limited.
- Unbounded: In a practical situation, the probability that a valid token will be generated is escalated more by performing more trials [15].

Hashcash scheme is categorized as follows:

- Interactive Hashcash: In this case, a challenge value is required to compute the token sent by the server. The token is utilized to secure the connection between the client and server.
- Non-interactive Hashcash: It is without server and challenge value. The non-interactive model is applied in the proof-of-work algorithm.

The goal is to check if two sub-string bits are equal for the initial b bits of the hash values which started with $0^k = \underbrace{000.........000}_{\text{k times}}$. To explain the hashcash cost-function, we need to define the following [19]:

1. **s** is a parameter for differentiating between service names.
2. **b** is the total number of bits of the left side of bit strings.
3. $A \equiv_b B$ this notation is for checking if the two sub-bit strings of A, B are equal from the left side (indexed 0 to b) or not. The output is a Boolean value, being true or false.
4. The notation $\|$ is concatenation between two-bit strings.
5. x indicates random bit string.
6. If $H(s \,\|\, x)$ begins with b bits plugged in with zeros then output of V is true.

We explain two algorithms. The first is an interactive base between client and server, and the next is non-interactive.

Interactive Hashcash Cost-function algorithm is as follows:

- Inputs are s, x, and output is Boolean (true, false).
- **Step one:** The $C = Challenge(s, b)$ is calculated by server and send to the user.
- **Step two:** The user explore x as follows:

$$H(s \,\|\, C \,\|\, x) \equiv_b 0^k$$

- **Step three:** If $H(s \,\|\, C \,\|\, x) \equiv_b 0^k$ output is true then users send (s, x) to server
- **Step four:** Server computes $V = Evaluate\, H(s \,\|\, C \,\|\, x)$
- **Step fifth:** If $V = Evaluate\, H(s \,\|\, C \,\|\, x) \equiv_b 0^k$ as output is true, we return V; otherwise, we return false

Non-interactive Hashcash cost-function algorithm is as follows:

- Inputs are as (s, x) and Output is either true or false.
- **Step one:** User computes the $H(s \,\|\, x)$
- **Step two:** User publicises (s, x)
- **Step three:** All nodes in the network calculate $V = Evaluate(s \,\|\, x)$. If $V \equiv_b 0^k$ then it returns true, otherwise false.

The Hashcash is a partial hash function, i.e., there exist many outputs beginning with the same k zero bits. Suppose that clients identified the parameter x that satisfies $H(s \,\|\, x) \equiv_b 0^k$, and x is utilized several times. To avoid using x multiple times, we append parameter time that adds a timestamp to x. Finally, birthday paradox in the case of brute-forcing requires $2^{\frac{k}{2}}$ operations to achieve x. It ensures that the cryptographic puzzle yields either the exact or closest to the output in the polynomial-time [15].

2.4 Digital signature

The Digital Signature is a method to verify to improve the security of communications and transactions. Cryptographic methods create digital signatures where encrypted data exists to demonstrate the signer's authenticity. For instance, Alice sends important documents to Bob, and for this, Alice signs her documents with her private key. For this, Alice needs to use a hash function to obtain a unique hash value where she utilizes her private key to convey the hash digest into a digital signature. Then, Bob needs to verify the digital signature with Alice's public key. Hence, by utilizing the public key and new hash digest, Bob verifies if the digital signature is valid or not.

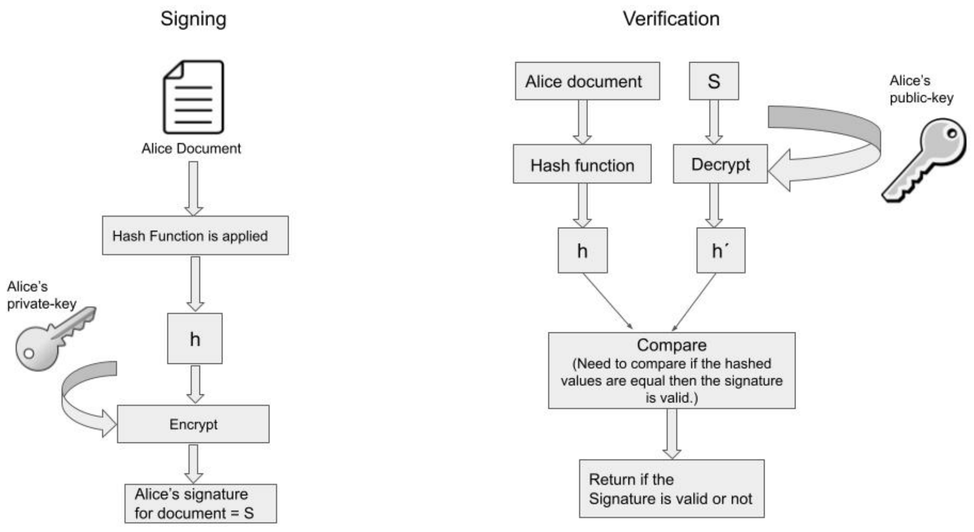
Signing
Alice Document
Hash Function is applied
Alice's private-key
h
Encrypt
Alice's signature for document = S
Verification
Alice document
S
Alice's public-key
Hash function
Decrypt
h
h´
Compare (Need to compare if the hashed values are equal then the signature is valid.)
Return if the Signature is valid or not

Figure 2.3: Illustration of digital signature and verification [2]

Moreover, Digital signatures are utilized in Bitcoin and some other cryptocurrencies for verifying transactions in the blockchain. In other words, digital signatures indicate the ownership of a particular currency and the participation of each person in the specific transaction [2].

Elliptic Curve

Elliptic curve cryptosystem (ECC) was introduced by Nael Koblitz and Victor Miller [20, 21]. The elliptic curve is comparable to discrete logarithm (DL) cryptosystems. Since the elliptic curve is a broad topic, we focus on the useful properties required for cryptography and blockchain. We define the elliptic curve on finite fields, which are important in cryptography.

Definition 2.4 (Elliptic curve on finite fields) Suppose that $p \geq 3$ be a prime. An elliptic curve on a finite field $\mathbb{F}_p$ is a set of all points $(X,Y) \in \mathbb{F}_p$ in the form of the following equation:

$$EC : Y^2 = X^3 + AX + B \tag{2.2}$$

where $A, B \in \mathbb{F}_p$ and the following inequality $4A^3 + 27B^2 \neq 0$ holds in $\mathbb{F}_p$. Together with an imaginary point at infinity $\mathscr{O}$.

$$E(\mathbb{F}_p) = \{(x,y) : x,y \in \mathbb{F}_p \; satisfy \; Y^2 = X^3 + AX + B\} \cup \{\mathscr{O}\}$$

Question

Why we consider $p \geq 3$? Elliptic curve over $\mathbb{F}_2$ is crucial in cryptography since computers are well suited for conducting calculation module 2. However if we consider elliptic curve over $\mathbb{F}_2$, then $\mathbb{E}(\mathbb{F}_2)$ includes at most 5 points which is not practical for cryptography.

Question

Why does $4A^3 + 27B^2 \neq 0$ hold ?
(See in appendix B in Eqn. (B.1))

Example 2.5 suppose EC: $Y^2 = X^3 + 3X + 1$ indicate a finite elliptic curve on $\mathbb{F}_7$. In order to obtain all the points on elliptic curve, it is required to solve the EC equation of our example for all possible x values in the finite filed $\mathbb{F}_7 = \{0,1,2,3,4,5,6\}$. (Since for some integer $n \geq 1$ $EC(\mathbb{F}_n) \approx \mathbb{Z}_n$ [22]). We need to list all values of $X^2 \; mod \; 7$ and all value of $X^3 + 3X + 1 \; mod \; 7$ and then compare based on the table 2.1 to obtain the points as follows:

X	$X^3 + 3X + 1$	X^2
0	1	0
1	5	1
2	1	4
3	2	2
4	0	2
5	1	4
6	4	1

Table 2.1: Example of finding number of points on Elliptic curve

$$EC(\mathbb{F}_7 = \{\mathscr{O}, (0,1), (0,6), (2,1), (2,6), (3,3), (3,4), (4,0), (5,1), (5,6), (6,2), (6,5)\}$$

Elliptic curve discrete logarithm problem

In order to explain elliptic curve discrete logarithm problem (ECDLP) we need to define discrete logarithm problem (DLP) in a group G. For discrete logarithm it is required to explain cyclic group and primitive root definitions.

Definition 2.6 (Cyclic group) Assume that G is a group and $a \in G$. By assuming that G

is a finite group with $order(G) = n$. If the order of element a is similar to order of group G such that $order(G) = order(a) = n$ then element a is the generator of group G. So $G = <a>$ where G denoted as a cyclic group.

Definition 2.7 (Primitive root) Assume p is a prime number and there exist an element $g \in \{1, \cdots, p-1\}$ such that its powers covers every element of $\mathbb{Z}_p^*$ given as:

$$\mathbb{Z}_p^* = \{g, g^2, g^3, \cdots, g^{p-1}\}$$

Such elements g possessing this property are termed as primitive roots of $\mathbb{Z}_p^*$ [2]. Particularly, g is a primitive root of p if and only if g is the cyclic group $\mathbb{Z}_p^*$ generator such that:

$$g^j \neq 1\ (modp)\ for\ j = 1, 2, ..., p-2$$

Question Why $j = p-1$ is not considered?
Since when $j = p-1$

$$g^{p-1} = 1\ (modp)$$

Because of Fermat's little theorem, which indicates as following [23]:

Definition 2.8 (Fermat's little theorem) If p is a prime number and for any integer $a \in \mathbb{Z}$ where $a \neq 0\ (modp)$

$$a^{p-1} = 1(modp) \tag{2.3}$$

Definition 2.9 (Discrete logarithm) Assume p is a prime number and g be a primitive root modulo p. Then the discrete logarithm modulo p of h is based on the smallest positive integer x that is as follows:

$$g^x \equiv h\ (modp)$$

where discrete logarithm is denoted as $\log_g(h)\ (modp)$

Definition 2.10 (Discrete logarithm problem) Let G be a group, p a prime number and $g, h \in G$ where g is a primitive root of p, $1 \leq h \leq p-1$. The problem states that there exists $1 \leq x \leq p-1$ such that $g^x = h\ (modp)$ where $x = log_g h$ is denoted as discrete logarithm of h to base g.

DLP is also generalized to arbitrary finite cyclic groups such as groups on a finite elliptic curve. We explain DLP for our further studies, which ensures the security of cryptosystems.

In elliptic curves, we include a group of points over a finite field $\mathbb{F}_p$. Suppose $P, Q \in$

$EC(\mathbb{F}_p)$ such that it satisfies (Q,P) as follows:

$$Q := \underbrace{P+P+\cdots+P}_{\text{n-additions of point P on EC}}$$

then as per the discrete logarithm in elliptic curves n is the smallest positive integer as $n = \log_P(Q)$.

2.4.1 Elliptic curve digital signature scheme

Elliptic curve digital signature algorithm (ECDSA) was initiated by Scott Vanstone as a solution to the requirement of the National Institute of standards and technology (NIST) [24]. To apply ECDSA as a signature, we require suitable parameters as an elliptic curve which is determined over the finite field $\mathbb{F}_p$. Suppose Alice requires to sign a message m by applying ECDSA. At first, she requires applying the hash function to refer to the message she needs to sign. Then she chooses elliptic curve EC over a finite field $\mathbb{F}_p$. Afterward, she selects a point P on the curve such that the order of P is n. Alice selects randomly her private key as K_{pr} and calculates the public key $K_{pub} = K_{pr} * P$. The parameters of EC over $\mathbb{F}_p$, (n, P, K_{pub}) are publicized. Hence, to create the signature associated with elliptic curves, we are required to sign the signature by Alice and verify the signature by Bob.

- Alice calculates the hash value of message as $z = H(m)$
- Cryptographically secure random number k where $1 \leq k \leq n-1$ is used as a nonce[1] to calculate r and s values.
- K_{pr} is a private key and K_{pub} is a public key employed for signing and verifying the message.

Signing process by Alice are demonstrated as following:

1. Alice generates cryptographically secure random number k where $1 \leq k \leq n-1$.
2. She computes $(x,y) = k * P$ where P is a generator point of the curve.
3. She calculates $r = x \ (\ mod\ n)$ in case $r = 0$ generates other random integer k.
4. Afterwards, Alice computes $s = k^{-1}(z + r * K_{pr})\ (mod\ n)$.

Verification process of signature pair (r,s) by Bob are shown as following:

1. Bob computes $u_1 = z * s^{-1}\ (mod\ n)$ and $u_2 = r * s^{-1}\ (mod\ n)$.
2. Bob calculates $u_1 * P + u_2 * K_{pub}$ as (x,y)
3. The signature is valid if and only if $r = x\ (mod\ n)$

[1] nonce refers to a random number that is devised for particular usage in cryptography and is 'used once' for each iteration.

Why it works?

$$u_1 * P + u_2 * K_{pub} = z * s^{-1} * P + r * s^{-1} * K_{pr} * P = s^{-1} * P(z + r * K_{pr}) = k * P$$

2.4.2 ElGamal digital signature scheme

ElGamal signature scheme was introduced by Taher ElGamal in 1984 [25]. By implementing the ElGamal signature scheme, the verifier is able to check and authenticate the message m and prove that the signer conveys the message through the insecure channel. Assume that $0 < int(m) < p-1$ is the message that is required to be signed.

Key generation

- Select randomly a secret key s where $0 \leq s \leq p-1$.
- Calculate $\beta = g^s \ (mod p)$ where $g < p$ is randomly selected and is the generator of multiplicative group of integers module p which is Z_p^*. p is a large prime number where calculating discrete logarithm modulo p is challenging.
- (p, g, β) is the public key.
- (p, g, s) is the secret key.

Signature generation

The signer requires to sign a message m based on the following steps:

- Select an ephemeral key e randomly such that $0 < e < p-1$ and $e, p-1$ are co-prime.
- Compute $S_1 = g^e \ (mod \ p)$.
- Calculate $S_2 = (H(m) - sS_1)e^{-1} \ (mod \ p-1)$ where H is a collision-resistant hash function.

Following this, the pair (S_1, S_2) is considered as the digital signature of message m.

Verification process

Any signature needs to verified by the receiver of the message.

- Verification of the signature (S_1, S_2) is considered as follows: where $0 < S_1 < p$ and $0 < S_2 < p-1$
- $g^{H(m)} \equiv \beta^{S_1} {S_1}^{S_2} (mod \ p)$

The verifier authorizes the signature on the two conditions mentioned above to be validated. [25].

Correctness

To prove the correctness of the ElGamal digital signature algorithm, we indicate as follows:

$$H(m) = sS_1 + eS_2 \ (mod \ p-1)$$

Such that by implying Fermat's little theorem:

$$\begin{aligned} g^{H(m)} &\equiv g^{sS_1} g^{eS_2} \\ &\equiv (g^s)^{S_1} (g^e)^{S_2} \\ &\equiv (\beta)^{S_1} (S_1)^{S_2} \ (mod \ p) \end{aligned}$$

Moreover, the signer needs to randomly select a different ephemeral key e, following uniform distribution for each signature. Suppose that two messages send with the same secret key. In this case, an attacker is apt to calculate a secret key which is not desirable for the security of the system [25].

2.4.3 Schnorr digital signature scheme

The Schnorr signature scheme is proposed by Claus Schnorr [26]. This method obtains from the extension of Schnorr's identification protocol. This algorithm operation is based on the subgroup of the Z_p^* with the prime order q such that q is a factor of $p-1$.

Key generation

Inputs are γ and λ as security parameters where $\gamma > \lambda$. Following are the steps for key generation:

- Generate random $\lambda \in Z_q$ which is the set of congruence class module prime number q.
- Generate random $\gamma \in Z_p^*$ so that q divides $p-1$ hence $\exists \ r \in Z$ such that $p = qr+1$.
- Select an element $g \in Z_p^*$ with the order of q.
- Choose a random integer as $\alpha \in [1, q]$ and then compute $y = g^{\alpha}$ where it belongs to Z_p^*.
- Assume H is the hash function which is $H : \{0,1\}^* \to \mathbb{Z}_q$.

Accordingly, private key is (p, q, g, α, H) and public key is as (p, q, g, y, H). We consider (p, q, g) as global and public variables meaning that everyone have access to these three variables in the system.

Signing process

Sender requires to sign the message $m \in \{0,1\}^*$ by using private key (p,q,g,α,H).

- Choose a random $k \in Z_p^*$. Where $p \geq 2^{512} bits$.
- Calculate $r = g^k$ which belongs to Z_p^*. Then the signer will concatenate the r with the message m, and $c = H(m \,\|\, r) \in Z_q$. Afterwards obtain the value of s as $s = \alpha c + k \in Z_q$.
- The pair (s,c) are the signature for the message m where $(s,c) \in Z_q$ for $q \geq 2^{140} bits$.

Verifying process

The receiver of a message m with signature (s,c) needs to verify the message indicated as follows, by the public key.

- Compute $v = g^s y^{-c} \in \mathbb{Z}_p$
  Proof of correctness:
  $v = g^{\alpha c + k} y^{-c} = (g^{\alpha})^c g^k y^{-c} = y^c g^k y^{-c} = g^k \in \mathbb{Z}_p$
- Receiver will approve the signature in the condition that $c = H(m \,\|\, v) = H(m \,\|\, g^k)$ so that $r = r'$ otherwise, the signature will be rejected.

With respect to the performance characteristics of the Schnorr digital signature scheme in the signature generation, we require one exponentiation modulo prime p and one multiplication modulo q. The verification process needs two exponentiations modulo a prime number p. Although utilizing the subgroup of Z_p^* with the order of q does not necessarily improve the computational efficiency over the ElGamal scheme, it delivers a smaller signature with the same extent of security compared to the ElGamal protocol.

3 Analysis of various methods in probabilistic micropayments

3.1 Micropayment

Micropayments are considered as a small number of transactions ranging from one cent to a couple of dollars [4]. Micropayments have demand in different areas, including downloadable content such as articles, movies, music, and games [27]. Nevertheless, micropayment systems are confronted with some challenges, as below [28].

- Inquiries concerning micropayments are indispensable to offline payments, which are at risk of double-spending.
- Also, there are transaction fees on the chain, which are considered high for these micropayment values.

In an effort to lower the processing fees in micropayment systems, it is suggested by Rivest and Wheeler to gradually pay off the initial fees for probabilistic micropayments protocol [4, 5]. This system makes it possible for a customer to pay 'M' units of any currency to the vendor by the probability of p. As a result, the seller will obtain full payment with the probability of 1-p.

Example 3.1 Suppose we require to pay an order with PayPal. In the PayPal payment model, anything less than $12\$$ would be better off at the micropayment structure than it would be in the standard payment. The reason for different fee structures per transaction are as follows:

- Micropayment PayPal account fee structure: $5\% + 5\ cents$
- Standard PayPal account fee structure: $2.9\% + 30\ cents$

Assume that the order is $1\$$ we show that the transaction fee difference between standard payment, and micropayment is 23%. As the number of items increase the difference in transaction fees over a year will be huge.

Standard Transaction fees	**Micropayment Transaction fees**
$1 * 0.29 = 0.29 \approx 3\ cents$ $3\ cents + 30\ cents = 33\ cents$	$1 * 0.05 = 5\ cents$ $5\ cents + 5\ cents = 10\ cents$
which is 33% of total fee	which is 10% of total fee

Table 3.1: Example of Transaction Fees in different structures in PayPal

3.2 Probabilistic Micropayment

Wheeler and Rivest introduce the concept of probabilistic micropayments [4,5]. The idea relates to a lottery-ticket protocol rather than sending an exact micropayment amount; a lottery ticket is issued as a currency. The value per each ticket of micropayment is considered the ticket's expected value. Based on the statistic analysis, the expected value's formula is calculated by multiplying attainable results with the probability of every outcome and then summing it up.

$$E[X] = \sum_{i=1}^{n} x_i P(X = x_i) \tag{3.1}$$

Also, each lottery ticket has some face value that is paid out if the ticket wins. Moreover, a winning probability is pre-negotiated between two particular parties.

Overview of Probabilistic Micropayment Protocol

In figure 3.1, an overall overview of probabilistic micropayments is illustrated in the diagram. In the flow diagram, we have Alice, Bob, and John, where Alice sends tickets simultaneously to Bob and John, and none of them wins except the one sent to John. If the single one sent to John wins, it has the opportunity to settle on-chain with that ticket and noted that each party is aware of being paid, on average, a fair amount. The only time that they need to go on-chain is when one of these tickets pays out [3].

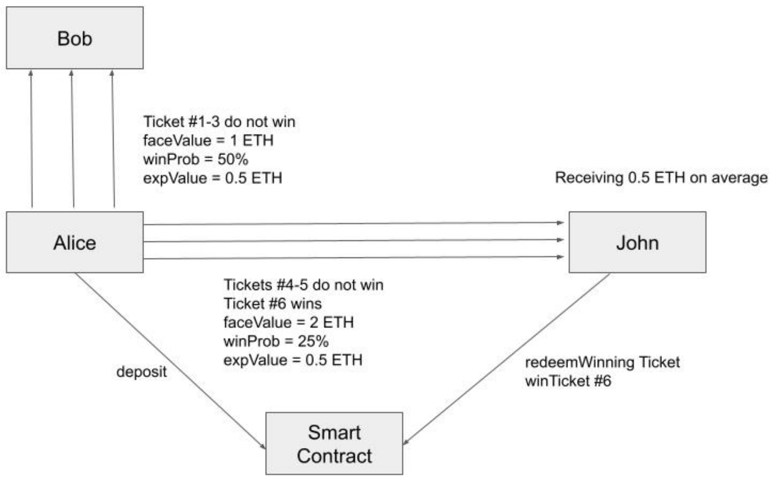

Figure 3.1: Example of Probabilistic micropayment

We consider some important properties of probabilistic micropayments.

- Many to many large volume micropayments.
- Flexibility of ticket parameterization: It is possible to modify the face value and winning probability to adjust to price fluctuation, such as in the Ethereum blockchain [9].
- Proper for high volume and high recipient/sender use cases [3].

With the purpose of lowering the processing and deployment costs, recent cryptography researchers and projects increased interest in decentralized systems for digital currencies [9, 28].

The term trusted third party as a bank is altered by blockchain, and payers construct a deposit with the smart contract containing the payer's escrow fund. Whenever a payer sends a ticket to a payee, at first payer calls for the tickets parameter from the payee, including faceValue (the payout if the ticket wins), winProb (winning probability) with the secret random number created by the payee. Then, the payer utilizes the obtained parameters of tickets and sends the ticket's expected value as $faceValue * winProb$ along with its random number and signature of each ticket. The winning tickets are approved based on the winProb and a random value created by hashing the payer's random number with the payee's secret random number. Assuming that the payee obtains a winning ticket, the ticket with the secret random number and the payer's signature is submitted to the smart contract. The smart contract verifies that the received ticket was won.

Blockchain scalability

The main concept of blockchain scalability is to execute more transactions per second such that the throughput in the blockchain increases. Blockchain is considered a distributed ledger-based technology in which all transactions are stored in a secure and immutable network. It does not depend on a trusted third party such as a bank to validate the transactions executed and add new blocks to the chain. To add a new node and extend the blockchain, network nodes as miners utilize a consensus mechanism to approve which new node generates a valid block, meaning mine faster and accurately so that the new block is attached to the end of the blockchain. For the issue of scalability in the blockchain to enlarge throughput, it is necessary to keep the security and decentralization property of the blockchain. Blockchain scalability solutions are categorized into the following:

- On-chain explanations such as sharding and increasing the block sizes for the issue of scalability in blockchain are provided to modify the blockchain network. Sharding splits the data sets into multiple sets.
- Off-chain solutions such as layer-two blockchains indicate the secondary protocol added on top of the exiting blockchain. They execute micropayment transactions,

particularly outside the blockchain, and store significant transactions such as the final one on the blockchain.

The orchid micropayments protocol focuses more on off-chain solutions for scalability for micropayment transactions [9].

3.3 Analysis of various methods for probabilistic micropayments

This section focuses on the academic literature review and comparison of different probabilistic micropayments. All discussed methods are related to the last three decades. The cryptographic research for micropayments concerned several issues, such as obsolete cryptographic schemes that required some improvements in the system's security, speed of the operating systems, and reducing high transaction fees for online e-payments.

3.3.1 Payword

The Payword method is proposed by Rivest and Shamir and is similar to a credit-based system where it utilizes cryptography to generate unique secure tokens [29]. Hash structures are utilized for developing the token by using the hash chains as one-time passwords [30].
We consider user, vendor, and bank U, V, and B, respectively. Banker provides user validation proving that the user is the authentic customer and provides Payword authorization, which is signed digitally along with a certificate including the banker's name, IP address, and user's name. The user generates Paywords as payment tokens for the transaction. The user selects a random value to repeatedly deploy one-way and collision-resistance hash functions such as 'SHA' to generate the chain.
Transaction steps when the user purchases from a vendor are as given below:

- User conducts a general Payword chain as $y_1, y_2, \cdots, y_n$. The user randomly selects the last Payword y_n from the Payword chain and computes the Payword chain in the reverse order. The user calculates the H-chain function that includes $y_0, y_1, \cdots, y_n$ as input:

$$y_j = H(y_{j+1}) \quad where \quad j = n-1, n-2, \cdots, 0 \tag{3.2}$$

  As we mentioned in 2.1, H is a one-way hash function that is simple to compute for each input entry, however, it is considered a complex problem for inverting with the given output.
- After calculation of the H-chain, the user utilizes its secret key to calculate the

commitment M for the payment.

$$M = \{V, C_U, W_0, D, I_M\}_{SK_U}$$

where D is the current date and I_M is any extra information that might be required. C_U is the Payword certificate is generated by a bank to authorize a user.

- Suppose user provides the payment P to the vendor and the payment contains payword and the number as $P = (y_j, j)$ for $j = 1, 2, \cdots$.
- The vendor verifies the user's certificate C_U and commitment W_0. The user's certificate is provided by the bank and signed by the bank's secret key as $C_U = \{B, U, A_U, PK_U, I_U\}_{SK_B}$. The vendor verifies messages by applying the user's and broker's public keys. Hence, provided pair (y_j, j) from user to the vendor is verified by applying y_{j-1}.
- Vendor repeatedly executes the hash function after obtaining the $P = (y_j, j)$. In each payment process, a hash function is executed by V. This process is suitable for the vendor to aggregate the user's payment in the chain.
- After transaction steps have been confirmed in the system, the vendor delivers goods or services to the user by the user's provided address.

This process is continued for j_{th} payment to a vendor for $j = 1, 2, \cdots$ and including the pair (y_j, j), which is verified by V. This process is suitable for the vendor with the purpose of aggregating user's payment in the chain. Suppose that U conducted the j micropayments and V aggregates all the transactions to make considerable macropayments. Therefore, V is required to store only one payment. For the validation purpose, the bank verifies U's signature of y_0 and calculates how many iterative applications of H are required to map y_j to y_0 and consider that y_0 is the root of payword chain.
However, the payword method has its drawback. Aggregation via the vendor is impossible since each user conducted its H chain operation related to each vendor. Hence, the payword sequence is a user-specific, and vendor-specific payword chain [29].
Suppose a customer decides to buy from several vendors such as V_i; therefore, the user generates multiple payword chains specific to each vendor.

An improved payword method for the purpose of multiple vendors as a new payword is proposed [31]. The general payword method allows users to conduct a single payword chain to pay for several vendors. The general payword scheme consists of the registration stage, transaction stage, and redemption [31].

Suppose that the user U wants to buy some commodities from several vendors $V_1, V_2, \cdots, V_k$. The transaction process for the user and the vendors is given as follows.

- Step 1: User U constructs a general payword chain $w_1, w_2, \cdots, w_n$. Payword chain is computed in reverse order

$$w_i = h(w_{i+1})\ for\ i = n-1, n-2, \cdots, 0$$

- Step 2: User U calculates a payment root R_{V_i} corresponding to vendor V_i as follows:

$$R_{V_i} = h(w_j \oplus (U \parallel V_i))$$

where $\parallel$, $\oplus$ indicate as concatenation and exclusive OR accordingly. Payword w_j is the first unused payword in the sequence of payword.

- Step 3: Commitment M_C for the payment includes below attributes:

$$M_C = \{V_i, C_U, R_{V_i}, D_C, I_M\}_{X_U}$$

- Step 4: The user U forwards the payment P to the vendor V_i. P contains a payword and a number as below:

$$P = (w_t, k)$$

where $k = t - j + 1$

- Step 5: The vendor V verifies the commitment M_C and the certificate C_U, and checks the expiration date D with the user and broker public keys as Y_U and Y_B respectively.
- Step 6: The vendor delivers the commodities to the user if $P'_{V_i} \overset{?}{=} P_{V_i}$. Vendor V_i conduct a hash function operation as follows:

$$h^{k-1}(w_t) = h^{k-2}(w_{t-1}) = \cdots = h(w_{t-k+2}) = w_{t-k+1}$$

As well the vendor V_i calculates

$$P'_{V_i} = h(w_{t-k+1} \oplus (U \parallel V_i))$$

if the equation stay true which means the paywords are conducted by the user U. At the end vendor forwards goods to the user U.

3.3.2 Electronic Lottery Tickets scheme

The electronic lottery tickets method is proposed by Rivest and is based on micropayments the only tickets that banks execute are the winning tickets, and the rest are not handled anymore [4]. In the lottery ticket protocol, hash chains applied are similar to payword. The vendor provides for the user the hash chain root $x = x_0$ and merchant's paywords are denoted as $x_0, x_1, \cdots, x_n$:

$$x_j = H(x_{j+1}) \quad where \quad j = 0, 1, 2, \cdots, n-1 \tag{3.3}$$

After 3.3 equation, the user adds paywords to its commitment (root x_0) and own Hash chain. Presume that the chance of winning is $\frac{1}{1000}$ and formerly the j_{th} payment of the buyer's chain is selected if it satisfies the equation 3.4:

$$x_j \ mod \ 1000 = y_i \ mod \ 1000 \tag{3.4}$$

where y_i is a Payword chain of a new micropayment protocol from the vendor to the buyer. The lottery-based protocol steps are summarized as follows:
Here U, V, and B respectively denote user, vendor and bank.

Steps No.	**Parties**	**Transactions**
1	U → V	Payment request
2	V → U	$W_0$is a root of hash chain
3	U → V	$SIG_u(ticket)$
4	V → U	Goods or services plus W_i if it is winning ticket
5	V → B	$SIG_U(ticket)$

Table 3.2: Summary of lottery ticket scheme

There is a possibility of fraud from the vendor side in the selection part. Because the vendor determines whether a micropayment x_j selected is a winning ticket or not by revealing the y_i. If the payment is not deposited, the vendor rejects the user's expected commodity. However, this behavior is not beneficial from the vendor's point of view since it may be of very little worth.
The electronic lottery protocol has some drawbacks. Processing time is slow due to interaction between vendor and user in the whole chain. There is a possibility of a collision attack in the lottery scheme.
Suppose the vendor does not commit the y_i precedently in a collision attack, and it still consists of winning tickets. Due to this, the vendor keeps storing a vast number of payments since the merchant is required to wait until the winning ticket is determined. Therefore, the merchant cooperates with the external body contrary to the user. This scenario also has the possibility of occurring in the opposite order.

3.3.3 Efficient Coin-Flipping scheme

Coin-Flipping schemes is proposed in 1998 by Lipton and Ostrovsky that compromises the probabilistic polynomial-time user, merchant, and the bank [32]. Properties of coin-flipping include efficiency, fairness, and validation. In this scheme, the executions are done based on a public-key setting, which is validated by a reliable foundation being considered as a banking system. Processing is on circular bases in which each round is categorized into two stages:

1. Pre-processing stage
2. Polynomially bounded sequences (where future output coin flips are defined).

Polynomially bounded functions and sequences play a significant role in the security of cryptographic protocols to assess and categorize the problems concerning the computational complexity of algorithm [33].

Definition 3.2 (Polynomially bounded functions) A function $f \in R$ is polynomially bounded if $\exists\, k \in N$ such that $f(x) \in O(x^k)$.

Definition 3.3 (Polynomially bounded sequences) Assume that s is a sequence of real numbers. Then s is defined as polynomially bounded if and only if $\exists\, k \in N$ such that $s \in O(\{n^k\}_{n \in N})$.

For generating the micropayment by the user to the merchant, the following steps are required. Assume that f is a bijective one-way permutation function significantly affecting cryptographic primitive algorithms.

- Step 1: merchant selects a random number a and then calculates the values on the chain as follows: $b = f(f(f \cdots (a)))$, and merchant sends b to user.
- Step 2: User examines the zero-knowledge proof of the knowledge related to the merchant to check whether it is being rejected in any of the steps [34, 35]. On the condition being approved, the user picks a random number a' then evaluates values on-chain as follows: $b' = f(f(f...(a')))$, then the user sends (a, a') with the signature to the merchant. Moreover, the user provides the merchant a zero-knowledge proof of the knowledge related to a'.
- Step 3: The merchant validates the zero-knowledge proof related to the user and user's signature and the public key; if any section is not valid, the merchant terminates the process.

Here we define the summary of steps related to future output coin-flips:

In the coin-flipping method, the interaction with the bank compared to a lottery-based ticket is less, reducing bank fees for the transactions. Also, it is an anonymous scheme because of the usage of pseudonyms in the algorithm process. Zero-knowledge proofs of knowledge are needed for security, although causing inefficiency in this scheme simultaneously eliminates the double-spending attack. Although it provides the security for all future coin-flips based on the pseudo-random principle for the cases comprising the execution of payments, it is required to merge with other schemes to increase security which is a costly process.

Steps No.	Parties	Transactions
1	U $\rightarrow$ V	Payment request
2	V $\rightarrow$ U	b is a root of hash chain starting from a
3	V $\rightarrow$ M	$ZK(a)$ denotes the zero-knowledge proof of knowledge of a.
4	U $\rightarrow$ V	$SIG_u(a,a')$
5	U $\rightarrow$ V	$ZK(b')$
6	U $\rightarrow$ V	Next b' image with 'get-page' message
7	V $\rightarrow$ U	Next b image with 'here it-is' message
8	V $\rightarrow$ B	$SIG_u(a,a')$

Table 3.3: Summary of output sequence of Coin-flipping scheme

3.3.4 MR2 Scheme

In this section, we present the selective base deposit micropayment scheme that overcomes the issues related to excessive payments for users and interactions in the payments network [36]. MR2 is a non-interactive protocol such that the user sends a micro cheque to the vendor, and the cheque is chosen from a macro deposit. First, in the selective deposit, the bank verifies the signatures. Then, it sends the serial number (SN) of the payable cheque and date to guarantee that there are no exceptional cases such as dates that are out of order or cases when the amount of cheque is excessive. The basic scheme for MR2 consist of the following steps:

- **Setup:** Every user and vendor publicise their public keys for digital signature protocol.
- **Transactions for payment:** User U transfer the payment transaction T by the cheque as $C = SIG_U(T)$ to merchant M.

  Remark 3.4 Cheque C is payable if it holds 3.5 inequality:

  $$F(SIG_M(C)) < s \tag{3.5}$$

  where s is a selection rate. The merchant forwards the cheque and $SIG_M(C)$ to the bank if the cheque is considered payable.
- **Selective deposit:** $maxSN_U$ denotes the maximum number of serial numbers (SN) related to the paycheque of the user. In the initial step $maxSN_U = 0$. If we have a new payable cheque C, the signature of U, and M are verified, then the bank credits from a merchant bank account with the rate of $\frac{1}{s}$ cents.

  If $SN > maxSN_U$ then the user account is debited as $SN - maxSN$ cents via the bank, and the value of a serial number with the maximum number is set as $MaxSN_U \leftarrow SN$ and thereby proving the user by the signature $SIG_M(C)$.

- **Selective Discharge:** In this case, banks record the statistics and reject the transactions when there are exceptional cases. Exceptional cases are when the bank recognizes that the new cheques have the same serial number as the previously executed cheques. Another case is when the date of the cheque is expired compared to other processed cheques. Furthermore, the case when the cheques are more frequently processed for payments than excepted is considered as an excessive amount.

Table 3.4 gives the summary of steps related to the MR2 method.

Steps No.	**Parties**	**Transactions**
1	U $\rightarrow$ V	$Cheque = SIG_C(T)$ where $T = \{Cert_C, pageno., SN, time, Bid\}$
2	V $\rightarrow$ U	goods or services is provided
3	V $\rightarrow$ B	cheque, $SIG_M(C)$

Table 3.4: Summary of MR2 scheme

Example 3.5 (Traffic fees related to the Internet) In this example, we calculate the costs related to the traffic generated for one page, which is added to the price of the page [37]. For traffic cost analysis, we process as follows: The average URL length is 320 bytes, and there exists 6 automatic URLs for each page. Then internet traffic costs as $\$550$ for each gigabyte, and encrypted message appends 30% to the size. In the table 3.5, we calculate the price of traffic which is issued to carry out one payment according to three schemes. The highest traffic cost corresponds to the coin-flipping method.

Lottery ticket scheme	**Coin-flipping scheme**	**MR2 scheme**
$5 \times 6 \times 320 \times 1 = 9600\ bytes$ leads to 0.528 cent/page	$8 \times 6 \times 320 \times 1 = 15360\ bytes$ leads to 0.844 cent/page	$3 \times 6 \times 320 \times 1.3 = 7488\ bytes$ where leads to 0.411 cent/page

Table 3.5: Traffic costs of different schemes

Three probabilistic micropayment methods are compared: lottery-based tickets, coin-flipping, and MR2, which are categorized as centralized. The following features in probabilistic micropayments are necessary to explain for the evaluation.

- **Computation overhead:** It is the number of digital signature's hash functions required to carry out the transaction.
- **Transaction delay:** It is the computational transaction delays involved in the communication channel.

- **Internet traffic Cost:** It represents transaction execution fees.
- **Anonymity:** It describes the level of anonymity in the information shared from a user to a merchant.
- **Double Spending:** It indicates the user utilizing a similar cheque/ticket frequently.
- **Cognition:** It shows how easy it is to understand the structure of a particular scheme.

We compare three different probabilistic schemes according to the features in table 3.6. Because of the feature summary and traffic cost analysis for probabilistic micropayments, we conclude that the coin-flipping method is the least proper method compared to other methods due to higher traffic cost and computation overhead with no improvements in other features compared to the two other schemes. The MR2 method is more suitable for probabilistic micropayments.

Features	**Lottery ticket scheme**	**Coin-flipping scheme**	**MR2 scheme**
Computation Overhead	low	high	low
Transaction delay	low	high	low
Traffic cost	medium	high	low
Anonymity	partial	partial	partial
Double spending	No	No	detected
Cognition	low	high	low

Table 3.6: Features summary for probabilistic micropayments

Probabilistic Micropayments are utilized in a large number of applications. However, processing the probabilistic micropayments consists of high transaction fees that exceed the payment value. However, the small transactions are aggregated into a few larger ones to overcome this issue.

3.3.5 Blind signature scheme based on discrete logarithm

We analyze the blind signature scheme derived from modifying the digital signature algorithm (DSA) based on a discrete logarithm.

Assume we have a user Alice along with a complete overview of her exchange values as V with documents signature pair as $(m, sig(m))$. Any signature scheme such as RSA, ElGamal, Schnorr, and discrete logarithm is called blind if the occurrence of signature pair $(m, sig(m))$ does not affect the possibility of occurrence of V, hence they are statistically independent. The idea of a blind signature was initially proposed by Chaum [7].

The main reason behind the blind signature scheme is to protect customers' privacy in online electronic payments.

Blind signature for modification of DSA

1. Alice selects randomly $k' \in \mathbb{Z}_q^*$
2. Alice chooses g which is a generator of the cyclic group $\mathbb{Z}_P^*$ and calculates $R' = g^{k'} \ (mod p)$

   Remark 3.6 Assume g is generator of the cyclic group $\mathbb{Z}_P^*$ if and only if g is primitive root of p.

   $$g^i \neq 1 \ (mod p) \ where \ i = 1, 2, \cdots, p-2$$

   Based on Fermat's little theorem [2], if $i = p-1$ then $g^{p-1} = 1 \ (mod p)$
3. Alice checks if R' and q are co-prime i.e. $gcd(R', q) = 1$ and then sends R' to vendor Bob. Otherwise she turns back to step 1.
4. Bob checks $gcd(R', q) = 1$ and then he chooses randomly $u, v \in \mathbb{Z}_q$, then he calculates $R = (R')^u g^v \ (mod \ p)$
5. Bob checks $gcd(R, q) = 1$ then he determines the value of $m' = umR'R^{-1} \ (mod \ q)$ and sends m' to Alice. Otherwise if they are not co-prime he returns to step 4
6. Alice conveys $s' = k'm' + R'x \ (mod \ q)$ to Bob
7. Bob verifies the signatures $s = s'R(R')^{-1} + vm \ (mod \ q)$ and $r = R \ (mod \ q)$

The pair (r, s) is the signature for a message m in the blind signature for modification of DSA. To check the validity of signature (r, s) we consider T from variation of DSA scheme. For this reason we explain the modification of DSA as follows and then we prove the verification. In this scheme we consider a prime number p with a prime factor q of $p-1$. The cyclic group generator is denoted as $g \in \mathbb{Z}_P^*$. The prvate key related to signer is selected randomly where $x \in \mathbb{Z}_q$. The public key is computed as $y = g^x \ (mod \ p)$. To conduct a sign a message m which is an integer co-prime to q, it is required to select a random number $k \in \mathbb{Z}_q$ and compute the below parameters.

$$R = g^k \ (mod \ p)$$
$$r = R \ (mod \ q)$$
$$s = km + rx \ (mod \ q)$$

The pair (r, s) consider the signature of message m. With the aim of checking the validity, we compute as below:

$$T = (g^s y^{-r})^{m^{-1}} \ (mod \ p)$$

where m^{-1} is termed as the modular inverse of m modulo q and verify that $r = T \ (mod \ q)$.

[2] For any prime number p and any integer $a \in \mathbb{Z}$ where $a \neq 0 \ (mod \ p)$, $a^{p-1} = 1 \ (mod \ p)$

Proof: To proof the verification of signature (r,s) we utilize T.

$$\begin{aligned} T = (g^s y^{-r})^{m^{-1}} &= g^{(s'rR'^{-1}+vm-xr)m^{-1}} \ (mod\ p) \\ &= g^{k'u+v} = R'^u g^v = R \ (mod\ p) \end{aligned}$$

□

Hence, (r,s) is a valid signature of m and it indicates that $r = T \ (mod\ q)$ why?.
For blind signature scheme the blind factors u and v are selected randomly. Since $m, R', \ and\ r$ are co-prime with q, then the blind factors u and v are identified as follows:

$$\begin{aligned} u &= m'm^{-1}rR'^{-1} \ (mod\ q) \\ v &= (s - s'rR'^{-1})^{m^{-1}} \ (mod\ q) \end{aligned}$$

Hence, we substitute u and v in the following:

$$k'u + v = k'm'm^{-1}rR'^{-1} + sm^{-1} - s'rR'^{-1}m^{-1} = (s - rx)m^{-1} \ (mod\ q) \tag{3.6}$$

Then we replace the $k'u + v$ with 3.6.

Proof:

$$\begin{aligned} r = R'^u g^v = g^{k'u} g^v = g^{k'u+v} &= g^{(s-rx)m^{-1}} \ (mod\ p) \\ &= (g^s y^{-r})^{m^{-1}} = T \ (mod\ p) \end{aligned}$$

□

The blind signature scheme ensures the anonymity of the users, and it has a significant impact on online payment to increase security.

3.4 Analysis of different methods for probabilistic micropayments based on decentralized network

This section describes the probabilistic micropayments schemes based on a decentralized network.

Decentralized Network

A decentralized network distributes information across multiple devices instead of depending on a single central server. Hence, decentralized systems are enticing for micropayments since there is no involvement of a single trusted party, and it helps to overcome the high processing fees [38].
Decentralized transaction systems are noticeably captivating for digital currencies, especially Bitcoin, which deploys peer-to-peer transaction systems [39]. Bitcoin operates

on a distributed public ledger on a blockchain to store all transactions in the blocks, and these proceedings are validated within the peer-to-peer systems. In figure 3.2 we present the comparison of centralized versus decentralized networks by visualization.

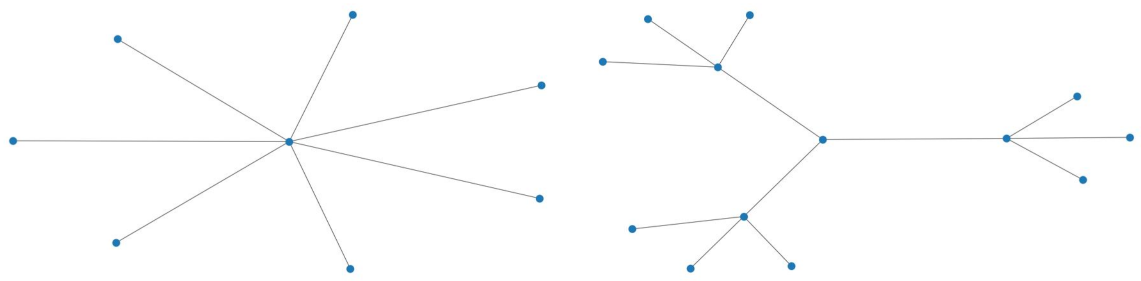
Figure 3.2: Visualization of centralized versus decentralized networks.

Decentralized anonymous micropayments (DAM)

The decentralized anonymous micropayments (DAM) scheme indicates the security of the offline probabilistic payments protocol. This system enables each party to enter a ledger to establish transactions with one peer to another, whether direct or indirect. Moreover, DAM officially cancels the deposits declared as double-spending and remains anonymous across macropayments and nullpayments.

3.4.1 MICROPAY: Abstract model in cryptocurrency system for probabilistic micropayments

In the following section, we study methods that are used in the decentralized network for transferring small amounts, such as $\frac{1}{10}^{th}$ to 1 cent. Various cryptocurrencies have removed the centralized dependency on traditional banks. They have considerably decreased fees related to the large international transaction, though there is no proper solution to reduce fees in micropayment transactions. Therefore, cryptography researchers are analyzing and proposing different methods for micropayment systems that are related to cryptocurrencies.
Authors proposed a lottery-based micropayment method for ledger-based transaction systems [8]. MICROPAY1, MICROPAY2, and MICROPAY3 methods are given by Pass & Shelat. The first method, MICROPAY1, is not feasible in real applications due to limitations in cryptocurrency scripting languages. In the second method, MICROPAY2, authors consider a 'verifiable transaction service' as VTS, a trusted third party. VTS checks if the transactions related to coin tossing win and then provides the escrow funds and signs the release transaction. However, it is confronted with attacks due to VTS dishonesty. In order to improve and overcome this problem, the MICROPAY3 method is proposed. This method includes hidden third parties meaning that VTS should not be activated if the users are honest. Here are the steps in the MICROPAY3 methods by

utilizing invisible VTS.

MICROPAY3

In the MICROPAY3 scheme, the user U transfers the money to merchant M while M obtains the winning ticket along with signed transaction T. The system consist of the following steps in more detail:

- **Penalty Escrow Setup:**

  1. User along with its address as $a = (pk, \pi)$ produce a new key pairs (pk^{pen}, sk^{pen}) where pen is penalty escrow.
  2. λX bitcoins are transferred to penalty escrow as $a^{pen} = (pk^{pen}, \tilde{\pi}_3^{pen})$ along with signed (a, a^{pen}) using public key pk.

     Remark 3.7 Release condition $\tilde{\pi}_3^{pen}(x, a_{pen}, a_2) = 1$ should be satisfied if and only if the transaction is signed by each of U and T.

- **Escrow Setup:**

  1. User generates new key pair (pk^{esc}, sk^{esc}) with its address $a = (pk, \pi^{std})$.
  2. X bitcoins are transferred to escrow address as $a^{esc} = (pk^{esc}, \tilde{\pi}_3^{esc})$ along with signed (a, a^{esc}).

- **Payment Request:**

  1. Merchant selects a random number as $r_3 \rightarrow \{0,1\}^{128}$ in a case of payment request from user.
  2. Generates a commitment as $c \rightarrow Com(r_1, s)$ where s denotes the string utilized to reveal the commitment.
  3. Creates new bitcoin address a^M in which the payments send.
  4. Send the pair (c, a^M) to the payer U.

- **Payment Insurance:**

  1. After agreement, user selects a random string r_2 to send probabilistic payment $\frac{X}{100}$.
  2. User generates signature σ_1 on transaction (a^{esc}, a^M).
  3. And user creates signature σ on transaction related to (c, r_2, a^M, a^{pk}) with respect to public key of escrow pk^{esc}.
  4. User sends the signatures to merchant M, and M verifies that payment and penalty escrows are not spent yet.

- **Claim Prize:**

  1. Merchant sends to user the winning tuple as (x, a^{esc}, a^{M}).
  2. User verifies and signs the respective transaction as (a^{esc}, a^{M}) by secret key sk_3^{esc}.
  3. User sends the signature σ_2 to merchant.
  4. Merchant utilizes respective signatures σ_1, σ_2 for the purpose of spending the escrow a^{esc}.

     Remark 3.8 In a case the user is not able to send verified signature during a particular timeframe, then merchant induces the 'Resolved Aborted Prize Method' [8].

- **Resolve Aborted Prize:**

  1. T obtains a tuple as (x, a^{esc}, a^{M}) where $x = (c, r_1, s, r_2, \sigma)$ and $c = Com(r_1, s)$.

     Remark 3.9 σ is an authenticated signature corresponding to (c, r_2, a^{M}) signed by pk_1^{esc} and if the last two digits of $r_1 \bigoplus r_2$ are 00 then T signs (a^{esc}, a^{M}) with pk^{T}.

  2. Merchant applies both signatures (σ, σ_T) to pay out the escrow a^{esc}.

- **Penalty:**

  1. If for the similar payment escrow a^{esc} two winning lottery tickets are provided by merchant within the partially signed penalty transaction σ_{pen} then T signs the penalty transaction and pays out to the address 0.
  2. T also publishes witness for penalty transaction corresponding to the two winning tickets (a^{esc}, a^{pen}).

Hence in the MICROPAY3, the invisible VTS is utilized. In case a user or a vendor diverges from the main structure, VTS is activated. On the contrary, if the user or merchant is honest the trusted third party is hidden.
MICROPAY3 also has a few drawbacks, such as being unsteady in gaining profits for a recipient. Also, because of the double-spending attack, the penalty escrow fund increases as the number of recipients grows. Furthermore, it is not based on a transferable scheme, which means that in MICROPAY3, the lottery ticket is transferred from the ticket issuer to a beneficiary. For this reason, we need to analyze other probabilistic micropayments methods.

3.4.2 Probabilistic Micropayments with a transferable scheme in blockchain

This section indicates a decentralized probabilistic micropayment transferable scheme based on the proportional fee method [3]. The transferable scheme secures offline payments by using a tamper-proof wallet. Moreover, it reduces the transaction fees for micropayments in the blockchain.

Transferable scheme design steps

- Step 1: Ticket issuer publishes escrow account e in the smart contract, registers it, and confirms that e has been registered in the blockchain.
- Step 2: The ticket issuer publishes the ticket t and forwards it to the user. Then the payee validates that the tickets are from a legitimate wallet. After authentication, the user obtains the tickets and relays the service or product to the payer.
- Step 3: If the obtained ticket matches the requisites for winning, the ticket is transferred to the escrow account e.

The tamper-proof wallet includes a tamper-proof device that does not approve unauthorized transactions such as double-spent tickets. The Keys in the wallet are as follows: sk^{W_x}, PK^{W_x} are pairs of keys in the wallet for personal usage.

- Hashed value of PK^{W_x} is the address related to the wallet owner.
- Secret key (sk^T) is applied to confirm that the ticket is issued and sent from the authentic wallet.
- $cert_T$ is the certificate associated with the private key (sk^T), the wallet owner acquires.

For the escrow setup in the Blockchain network the following steps are required:

1. The issuer X creates a new account request w_X from the wallet $w_x \leftarrow hash(PK^{W_x})$.
2. The creator issues the escrow transaction T_l by transferring β coins from the created account x to the issued wallet address w_X and executing in the network. $T_l \leftarrow Sign(sk^x; x \rightarrow w_x, \beta)$.
3. After verification of T_l and integration into the blockchain network B_i, X is added in the chain and takes B_i. Then T_l, and B_i is forwarded to W_X.
4. Generate the escrow account e as followed: $e \leftarrow Sign(sk^{w_x}; (\beta, h_0, T_0, p.\mu))$ and T_0 as $T_0 \leftarrow Sign(sk^{W_x}; w_x \rightarrow ., \beta)$. Eventually, X transfers the e and T_0 to the Blockchain network.

The overall design of the transferability system is illustrated in the figure 3.3 based on

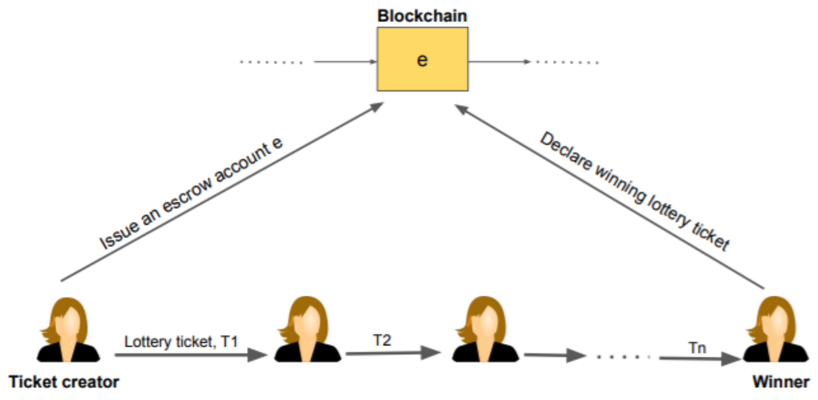

Figure 3.3: Outline of probabilistic micropayment with transferable scheme [3]

the idea of a probabilistic micropayment transferable scheme [3]. For payment with lottery tickets in the blockchain below steps are necessary to follow:

1. The payee Y transfer PK_{W_Y}. Payee Y's wallet termed as W_Y consists of $(sk^{T'}, PK_{W_Y}, sk^{W_Y})$ and payers X's Wallet denoted as W_X includes $(sk^T, PK_{W_X}, sk^{W_X})$.
2. Ticket T_1 is generated by wallet W_X as followed: $T_1 \leftarrow Sign(sk_W; w_X \rightarrow w_Y, e)$. Then signed with private key sk^{W_X} and also signing with the wallet constructor secret key sk^T. Afterwards, wallet W_X sends ticket T_1 and $proof_1$ which is $(proof_1 \leftarrow Sign(sk^T, T_1))$ and $cert^T$ to the payee with wallet W_Y that is generated as hashed value of public key as followed: $w_Y \leftarrow hash(PK^{W_Y})$.
3. Approve and store T_1 and $proof_1$ if and only if the following conditions are satisfied:
   $cert^T$ is trustworthy
   $Verify(PK_T; proof_1) = 1$
   T_1 is verified and valid if payment conditions are satisfied. Then Y responds to W_X with the status as : $Sign(sk_Y; status)$.

If payee Y requires to send the obtained ticket to a different user, the same process is followed [3]. For claiming the winning ticket and avoiding double-spending, we consider the steps given below:

1. Suppose ticket T is the winning ticket ($T \in \{X \mid X\ is\ a\ winning\ ticket\}$) then Y sends T and $proof$ to the contract account.
2. In case T is validated and eligible, the escrow account e records the escrow transaction T_0 by w_Y. Also payee Y from time to time upgrades the local chain and approve that T_0 signed as $T_0 \leftarrow Sign(sk_{W_X}; w_x \rightarrow w_Y, \beta)$ is authentic, and note that β is the lottery winning value.
3. In case T is one of the double-spent tickets which is issued by a double-spending attacker, then the contract account e approves that T is the same unit of digital currency used more than once.

4. In this step, Y establishes the revocation by issuing the revocation transaction as $T_r = Sign(sk_Y; proof, cancel\ T)$, then sends it to insurer termed as Z.
5. Z checks the T_r and refunds to Y for the attacker damage, then issuer creates and commits $T_z = Sign(sk_Z; z \rightarrow y, (1-q)^j\beta)$ to the network.

We define the transferred transaction and winning condition with eligibility as follows:

Definition 3.10 (Transferred transaction) Suppose we consider two tickets in the chain as $T_i = (A \rightarrow B, T_{pre})_X$ and $T_{i+1} = (A' \rightarrow B', T'_{pre})_{X'}$. The tickets are transferred if and only if the proposed properties are satisfied [3]:

- $H(T_i) = T'_{pre}$
- $A = X, B = A' = X'$
- $cert^T_{X'}$ is validated
- multi signatures as $\sigma^W_{X'}$ and $\sigma^T_{X'}$ are also valid.

Then $T_i < T_{i+1}$. If T includes no preceding lottery tickets, the ticket is termed as **genesis** ticket. The genesis ticket T_1 connects to escrow account e where $e < T_1$, and the lottery tickets are shown as follows:

$$e < T_1 < T_2 < ... < T_n \tag{3.7}$$

A is the sender account, and B is the receiver account. Multi signatures σ^W are signed by the key related to the sender's account, and σ^T is signed with the tamper-proof tool to validate the signing device with the issued $cert^T$. In the following definition, we explain the condition that a ticket needs to succeed in order to be considered as a winning ticket.

Definition 3.11 (Winning condition of Tickets) The ticket T_i is considered a winning ticket if and only if the proposed condition is satisfied [3]:

$$win = \{T_n \mid p : H(VDF(h_0 + n \cdot \mu)) < D \ \ \forall n \in N\} \tag{3.8}$$

where $X \& Y \in U$ are termed as payer and payee, respectively. Then, n is the total cardinality of generators of T, and μ is the fixed number that is stated in the escrow account e to calculate the winning ticket. P is the probability to determine whether the ticket is a winning ticket or not. And h_0 is the height of block to determine probability of p which is calculated based on verifiable delay function (VDF) that is the triple denoted as $VDF = (Setup, Eval, Verify)$ [40]. Duplicated tickets are shown by D.

After the winning conditions are satisfied β is obtained, and the ticket
$(T \in \{X \mid X\ is\ a\ winning\ ticket\})$ is transferred. The user who possesses the eligible ticket is able to receive β from the escrow account e.

Definition 3.12 (Eligibility of T_n) The winning ticket T_n is eligible if and only if the condition is given below satisfied:

$$eligible = \{T_{n'} \mid T_{n'} = max(\{T_{n''} \mid n'' \geq n\})\} \tag{3.9}$$

whereas the final winning ticket is the eligible one [3].

There is no benefit in utilizing transferable methods for the issuer while the blockchain fees γ are considered. Since there exists a deduction from the available amount, which is $\beta - \gamma$ the proportional fee scheme is suggested [3].

Proportional fee scheme

Suppose that we consider the transaction fee rate as q. Payer sends a ticket T_i, and in response, the payee transfers goods or services as $(1-q)^i\beta$ to the sender. The transaction fees are $(1-q)^{i-1}q\beta$. Particularly, the respective fee for each payment is as follows:

$$T_{i-1} - T_i = (1-q)^{i-1}q\beta \tag{3.10}$$

Moreover, to calculate the profit, we deduct expenditure from the income $(income - expenditure)$. So the ticket T_i is eligible to be considered as a winning ticket, and the profit is:

$$\beta - (T_i + \gamma) = (\beta - ((1-q)^i\beta) + \gamma) = \beta(1-(1-q)^i) - \gamma \tag{3.11}$$

The transaction fees related to lottery tickets are reduced by applying a proportional fee scheme on the blockchain.

3.4.3 Efficient and double-spending resistance micropayment method for cryptocurrencies

Authors have demonstrated a new approach for micropayment in the transactions of cryptocurrencies on blockchain [41]. This method is based on the transition commitment, which succeeds in being more cost and time effective. The following cryptography functions are implemented in this scheme.

- **Chameleon hash function:** Chameleon hash function is a collision resistance hash function based on public and private key cryptography. With respect to input as message m and r a random string the chameleon hash function creates a hash value as $CHash_U(m,r)$ with the properties given below:
  - Collision resistance: There is no algorithm associated with the public key pk_U that perceives distinct pairs of (m_1, r_1) and (m_2, r_2) such that

$CHash_U(m_1, r_1) = CHash_U(m_2, r_2)$.

- Trapdoor collisions: There is an algorithm associated with the private key sk_U that can perceive distinct pairs of m_1 , r_1 and m_2, r_2 such that $CHash_U(m_1, r_1) = CHash_U(m_2, r_2)$.
- Uniformity: For every value of m it obeys the uniform distribution on $CHash_U(m, r)$ where r is chosen randomly based on uniform distribution.

- **Digital signature:** In this method, ElGamal signature or Schnorr signature schemes are possible to implement. This signature scheme is related to the public and private keys where the signer signs the massage m with its private key sk as $\sigma \leftarrow Sign_{sk}(m)$. The verification is indicated as $Verify_{pk}(\sigma)$ where the input message m along with the signature σ is considered. For verification output it returns true or false to indicate the validity of signature.

The efficient micropayment scheme

In this section the details regarding the transaction scheme for cryptocurrencies such as bitcoin is presented.

- KeyGen[3](1^λ): Considers the input as a security parameter λ. Pair of private and public keys as (x,y) such that $x \in Z_q{}^*$ and $y = g^x(\ mod\ p)$ where $g \in Z_p{}^*$. We utilize $Z_p{}^*$ and cyclic sub group of order q such that $\exists\ k \in Z$ satisfying $p = kq+1$.
- Sign(sk, m): It considers the input as $sk = x$ along with message m and utilizing the Schnorr digital signature scheme, returning the signature as σ.
- Verify(σ, pk, m): It considers input as Schnorr signature σ, message m and public key $pk = y$.Then based on the Schnorr scheme it returns true or false.
- CHashGen(pk, r, m): The chameleon hash function is created with respect to discrete logarithm [42]. It is followed by chameleon commitments scheme as given below:

  - Setup: Setup process is same as KeyGen that we explained in micropayment scheme.
  - Chameleon function: For any value of $m \in Z_q{}^*$ we choose a random value $r \in Z_q{}^*$ and determine the chameleon hash function as:

$$CHash_y(m,r) = g^m y^r\ mod\ p \tag{3.12}$$

- CollComp(sk, pk, r, m, m'): Chameleon hash function includes collision resistance property i.e. for any given $m, m', r \in Z_q{}^*$ and private key $sk = x$ there exists a value $r' \in Z_q{}^*$ where $CHash_y(m,r) = CHash_y(m', r')$. It is derived by solving the

[3] A process of generating key pairs in cryptography. The program that is utilized to generate keys is called a keyGen.

equation:

$$m + xr = m' + xr'(\ mod\ q)$$

In order to verify, we require to use a public key as $pk = y$.

- MicroCoinGen(b, sk, pk, n): For the creation of a chain of microcoins, the user's transactions are required so that the bitcoin is altered into n chain of microcoins and spent separately based on a single recipient. For inputs we consider bitcoin value b and private key for each i as $sk_i = x \in Z_q{}^*$ along with public key $pk_i = y = g^x(\ mod\ p)$. For every $(j = 1, 2, ..., n)$ it returns n chain of microcoins as $\{c_j\}$ denoted as c. Finally, we include Aux as auxiliary information specific to the scheme parameters.

To create a micropayment chain for the n coins the owner of the bitcoin b chooses the committed integer n and proceeds as below:

$$\begin{aligned}
&h_n \leftarrow Sign_x(H(b, y, n)),\\
&h_{n-1} \leftarrow H(h_n),\\
&h_{n-2} \leftarrow H(h_{n-1}),\\
&\vdots\\
&h_1 \leftarrow H(h_2)
\end{aligned}$$

Then the holder of bitcoin calculates the chameleon hash function as follows:

$$\begin{aligned}
&b = CHash_y((n, h_1), r')\\
&where\ \ CHash_y((n, h_1), r') = g^{n_1} y^{r'}\ mod\ p
\end{aligned}$$

The micropayment coins where each indicates the same monetary unit are as follows:

$$c_n, c_{n-1}, \cdots, c_1, c_0$$

Suppose the holder of microcoins requires to transfer k coins to the receiver R. In that case, it is necessary to conduct the process based on the commitment transaction and consider the input as microcoins c_k and auxiliary information as Aux. The commitment of transactions:

$$T_{i+1} \leftarrow Sign_x(H(c_0, pk_{i+1})) \tag{3.13}$$

where T_{i+1} is located in the ledger and publicised in the bitcoin network for which miners perform the verification process. In this method, the minimal transaction is one microcoin. Suppose that U transfers k microcoins, then U user forwards c'_k to receiver R. Then, R verify the validity by following process:

1. Compute hashing chain:

$$\begin{aligned} &c'_{k'-1} \leftarrow H(c'_{k'}),\\ &c'_{k'-2} \leftarrow H(c'_{k'-1}),\\ &\vdots\\ &c'_1 \leftarrow H(c'_2) \end{aligned}$$

2. Checks if below equality holds:

$$CHash_y((n,c'_1),r') \stackrel{?}{=} CHash_y(b,r)$$
$$g^{n \cdot c'_1} y^{r'} = g^b y^r \ (mod\ p)$$

   Hence, if the equality holds the algorithm returns true, i.e.

$$\begin{aligned} &c'_1 = c_1,\\ &c'_2 = c_2,\\ &\vdots\\ &c'_k = c_k \end{aligned}$$

   The receiver of the following transaction updates the ledger and publicises to all other users in the network.

Assume that the owner of the coins considered as U_i, it transfers continuously for remaining $k'-k$ microcoins where $k' > k$ so that U_i transfers $c'_{k'}$ to receiver R.

$$\begin{aligned} &c'_{k'-1} \leftarrow H(c'_{k'}),\\ &c'_{k'-2} \leftarrow H(c'_{k'-1}),\\ &\vdots\\ &c'_k \leftarrow H(c'_{k+1}) \end{aligned}$$

Then R checks if the equality hold as $c'_k = c_k,\ c'_{k+1} = c_{k+1}, \cdots,\ c'_{k'} = c_{k'}$ so that the algorithm will return true. Then the U_i updates the ledger and publicise the information for all users. The micropayment commitment is created based on the main chain, and it indicates a proof that the transaction of the micropayment is executed.

4 Attack analysis in probabilistic micropayment methods

Probabilistic micropayments (PM) is a necessary and beneficial scheme for many-to-many payments by solitary funding deposit authorized by universal payment aggregation. Since winning tickets determination is provably fair, a recipient does not care that a ticket from any individual sender does not pay out because, on average, it will be paid fairly throughout many tickets across many senders.

The main challenge to be addressed with PM is securing it against double-spending. Unfortunately, in real applications double-spend attack is not fully prohibited due to the following reasons:

- Senders have a single deposit for multiple recipients.
- Payments are offline. In other words, participants in the network do not put on hold for transactions to confirm on-chain, hence, guaranteeing global consensus on the availability of funds.

One of the solutions that are provided to overcome the problem of double-spending is to force senders to have a non-spendable penalty escrow that creates tickets along with the sender's funding deposit [8]. If the sender is involved in double-spending, penalty escrow will charge it. As a result, a sender would lack an economic motivation to try double-spending if the additional benefit it gains is less than the economic loss encountered by deducting its penalty escrow amount. For probabilistic micropayment security against double-spending attacks, the following condition is proposed:

- A recipient needs to agree on payments for services from an anonymous sender safely.
- A recipient needs to attain simultaneous payments from different senders.
- A sender needs to pay any recipient regardless of whether the sender has been a customer to the recipient or not.

However, the PM double-spend security protocol still requires more research and development. The double-spending attack steps are illustrated in figure 4.1, as follows [43]:

- (a) It is a state in the blockchain when the attack starts. It is a leaf block that does not include corresponding transactions yet.
- (b) On the left side, there is a branch known to the network. This branch consists of a payment transaction to the seller and two confirmations. The seller transfers the product or service, and simultaneously the attacker detects one block in the other private branch credited to himself.

- (c) In this state, if the attacker makes its branch longer, one block is already known to the network and pays himself as the attacker is confirmed in the network.

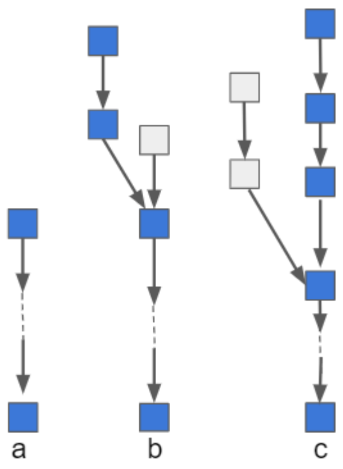

Figure 4.1: Double-spending outline [3]

With respect to the transferable scheme that is discussed in chapter 3, the double-spending attack is indicated as followed: [3]:

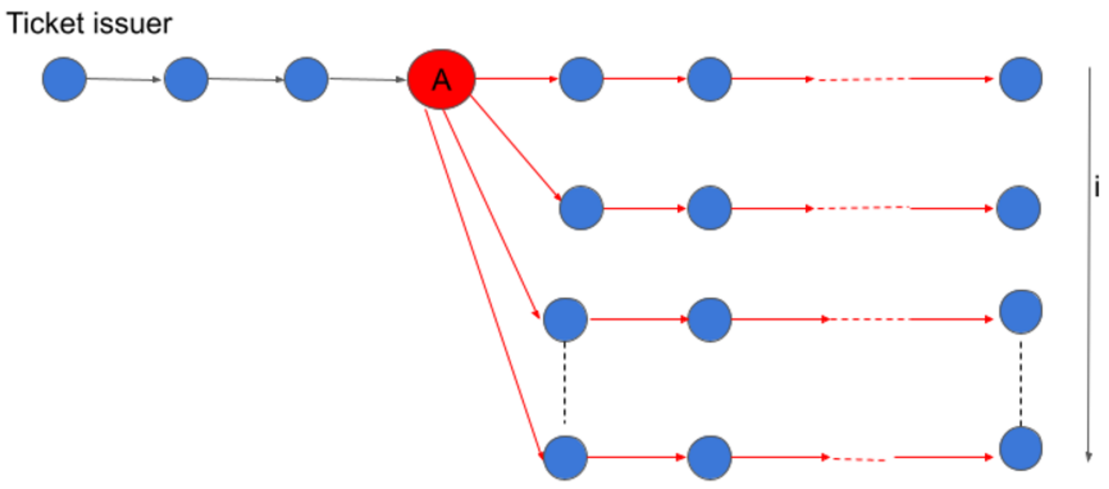

Figure 4.2: Double-spending Attack scheme [3]

In figure 4.2, for the transferable method, the attacker profit is $i \cdot \beta$ where β is the winning amount and the i is the number of duplicated tickets. In the transferable scheme, the adversary attacker interferes, breaks the k-tempered proof wallet, and obtains tickets with various wallet addresses. Those issued along with double-spending attacks are termed duplicated ones.

4.1 Detection method for double-spending attacks

We analyze two methods to identify the double-spending attacks and obtain the adversary address. If the attacked address is revealed, it is broadcasted among all users in the network. Afterward, it is discarded by all users in the chain. So in this situation, the adversary does not obtain utility from a single block attack unless this attack surpasses the corresponding cost related to k-tamper-proof wallet.

Theorem 4.1 (Fork[4] detection) *The double-spending attacks are able to be detected by fork detection.*

In order to prove this theorem, it is essential to define the fork of a series of transactions.

Definition 4.2 (Fork of series transaction) Suppose we conducted two series of transactions which begins with the same escrow account $e \prec ... \prec \theta$ and $e \prec ... \prec \theta'$. This series of transactions are considered as fork if and only if it satisfies conditions simultaneously $\theta \not\prec \theta'$ and $\theta' \not\prec \theta$ [3].

Proof: Suppose two forked series of transactions exist as θ and θ'. Assume that users check the eligible tickets which are registered in the blockchain. The user who owns the θ' reports the adversary attack as double-spending. Then the adversary address is confirmed by the recent similar prefix of θ and θ'. □

Theorem 4.3 (Collision Recognition) *Assume that u is the cardinality of users in the transfer scheme network. So, in the collision detection, E_d is the upper bound for the below inequality [3]:*

$$E_d \leq \sqrt{\frac{u}{e}\beta} \tag{4.1}$$

where E_d is the statistical expected amount of utility for double-spending attacks.

Proof: Hypothetically assume that each user has αu addresses. However, in real time scenario, user's number of addresses are considered as an exponential distribution rather than uniform. Suppose $p(i,u)$ denotes the probability of selection of the particular user address that is selected i-times by user u. The probability is as follows:

$$p(i,u) \approx 1 - e^{-\frac{i^2}{2u}} \tag{4.2}$$

Presumably, the attacker double-spends the i tickets along with the greatest possible extent value of β for each ticket. The attacker's expected utility amount is as follows:

$$E_d < \max_i \{ i\beta \cdot (1 - p(i,u)) \} \tag{4.3}$$

[4] a situation that "occurs when two or more blocks have the same block height."

Hence, the E_d is maximum $\sqrt{\frac{u}{e}}\beta$ in a case $i = \sqrt{u}$ $\square$

Therefore in the transferable scheme, the adversary attack as double spending is distinguished by all users. In this case there is no benefit for attacker as long as the fees of breaking a single tamper-proof wallet surpasses the maximum expected value which is obtained by the attack. If the ϕ is considered as the fees obtained on breaking k-tamper proof wallet, the attacker receives no advantage under the below condition:

$$\sqrt{\frac{u}{e}}\beta < \phi \tag{4.4}$$

4.2 Double-spend attack with concurrent of eclipse attack in blockchain

The security of transactions in Bitcoin is analyzed based on a double-spending attack with the occurrence of an eclipsed attack. A mathematical model associated with a blockchain mining steps are given, in which the model is utilized to conduct an economic assessment of double-spend attack in the blockchain [44].
Eclipse attack is represented by Heilman et al. for peer-to-peer Bitcoin network [45]. In an eclipse attack, an attacker blocks a peer's view from the peer's neighboring connections by isolating a peer node to connect only to the attacker node. Suppose that the attacker blocks the merchant's view in the blockchain by a double-spend attack with eclipsing possibility. The following assumptions are considered for the attacker model:

- q is a fraction of total mining power. An attacker's mining power is $0 < q < 0.5$. If $q \geq 0.5$, the Bitcoin is not able to secure any transactions since the attacker controls the most mining power.
- New blocks are created every 10 minutes. Mining follows Poisson point process consisting of properties of independence and Poisson property, which indicates the occurrence of mining continuously and independently at a constant average rate.
- Cost associated with eclipse attack is not considered in this model. Also, we assume that the attacker does not conduct a denial-of-service attack concerning honest miners.

To identify Bitcoin security for eclipse-base attack, economic break-even point [5] is calculated. The Break-even point is calculated as $R - C = 0$, where R is the revenue generated by the company and C is associated with the cost incurred in a company. A merchant forwards goods or services when the paying transaction reaches z-blocks in the blockchain.

[5] The break-even point in the economy is the point where a company's revenue equals the cost incurred in a company.

Remark 4.4 A random variable is denoted as X which is a function from possible outcomes Ω mapped to a measurable space E. The random variable function associates values to every experiment's outcomes. So the probability of $X : \Omega \rightarrow \mathrm{E}$ obtaining a value in a measurable set $S \subseteq \mathrm{E}$ is given as: $P(X \in S) = P(\{\omega \in \Omega \mid X(\omega \in S)\})$ [46].

Suppose that X_i^q is an exponential random variable, i.e. $X_i^q \sim exp(\beta)$ where $\beta = \frac{10}{q}$. X_i^q indicates the time duration for an attacker with mining power of q to mine the respective i^{th} block.

$$X = \sum_{i=1}^{z} X_i^q \tag{4.5}$$

The time taken by an attacker to mine in order to obtain block z is given as 4.5.
$X \sim \Gamma(z, \frac{10}{q})$, since $X_i^q \sim exp(\beta)$ is an exponential distribution. Exponential distribution is subcategory of gamma distribution with rate of β which is the inverse of scale parameter, and shape of $\alpha = z$. The probability density function (PDF) associated with gamma distribution with shape-scale parameters is $f(x; z, \beta) = \frac{x^{z-1}e^{-x/\beta}}{\beta^z \Gamma(z)}$. Evaluating the cost of an attack with a mining power of q in duration x along with deadline d is denoted as $C(x; d, q)$, which is computed as given below:

$$C(x; d, q) = \begin{cases} \dfrac{qxB}{10} \; for\, x \leq d \\ \dfrac{qdB}{10} \; for\, x > d \end{cases}$$

where B denotes block reward, and blocks are mined in a frequency of 10 minutes. The reason to evaluate the attack cost is to determine the utility gaining of an attacker compared to being an honest miner. Assume that $g(x; z, \beta)$ is a probability density function (PDF), and $G(x; z, \beta)$ is its cumulative distribution function (CDF). We compute expected value of opportunity cost as given below:

$$\begin{aligned} E[C(X; d, q)] &= \int_0^\infty C(x, d, q) g(x; z, \beta) dx \\ &= \int_0^d \frac{qxB}{10} g(x; z, \beta) dx + \int_d^\infty \frac{qdB}{10} g(x; z, \beta) dx \\ &= \frac{qB}{10} \int_0^d x \cdot \frac{x^{z-1}e^{-x/\beta}}{\beta^z \Gamma(z)} dx + \frac{qdB}{10}(1 - G(d; z, \beta)) \\ &= \frac{qB}{10\beta^z (z-1)!} \int_0^d x^z e^{-x/\beta} dx + \frac{qdB}{10}(1 - G(d; z, \frac{10}{q})) \\ &= \frac{qB}{10\beta^z (z-1)!} \frac{x^z e^{-x/\beta}}{-\beta} - z \int x^{z-1} e^{-x/\beta} dx + \frac{qdB}{10}(1 - G(d; z, \frac{10}{q})) \end{aligned}$$

Hence, we continue with integration by parts till we obtain the expected value as given below:

$$\begin{aligned} E[C(X; d, q)] &= \frac{qB}{10}[\frac{10z}{q} G(d; z+1, \frac{10}{q})] + \frac{qdB}{10}(1 - G(d; z, \frac{10}{q})) \\ &= \frac{qdB}{10} + zBG(d; z+1, \frac{10}{q}) - \frac{qdB}{10} G(d; z, \frac{10}{q}). \end{aligned} \tag{4.6}$$

The probability of earning benefit from the attack is $P(X \leq d) = G(d; z, \frac{10}{q})$. The attacker gains revenue R if the mining time is less than the deadline for mining each block. For example, revenue is given as below:

$$R(x;d) = \begin{cases} v \; for \; x \leq d \\ 0 \; for \; x > d \end{cases}$$

The expected revenue associated with the eclipse attack is calculated as $E[R(X;d)] = vG(d; z, \frac{10}{q})$. The break-even point concept is utilized to compute v. The expected break-even point takes place when the expected value of revenue is equal to the expected value of cost; hence we substitute the expected value of revenue with the cost as given in 4.7:

$$E[R(X;d)] - E[C(X;d,q)] = 0$$

hence it follows that:

$$\begin{aligned} v &= \frac{E[R(X;d,q)]}{G(d;z,\frac{10}{q})} \\ &= \frac{E[C(X;d)]}{G(d;z,\frac{10}{q})} \\ &= \frac{\frac{qdB}{10} + zBG(d;z+1,\frac{10}{q}) - \frac{qDB}{10}G(d;z,\frac{10}{q})}{G(d;z,\frac{10}{q})} \end{aligned} \tag{4.7}$$

With utilizing this model, we compute the revenue of an attacker in the case of double-spend attack with occurrence of eclipse attack in Bitcoin network. The important parameters for success of attacker are as below:

- Attacker mining power q
- Depth of block that is to be mined
- Confirmation deadline d which is announced by merchant.

The security of transaction execution in a blockchain network against a double-spend with the eclipse attack increases logarithmically with the z depth of the block, consequently adding more profit to an attacker. For the merchants that set the long confirmation time rather than one confirmation with the same mining power $q = 10\%$, the merchants prohibit the system from an attacker.

4.3 Adversary attacks

In the adversary attacks, we define two types of malicious attacks as Types I and II. In Type I, the attacker is the owner of the crypto coin who plans to spend more than its sufficient fund during the transactions. In Type II, the adversary attack is related to

the receiver of the crypto transaction, who has the purpose of declaring more than the amount obtained.
The efficient and double-spending resistance micropayment method for cryptocurrencies which is discussed in chapter 3, is secure against Type I and Type II adversary attacks [41].

Theorem 4.5 *The efficient micropayment scheme for cryptocurrencies is secured in case of a Type I adversary attack if the security of the Chameleon hash function and digital signature is guaranteed.*

Proof: Let us suppose Type I adversary as A and simulator as B. The adversarial attack is modeled as an interaction between A and B.
Setup: Algorithm[6] $KeyGen(1^{\lambda})$ is requested by simulator B with the purpose of issuing a pair of public and private keys (pk, sk). The pair of keys as (x, y) is selected from multiplicative cyclic subgroup as $x \in Z_q^*$ and $y = g^x(\; mod \; p)$ where $g \in Z_p^*$. Then key pair (x, y) is forwarded to adversary A.
Query: The adversary A is able to issue as many microcoins as possible. Suppose we consider bitcoins as b and set of microcoins as $Z = (c_1, c_2, \cdots, c_u)$ where c_i termed as a whole set of microcoins initiated from b.
Challenge: The simulator B chooses a set of microcoins from $Z = (c_0^*, c_1^*, \cdots, c_n^*)$ and forwards the set to A. Adversary A win the challenge, if A is able to replace a valid microcoin as $c_{n'}$ such that $n' > n$. The probability of adversary winning is trivial since it is supposed that blockchain is secured and integrating into the ledgers, then

$$T_{i+1} \leftarrow Sign_x(H(c_0, pk_{i+1}))$$

transactions are recorded into the ledgers. Hence, it is impossible to mutate the committed value n even though the private key is recognized. While the set of microcoins $(c_0^*, c_1^*, \cdots, c_n^*)$ is possible to generate, minors still eliminate it since the commitment for transactions T_{i+1} is already documented in the immutable ledgers. □

Theorem 4.6 *The efficient micropayment scheme for cryptocurrencies is secured against Type II adversary attacks. Suppose micro coins c_i are stored in the network since it is mathematically impossible for A to build a valid microcoin c_j hence the security is ensured.*

Proof: Suppose Type II adversary as A and simulator as B.
Setup: In the setup it is followed the process based on $KeyGen(1^{\lambda}$. It is generated the public and private key pairs as (x, y) where $x \in Z_q^*$ and $y = g^x(\; mod \; p)$ where $g \in \mathbb{Z}_{\shortmid}^*$ sends it to adversary A.
Query: The adversary is able to create at most $n - 1$ queries to simulator B. Since the queries are created in an order so even if a microcoin is not queried, it is still considered

[6] In 3.4.4 the method is discussed.

a valid microcoin. The adversary A is able to gain multiple set of microcoins from different bitcoins such as $(c_0, c_1, \cdots, c_{l_1})_{b_1}$, $(c_0, c_1, \cdots, c_{l_2})_{b_2}$, ... denoted by Z.
Challenge The simulator B chooses a challenge for a microcoin set as $(c_0^*, c_1^*, \cdots, c_n^*)_{b_i}$ from the query set Z and forward it to adversary A. A is able to return a valid microcoin c_l' such that $l' \geq l_i$ if $c_{l_i}^* = H(c_l')$. Based on the definition of hash function given on chapter 2, it is infeasible for adversary A to calculate the inverse of a hash value. Hence the probability of winning the challenge by A is trivial.

$\square$

5 Conclusion

This thesis has undertaken a detailed analysis of different probabilistic micropayments systems based on centralized and decentralized networks. We have also compared the methods based on particular features such as cost-saving and time-efficiency. Our study indicates that probabilistic micropayments provide flexible payment methods, increasing cryptographic security and reducing the network's transaction costs.
According to the evaluation of comparison among Lottery tickets, coin flipping, and MR2 schemes, we conclude that the MR2 and lottery tickets are almost the same in the computation time, with partial anonymity of low risks. The coin-flipping method is not proper for micropayment since it has high computational and transaction costs. Since the MR2 scheme has less traffic cost, it delivers the message authentication with no overcharging fees. Hence, MR2 proves to be an important scheme for the case of micropayments in centralized networks.

The comparison between lottery tickets, coin flipping, and MR2 schemes are as follows:

- MR2 schemes include a selective deposit approach ensuring that the customer is not overcharged. While in the case of coin-flipping, there exists a probability that the customer may be charged more than the actual value.
- Based on the complexity of the payments scheme, the user is required to identify the hashing process for the lottery ticket method. The customer needs to know about the zero-knowledge proof of the information, hashing process, and signatures in the coin-flipping scheme. While in the MR2 scheme, it is only necessary for the buyer to identify signatures.
- In the lottery tickets and coin-flipping methods, there is no double-spending attack since the generated tickets include the customer data as the root of the hash chain. However, in the MR2 scheme, there is a possibility of double-spending attacks. Since the user submits the same cheque to different sellers by using the same serial number again, and bank detects this issue later if the same serial number is already deposited from the bank or when the serial number and the timestamp of the cheque are expired.

The MR2 scheme is the proper solution in the case of centralized networks. It is necessary to conduct more research from the commercial and usability point of view and usability purpose. Besides those, to improve the payword scheme, we analyze the general payword scheme and blind signature method, which delivers better results for transaction fees.
In the case of decentralized network implementation in the blockchain system, we scrutinize different methods based on transferability and proportional fee schemes. Then,

we analyze the micropayments in cryptocurrencies, such as bitcoin utilizing chameleon hash functions and a commitment scheme. The MICROPAY3 protocol, a ledger-based transaction system, is derived from lottery tickets. The orchid payment protocol is influenced by the MICROPAY3 protocol as a second layer solution to reduce transaction fees in the Ethereum blockchain.

Summary of different methods for micropayments and probabilistic micropayments

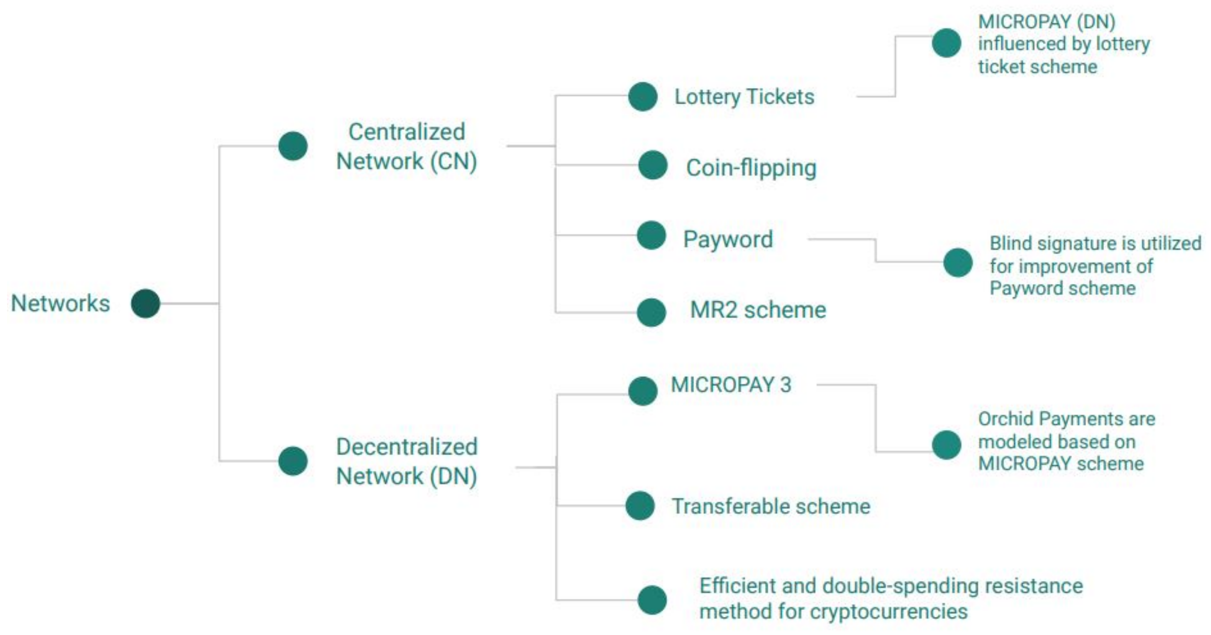
Networks
Centralized Network (CN)
Lottery Tickets
MICROPAY (DN) influenced by lottery ticket scheme
Coin-flipping
Payword
Blind signature is utilized for improvement of Payword scheme
MR2 scheme
Decentralized Network (DN)
MICROPAY 3
Orchid Payments are modeled based on MICROPAY scheme
Transferable scheme
Efficient and double-spending resistance method for cryptocurrencies

Figure 5.1: Summary of various methods

Future Work

- To improve probabilistic micropayment scheme it requires to analyze on the blind signature.
- Concerning attack analysis, it is important to specify the structure of the penalty escrow fund. Furthermore, it is important to design the PM double-spend security protocol as a future implementation.
- The traffic detection in the payment protocol is ongoing research by utilizing different randomization transformations. In which it is possible to relate with deep learning area for traffic detection in payment systems [9].

Bibliography

[1] Wenbo Wang, Dinh Thai Hoang, Peizhao Hu, Zehui Xiong, Dusit Niyato, Ping Wang, Yonggang Wen, and Dong In Kim. A survey on consensus mechanisms and mining strategy management in blockchain networks. *IEEE*, 7:22328–22370, 2019.

[2] Jeffrey Hoffstein, Jill Pipher, Joseph H Silverman, and Joseph H Silverman. *An introduction to mathematical cryptography*, volume 1. Springer, 2008.

[3] Taisei Takahashi and Akira Otsuka. Probabilistic micropayments with transferability. *European Symposium on Research in Computer Security*, pages 390–406, 2021.

[4] Ronald L Rivest. Electronic Lottery Tickets as Micropayments. *International Conference on Financial Cryptography*, pages 307–314, 1997.

[5] Dan Wheeler. Transactions Using Bets. *Security Protocols Workshops*, 1996.

[6] Theodor Holm Nelson. Complex information processing: a file structure for the complex, the changing and the indeterminate. In *Association for Computing Machinery: Proc. 20th National Conference*, pages 84–100, 1965.

[7] David Chaum. Online cash checks. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 288–293. Springer, 1989.

[8] Rafael Pass and Abhi Shelat. Micropayments for Decentralized Currencies. *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 207–218, 2015.

[9] David L Salamon, Gustav Simonsson, Jay Freeman, Brian J Fox, Brian Vohaska, Stephen F Bell, and Steven Waterhouse. Orchid: enabling decentralized network formation and probabilistic micro-payments. *Orchid Labs, Tech. Rep., Available: https://www. orchid. com*, 2018.

[10] Paul C Van Oorschot, Alfred J Menezes, and Scott A Vanstone. *Handbook of applied cryptography*. CRC press, 1997.

[11] Joseph H. Silverman Jeffrey Hoffstein, Jill Pipher. *An introduction to mathematical cryptography*. Undergraduate texts in mathematics. Springer, 1 edition, 2008.

[12] Mohammad Jabed Morshed Chowdhury, Alan Colman, Muhammad Ashad Kabir, Jun Han, and Paul Sarda. Blockchain versus database: A critical analysis. In *2018*

17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE), pages 1348–1353. IEEE, 2018.

[13] Ralph Charles Merkle. A digital signature based on a conventional encryption function. In *Conference of the Theory and Application of Cryptographic Techniques*, pages 369–378. Springer Berlin Heidelberg, 1987.

[14] Ralph Charles Merkle. *Secrecy, authentication, and public key systems.* Stanford university, 1979.

[15] Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. In *Annual International Cryptology Conference*, pages 139–147. Springer, 1992.

[16] Cynthia Dwork, Moni Naor, and Hoeteck Wee. Pebbling and proofs of work. In *Annual International Cryptology Conference*, pages 37–54. Springer, 2005.

[17] Markus Jakobsson and Ari Juels. *Proofs of Work and Bread Pudding Protocols(Extended Abstract*. Springer US, 1999.

[18] Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE transactions on Information Theory*, 22(6):644–654, 1976.

[19] PoW function hashcash. `http://www.hashcash.org/`, 1997. Accessed: 10-01-2022.

[20] Neal Koblitz. Elliptic curve cryptosystems. *Mathematics of computation*, 48(177):203–209, 1987.

[21] Victor S Miller. Use of elliptic curves in cryptography. In *Conference on the theory and application of cryptographic techniques*, pages 417–426. Springer, 1985.

[22] Lawrence C Washington. *Elliptic curves: number theory and cryptography*. CRC press, 2008.

[23] Moses Liskov. *Fermat's Little Theorem*, volume 455–457. Springer, 2005.

[24] Scott Vanstone. Responses to nist's proposal. *Communications of the ACM*, 35(7):50–52, 1992.

[25] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on information theory*, 31(4):469–472, 1985.

[26] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In *Con-*

ference on the Theory and Application of Cryptology, pages 239–252. Springer, 1989.

[27] Paula L Hernandez Verme and Ruy A Valdes Benavides. Virtual currencies, Micropayments and the payments systems: a challenge to fiat money and monetary policy? *European Scientific Journal*, 9(19):325–343, 2013.

[28] Alessandro Chiesa, Matthew Green, Jingcheng Liu, Peihan Miao, Ian Miers, and Pratyush Mishra. Decentralized Anonymous Micropayments. *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 609–642, 2017.

[29] Ronald L Rivest and Adi Shamir. Payword and micromint: Two simple micropayment schemes. pages 69–87. Springer, 1996.

[30] Leslie Lamport. Password authentication with insecure communication. *Communications of the ACM*, 24(11):770–772, 1981.

[31] Ching-Te Wang, Chin-Chen Chang, and Chu-Hsing Lin. A new micropayment system using general payword chain. *Electronic Commerce Research*, 2(1):159–168, 2002.

[32] Richard J Lipton and Rafail Ostrovsky. Micro-payments via efficient coin-flipping. *International Conference on Financial Cryptography*, pages 1–15, 1998.

[33] Hiroyuki Okazaki and Yuichi Futa. Polynomially bounded sequences and polynomial sequences. *open access: De Gruyter open*, 2015.

[34] Mihir Bellare and Oded Goldreich. On defining proofs of knowledge. In *Annual International Cryptology Conference*, pages 390–420. Springer, 1992.

[35] Oded Goldreich and Yair Oren. Definitions and properties of zero-knowledge proof systems. *Journal of Cryptology*, 7(1):1–32, 1994.

[36] Silvio Micali and Ronald L Rivest. Micropayment Revisted. *Cryptographers' Track at the RSA Conference*, pages 149–163, 2002.

[37] Ahmed Kosba, Andrew Miller, Elaine Shi, Zikai Wen, and Charalampos Papamanthou. A review of internet payments schemes. Proceedings of the Australian Telecommunication Networks and Applications Conference (ATNAC'96), 1996.

[38] Paul Baran. On distributed communications networks. *IEEE transactions on Communications Systems*, 12(1):1–9, 1964.

[39] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Decentralized Business Review*, 9:21260, 2008.

[40] Dan Boneh, Joseph Bonneau, Benedikt Bünz, and Ben Fisch. Verifiable delay functions. In *Annual international cryptology conference*, pages 757–788. Springer, 2018.

[41] Fatemeh Rezaeibagha and Yi Mu. Efficient micropayment of cryptocurrency from blockchains. *The Computer Journal*, 62(4):507–517, 2019.

[42] Hugo Krawczyk and Tal Rabin. Chameleon hashing and signatures. 1998.

[43] Meni Rosenfeld. Analysis of hashrate-based double spending. *arXiv preprint arXiv:1402.2009*, 2014.

[44] George Bissias, Brian Neil Levine, A Pinar Ozisik, and Gavin Andresen. An analysis of attacks on blockchain consensus. *arXiv preprint arXiv:1610.07985*, 2016.

[45] Ethan Heilman, Alison Kendler, Aviv Zohar, and Sharon Goldberg. Eclipse attacks on {Bitcoin's} {Peer-to-Peer} network. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 129–144, 2015.

[46] Moshe Zukerman. Introduction to queueing theory and stochastic teletraffic models. *arXiv preprint arXiv:1307.2968*, 2013.

[47] Mathematical Cryptography elliptic curves in cryptography. `https://web.northeastern.edu/dummit/handouts.html`. Accessed: 20-03-2022.

Appendix A: Implementation: Python Code

https://git.hs-mittweida.de/amirian/master-thesis

A.1 Probabilistic Micropayment scheme's: Payword

```
import hashlib
class payword_initial:
    def __init__(self,seq):
        self.seq = seq

   #payword creation by user
    def payword_init(self):
        self.seq.reverse()
        w1 = self.seq[0]
        w2 = self.seq[0]
        w3 = self.seq[0]
   # h is a hash function and three different hash functions are
   applied.
        MD5 = []
        SHA256 = []
        SHA512 = []
   # not including the root W0 in the range function.
        for _ in range(0,len(self.seq)-2):
            c1 = hashlib.md5(w1.encode()).hexdigest()
            c2 = hashlib.sha256(w2.encode()).hexdigest()
            c3 = hashlib.sha512(w3.encode()).hexdigest()
            MD5.append(c1)
            SHA256.append(c2)
            SHA512.append(c3)
            w1 = c1
            w2 = c2
            w3 = c3
        return MD5,SHA256,SHA512
 # number of occurance and applying hash functions on the payword
   chain based on each input.
if __name__=='__main__':
    x = int(input())
    L = []
    for _ in range(x):
        y = input()
        L.append(y)
    obj = payword_initial(L)
    print(L)
    print(obj.payword_init())
```

Listing A.1: Payword creation by User

A.2 Merkle Tree

```python
from hashlib import sha512

class Merkle_tree:

    def build_merkle_tree(self,leaves):
        self.leaves = leaves
        if len(self.leaves)==1:
            return sha512(self.leaves[0].encode()).hexdigest()

        else:
            N = []
            for i in range(0,len(self.leaves),2):
                x = sha512(self.leaves[i].encode()).hexdigest()
                y = sha512(self.leaves[i+1].encode()).hexdigest()
                N.append((sha512((x+y).encode())).hexdigest())
            if len(N)==1:
                return N[0]



        return self.build_merkle_tree(N)

#Adjust the blocks of hashes until we have an enen No. of items in
    the blocks
#This entails appending to the end of the list as the last entry
#To do this we use power function to determine if the given list is
     power of 2 and be even no. of items.


if __name__=='__main__':
    x = int(input())
    L = []
    for _ in range(x):
        y = input()
        L.append(y)
    try:
        for i in range(0,int(x/2)):
            if pow(2,i)==len(L):
                break
            x = i
        obj = Merkle_tree()
        print('\n',obj.build_merkle_tree(L))
    except:
        print('\n Total number of elements should be a power of
    2.')
```

Listing A.2: Merkle Tree

A.3 Elliptic Curve Addition operation

```
# Program for performing addition operation over an elliptic curve
import math

# Class of functions required to perform the addition operation
class elliptic_curve_operations:

    #1 Function to check whether number is prime or not
    def prime_check(x):
        if x > 1:
            z=int(math.sqrt(x))
            for i in range(2,z+1):
                if x%i!=0:
                    continue
                else:
                    return False
        else:
            return False
        return True

    #2 Function to perform addition operation
    def Elliptic_curve_operations(self,p,a,b,points):
        self.p = p
        self.a = a
        self.b = b
        self.points = points
        if 4*pow(self.a,3)+27*pow(self.b,2)!=0:
            if elliptic_curve_operations.prime_check(self.p):
                EC = EllipticCurve(GF(self.p),[self.a,self.b])
                if len(self.points)==1:
                    x = (int(points[0][0]),int(points[0][1]))
                    if x in EC:
                        return EC([int(self.points[0][0]),int(self.
    points[0][1])])
                else:
                    point1 = (int(self.points[0][0]),int(self.
    points[0][1]))
                    point2 = (int(self.points[1][0]),int(self.
    points[1][1]))
                    if point1 in EC and point2 in EC:
                        SUM=EC([int(self.points[0][0]),int(self.
    points[0][1])])+EC([int(self.points[1][0]),int(self.points
    [1][1])])
                    if len(self.points)>2:
                        for i in self.points[2:len(self.points)]:
                            x = (int(i[0]),int(i[1]))
                            if x in EC:
                                SUM+= EC([int(i[0]),int(i[1])])
                    return SUM
            else:
                return 'Since p is not prime power it does not
    create Galois Field.'
        else:
            return 'Since 4a^3 + 27b^2 = 0, elliptic curve is not
    considered.'

# passing the variables in the form of input to create an elliptic
    curve over a Galois field and performing the addition operation
if __name__=='__main__':
    p = int(input('Enter a prime number for creation of GF : '))
    a = int(input('Enter a in elliptic curve y^2 = x^3+ax+b : '))
    b = int(input('Enter b in elliptic curve y^2 = x^3+ax+b : '))
    n = int(input('No. of points for addition operation : '))
    points = []
    for _ in range(n):
        point = tuple(input().strip().split())
        points.append(point)
    obj = elliptic_curve_operations()
    print(obj.Elliptic_curve_operations(p,a,b,points))
```

Listing A.3: Elliptic Curve

A.4 Decentralized vs.Centralized visualization

```python
import networkx as nx
import matplotlib.pyplot as plt
#N is degree
class graph_cons:

    def graph_centralized(self,N):
        self.N=N
        L = []

        for i in range(1,self.N+1):
            L.append(i)
        G = nx.Graph()
        G.add_nodes_from(L)
        for j in range(2,len(L)+1):
            G.add_edge(1,j)
        nx.draw(G)
        plt.show()


    def graph_decentralized(self,N,M):
        self.N=N
        self.M=M
        L = []

        for i in range(1,self.M+1):
            L.append(i)
        V1=[]
        a = self.M+1
        for i in range(2,self.M+1):
            V2=[]
            for k in range(len(self.N)):
                for j in range(a,a+self.N[k]):
                    if L.index(i)==k+1:
                        V2.append(j)
                    a=a+self.N[k]
            V1.append(V2)
        G = nx.Graph()
        G.add_nodes_from(L)
        for j in L:
            G.add_edge(1,j)
        for k in L[1:len(L)]:
            for j in V1:
                G.add_nodes_from(j)
                for i in j:
                    if L.index(k)==V1.index(j)+1:
                        G.add_edge(k,i)
        nx.draw(G)
        plt.show()

if __name__ == '__main__':
    M=int(input())
    n=input().strip().split()
    N = []
    for i in n:
        N.append(int(i))
    obj= graph_cons()
    #obj.graph_centralized(N)
    print(obj.graph_decentralized(N[1:len(N)],M))
#Calling methods separately instead simultaneously for generating
   different figures.
```

Listing A.4: Decentralized vs.Centralized visualization

Appendix B: Mathematics

B.1 Elliptic Curves: Weierstrass Form for Singular & Non-singular Curves

Motive is to study cubic curves in the plane [47]. An elliptic curve over a field K is the curve in the Weierstrass form as given below:

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \tag{B.1}$$

$a_1, a_2, ..., a_6$ are appropriate coefficients in K where in cryptography we restrict our attention to the field of integers modulo p.
Suppose that we reduce Weierstrass equations to the form as given below:

$$\begin{gathered} y' = y + (\tfrac{a_1}{2})x + (\tfrac{a_3}{2}) \\ x'' = x + (\tfrac{a_2}{3}) \\ (y')^2 = (x'')^3 + A(x'') + B \end{gathered}$$

hence the elliptic curve equation in the form of $y^2 = x^3 + Ax + B$ is denoted as a reduced Weierstrass equation, which is more flexible for computation.
Repeated root in the polynomial $x^3 + Ax + B$ means that it includes the root in common with its derivative $3x^2 + A$ where replacing the x in polynomial result in $\Delta = 0$.

- A curve is called singular if and only if its discriminant $\Delta = -16(4A^3 + 27B)$ is equal to zero, i.e., the polynomial $x^3 + Ax + B$ includes a repeated root, and the elliptic curve $y^2 = x^3 + Ax + B$ is singular. Also, the factor of -16 is added to avoid denominators in the Δ expression.
  - Singular points are the points where the curve is non-differentiable. In some cases, for the real field, it occurs when $x^3 + Ax + B$ has a double root where the curve crosses itself and is denoted as a node.
  - The other case is when a singular point occurs in the origin $(0,0)$ where polynomial $x^3 + Ax + B$ has a triple root as $(A = B = 0)$.
- A curve is categorized as non-singular if the roots are distinct.

The elliptic curves have properties related to non-singular curves where $\Delta \neq 0$.

Erklärung

Hiermit erkläre ich, dass ich meine Arbeit selbstständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und die Arbeit noch nicht anderweitig für Prüfungszwecke vorgelegt habe.

Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht.

Mittweida, im May 2022