Application of SHA256 Hashing Algorithm in Blockchain-Based Trustless Transaction

Faza Fahleraz 13516095
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13516095@std.stei.itb.ac.id

***Abstract*—Transactions of any kind has been the backbone of our economy since the earliest civilizations. Therefore, it is only fitting for us to find yet better ways to transact. In it's early form, transactions are made between parties solely based on trust in each other. As time progressed, more and more values are at stake in each transactions. In order to reduce risks, these trust between parties are put on more centralized bodies such as banks and governments. But since all the transactions data are stored in centralized locations, this also presents a huge vulnerability of a breach that may lead to large-scale fraud. With this situation in mind, a new method of transaction is gaining popularity. This method, called blockchain, differs from it's predecessors in which it does not require trust to be put anywhere besides mathematics. This paper will discuss how a cryptographic hash function, called SHA256, enables this new form of transaction.**

***Keywords*—cryptographic hash functions, SHA256 algorithm, blockchain, trustless transaction.**

I. Introduction

A normal, centralized transaction requires the participating parties to put trust in a bank, organization, or any other centralized bodies to verify the transaction. These centralized bodies also keeps track of all past transaction in some form of a ledger. This system makes it easier for transacting parties to be sure that their transaction is safe and recorded in the ledger —hence, universally acknowledged. However, this situation also makes it easier for anyone looking to hack into these ledger because it's all located in only a handful of centralized locations.

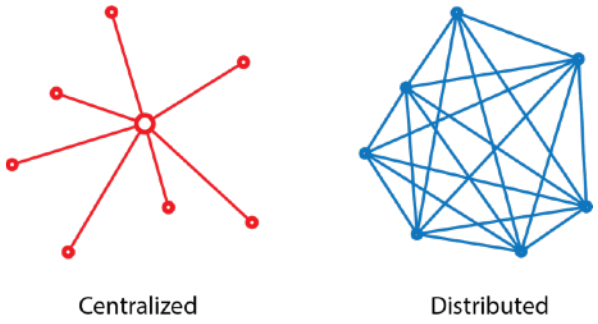

Figure 1.1 Two graphs comparing a visualization of centralized systems (red) and distributed systems (blue).

A trustless transaction, on the other hand, does not require any centralized bodies to verify and record transactions in a ledger. In order to achieve this, instead of having one centralized ledger, a system should have many distributed ledgers that are all the same and equally and universally acknowledged. These ledgers can be kept by each transacting parties or any groups the transacting parties choose. Every transaction can be recorded in each one of these ledgers but every ledgers in the system also have to record this transaction.

The first problem is how to ensure the validiy of each entry in the ledger without any centralized authority. This problem is addressed by using a technology called blockchain. In a blockchain, transaction entries are grouped into many blocks that chains together to create a complete ledger—hence the name blockchain. Each entries in a blockchain is given a 'digital signature' that ensures it's validity. The underlying key to making a digintal signature is by using a form of cryptographic hash functions, in this case it is SHA256.

The second problem is how to tell all other ledgers—in this case blockchains—in the system if a new transaction has been recorded in one of those ledgers. The system also has to keep all of these blockchains the same and if there are any conflicts, resolve them. A way to make this work is by using another technology called distributed blockchains.

In a distributed blockchain system, there are many blockchains kept by each transacting parties or any group the transacting parties choose. In each occuring transaction, it's record is added to a blockchain before being verified and broadcast to all other blockchains in the system. This verification process takes a very large amount of computational power and acts as a 'proof-of-work'. This 'proof-of-work' tells other blockchains that it is safe to add this block to their existing chain of blocks, therefore updating them and all blockchains are kept the same.[1]

As you can see, this 'proof-of-work' is the basis that keeps the integrity of the system. Just like the 'digital signature' concept, the key that enables this is also the SHA256 hashing algorithm. In this paper, we will discuss how such algorithm makes this system of distributed blockchains works.

As a side note, this system is already widely implemented in the form of many cryptocurrencies. That is, currencies that entirely rely it's existence on a distributed blockchain. The first and the most popular decentralized cryptocurrency is Bitcoin. In the last five years alone, Bitcoin has grown it's value by over 79.000 percent to over 9.000 US dollars per one Bitcoin. It's creation was proposed in a 2008 paper titled 'Bitcoin: A Peer-to-Peer Electronic Cash System' by an unnamed author known as Satoshi Nakamoto. That paper was also first to propose the idea of the distributed blockchain.[2]

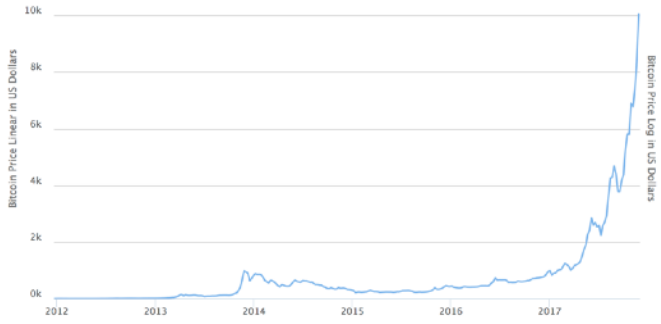

Figure 1.2 The skyrocketing price of Bitcoin in the last five years. (Source: screenshot from https://www.buybitcoinworldwide.com/price at December 2nd 2017)

II. Hash, Hashchains, and Public Key Cryptography

What lies in the heart of a blockchain-based trustless transaction system is a cryptographic hash function called SHA256. This section will explain what such functions are and how they work.

A. Hash Functions

Hash functions are methods to map data of arbitrary size into data of predefined length, called hash or digest. Hash functions has many properties that make them useful. The first is determinism, that is, for a given input value it must always generate the same hash value. The second is uniformity, that is, a good hash function should map the expected inputs as evenly as possible over its output range. That means every hash value in the output range should be generated with roughly the same probability. The reason for this is to minimize the collision happening tho different input values—two different inputs that generate a same value. One of the most common use of this function is in a data structure called hash table.[3]

One of the simplest hash functions is the modulo operation. Any data can be converted into a number (possibly very big), then this number can be divided by a constant and the remainder of that division is the result, or hash. Obviously the result is deterministic because the same string should result in the same modulo value. But, it will result in a lot of collisions since the output range is very small.

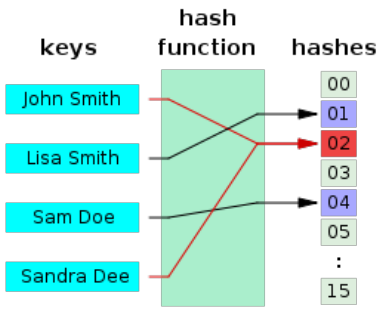

Figure 2.1 A hash function collision. (Source: Wikimedia Commons)

B. SHA256 Cryptographic Hash Function

The traditional hash function is simple but at the same time not widely used outside data structures because people want to have another property to enable it to be used in cryptography that is one-way computation.

One-way computation means it should be easy to compute the hash, but finding any input to the hash function (finding the reverse function) must be very difficult, or better impossible. This form of hash functions are called cryptographic hash functions. Such functions digest the bits of the input data in a very convoluted way to make the reversible computation impossible. The best known cryptographic hash functions are MD5, SHA1 and SHA2. The most common form of SHA2 and the one discussed in this paper is SHA256. Here is an example of the SHA256 algorithm in action:

```
SHA256("hello")=2cf24dba5fb0a30e26e83b2a
c5b9e29e1b161e5c1fa7425e73043362938b9824
```

The result is a 256-bit string shown here in the hexadecimal format.

Because there is no way (at least yet) of reversing the hash. The only way to get the correct input is to guess every possible input combination. In the case of SHA256 that would be, in average, two to the power of 256 guesses before one gets the correct input. This means, with the current speed of processors, it is virtually impossible to reverse a SHA256 hash value because it would take an impossibly long time to guess the correct input.

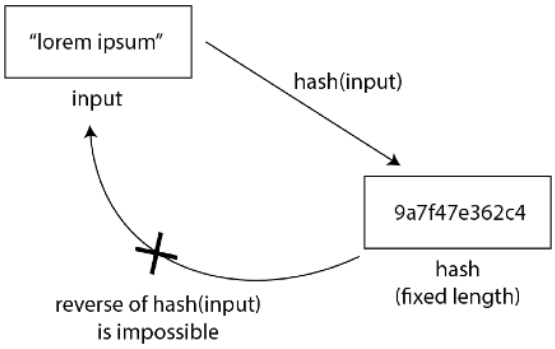

Figure 2.2 Visualization of a cryptographic hash function.

Even more than traditional hash functions, cryptographic hash functions are supposed to be collision-free. This means that it should be impossible (or at least very, very hard) to find two different input that generates the same hash values.

The cryptographic reasoning behind this is very straight forward. Suppose you want to make a program that use a cryptographic hash function to verify a password. First, you should hash your chosen password and save the result in the program. Then, if someone wants to guess your password, the program don't have to compare it to your password to check if they are the same. It just need to hash that person's password and check if the result matches you password's hash. This way, you don't have to store your password in the program, you just need to store the hash. In fact, this is how Linux systems verify your password.

But, what if there are more than one input that can result in the same hash values. If that is the case, that means someone with a completely different password than yours can open your computer with a bit of luck. Of course you don't want this, that is why cryptographic hash functions have to be collision free. With these very powerful properties, much more than classic encryption algorithms, cryptographic hash functions are the heart and soul of modern cryptography.

If a hash value, represented in some form, is fed again into the hash function, a new hash is obtained. If this process is repeated and the results are combined into a sequence of hashes, one obtains what's called a hashchain.[3]

C. *Hashchains*

Hashchain is a sequence of homogeneous piece of data, or blocks, linked together by a hash function. This link is achieved by successively applying a cryptographic hash function to a piece of data and adding new data (blocks) in each hashing step. The first block of data is hashed and the resulting hash is combined with another block of data and then hashed again before combined with another block and hashed again. This process continues until all of the desired blocks of data is hashed.

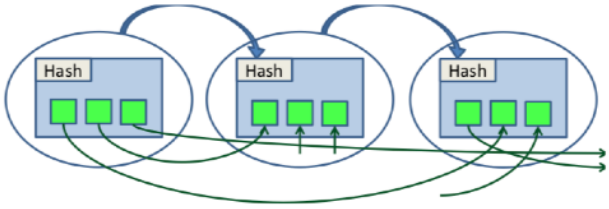

Figure 2.3 A visualization of hashchain.
(Source: Mazonka. Oleg. (2016). "Blockchain: Simple Explanation")

A hashchain has a very interesting and important property. That is no data in any blocks can be modified without affecting the integrity of the subsequent blocks. For example, if the payload of the first block is changed, then the hash of the second block will be changed as well, and hence the hash of the third, and so on. This means that no one can change the data in any blocks without changing the whole hashchain therefore making it invalid. This also means that in order to add a new block to the hashchain one needs to have the hash of the previous blocks.

This property, combined with public key cryptography will become the basis of a single blockchain.

C. *Public Key Cryptography*

Public key cryptography works by using pairs of keys: public keys and private keys. A public key of an individual may be stored publicly and can be seen by anyone. Whereby the private key must be stored privately and should not be revealed to anyone besides the owner. These keys accomplishes two functions: authentication, which is when the public key is used to verify an encrypted messeage that it is sent by the holder of the paired private key, and encryption, whereby only the holder of the paired private key can decrypt the message encrypted with the public key.

The basic idea of public key encryption system is similar to that is of a hash function which is one way computation. In a public key encryption system, any person can encrypt a message using the public key of the receiver, but such a message can be decrypted only with the receiver's private key. For this to work it must be computationally easy for a user to generate a public and private key-pair to be used for encryption and decryption. The strength of a public key cryptography system relies on the degree of difficulty (computational work needed) for a properly generated private key to be determined from its corresponding public key. It has to be very difficult, or better impossible, to get the paired private key from a public key. If this degree of difficulty is achieved, security then depends only on keeping the private key private, and the public key may be published without compromising security.[4]

Let's say there are two person trying to exchange messages privately: Alice and Bob. They both have their own public and private keys. Each person keeps their private key to their own and put the public key public for anyone to see. If Alice wants to send Bob a private message m, Alice first have to encrypt the message using her private key resulting in $Enc_B(m)$ (the subscript B denotes that it is encrypted using Bob's public key which is available to Alice) and send it to Bob. Then Bob should be able to decrypt it using the paired private key that is used to encrypt the message, that is his private key. So $Dec_B(m)$ should produce the original message m. Now, let's say Bob wants to reply to Alice with another message m'. First he needs to encrypt the message using Alice's public key $(Enc_A(m'))$ then sends the message. But unknown to Bob and Alice, there is another person (Charles) tapping their communication and stealing the message. Now Charles wants to encrypt and read the content of the message. Unfortunately for him, because he only knows the public key of Alice and Bob, there is no way he can encrypt the message because that will require him to have the paired public key that is used to encrypt the message which is Alice's private key. As you can see, the communication is kept private.

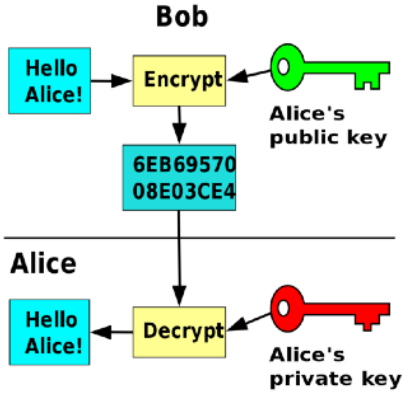

Figure 2.4 A diagram showing the flow of encryption in a public key cryptography.
(Source: Wikimedia Commons)

There are two main scenarios where this system works. The first is when one party wants to send a secure message to another (just like outlined above). The only vulnerability of this method (apart from the theoretical one by cracking the math) is knowing and trusting the public key in the first place. To attack this problem a whole system of hierarchical public keys must exists. For example, the internet addresses this with the https protocol.[4]

The second scenario is quite opposite to the first. Given some data, Alice encrypts it using its private key (or encrypts only hash from the original data) and then publishes both data: the original and encrypted. Anyone knowing the public key can verify that encrypted data is actually computed using the private key that is paired with the public key used for decryption. This verification works as a validation that Alice actually did that. That process is called digital signing and the encrypted part is called digital signature.[4]

III. Blockchain and Digital Signatures

After in previous sections explain the basic concepts that ultimately makes the trustless transaction system possible, this section will further explain in detail how the SHA256 hashing algorith in particular acts as the key in enabling the system of trustless transaction. The the first key use of SHA256 is in making digital signatures in a blockchain. But before that we need to understand what a blockchain is and why digital signatures are at all required.

A. Blockchain

Basically, a blockchain is a ledger that keeps track of transactions. It does so by grouping transaction entries into many blocks that chains together to create a complete ledger. Transactions grouped into blocks for a reason, that is to enable blockchain to be a distributed system. We will discuss that in the next chapter, so now we will focus on how a single block inside a blockchain works.

Inside each block of a blockchain there is a hashchain containing all the transaction entries of that particular block. Therefore, you can think of a blockchain as a hashchain of a hashchain. The transaction entries are structured using a hashchain because of it's property that has been explained earlier. That is, no one can change the contents of any transaction entries without changing the whole hashchain, therefore making it incompatible with the rest of the system. The hashchain used here is a special form called binary-hashchain or Merkle tree.

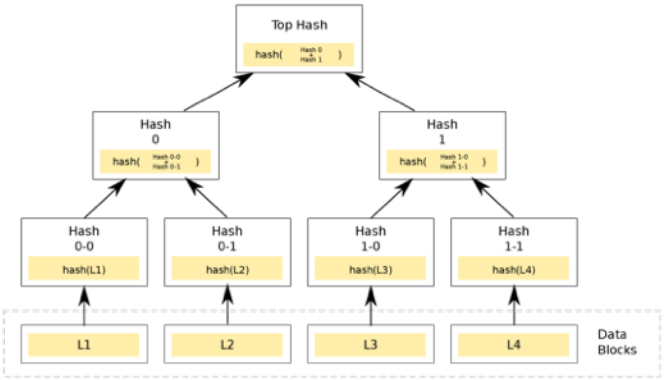

Figure 3.1 A Merkle tree with four data blocks.
(Source: Wikimedia Commons)

The top hash of the Merkle tree is called the combined hash. This combined hash is placed in the header of every block in the blockchain along with a time stamp and a proof-of-work which we will get to in the next section.

Each transaction recorded consist of a payee and a paid party. In order to act as a confirmation from the payee, each transaction entries must have somekind of a signature from the payee to verify it's autheticity. This is where digital signatures is used.

B. Generating Digital Signatures Using SHA256

Let's say Alice wants to pay Bob some money and record it in the blockchain. In order to make sure that Alice really did pay Bob (not that Bob made it up), Alice must put somekind of a signature in the record. In the real world, this may her handwritten signature or her biometrics data. But in this digital world, there is actually a stronger signature.

Just as explained before, public key cryptography can be used to create a digital signature. So, this is what Alice should use. In this case of a blockchain, Alice should sign the transaction by hashing the transaction data with her private key using the SHA256 algorithm. The result will be a 256-bit signature that is unique to this combination of transaction data and Alice's private key.

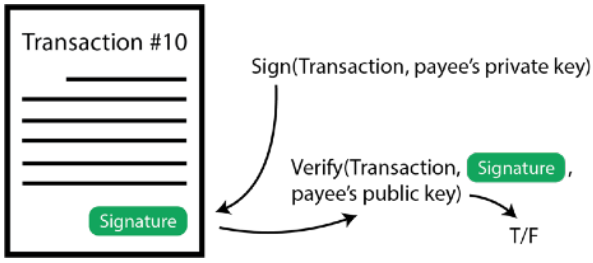

Figure 3.2 Digital signatures in trustless transactions.

Anyone looking to validate this transaction can easily use Alice's public key to confirm that it is indeed Alice that signed that transaction. This can also be used to keep the integrity of the transaction record since a signle alteration in the transaction will create a completely different signature.

This way everyone can be sure that each transaction entries added to the blockchain is indeed done by the transacting parties and the content's integrity is preserved.

Another way to think about this is by visualizing the flow of values—in the case of cryptocurrencies, a coin—in the system. In the original Bitcoin paper[1], the author explains that we can define an electronic coin as a chain of digital signatures. Each owner transfers the coin to the next by digitally signing a hash of the previous transaction and the public key of the next owner and adding these to the end of the coin. A payee can verify the signatures to verify the chain of ownership.[1]

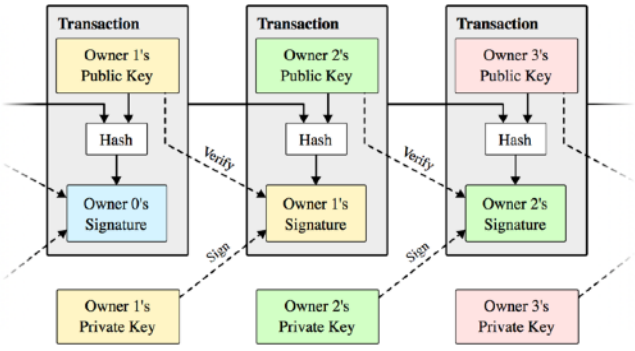

Figure 3.3 A representation of the flow of values as a chain of digital signatures.
(Source: Nakamoto, Satoshi. "Bitcoin: A Peer-to-Peer Electronic Cash System". 2008)

IV. Distributed Blockchains and Proof-of-Work

This section will explain another application of SHA256 algorithm in enabling a trustless transaction system. This involves how the algorithm is used to enable a concept called proof-of-work and why it is needed to keep the integrity of a distributed blockchain system.

A. *Distributed Blockchains*

In a distributed blockchain system, there are many blockchains kept by each transacting parties or any group the transacting parties choose. In each occuring transaction, it's record is added to a blockchain and should be broadcasted to all other blockchains in the system to ensure eveyone is on the same page.

But if there are two conflictiong transactions occuring in different blockchains being broadcasted, then how can we be sure which new block is to be added. And if there are blocks containing fraud transactions, how can we find it and keep in form propagating to other blockchains in the system.

This problem is addressed by choosong whichever block has the most computational work put into it. This way, it would require a very huge amount of computational work for fraudulent or conflicting blocks to stay relevant in the system. Of course, there also should be a way to determine how much computational work has been put into each blocks. The way to do that is to give each block a proof-of-work as a sign of computational work being done before being broadcasted.

B. *Generating a Proof-of-Work Using SHA256*

The proof-of-work involves scanning for a value that when hashed, in this case with SHA-256, the hash begins with some number of zero bits. The average computational work required is exponential in the number of zero bits required and can be easily verified by executing a single hash.

One can implement the proof-of-work by finding some value that when hashed together with a block produces the required number of zero bits. This value is called the nonce and is stored in each block's header. Once the CPU effort has been expended to make it satisfy the proof-of-work, the block cannot be changed without redoing the work. As later blocks are chained after it, the work to change the block would include redoing all the blocks after it.

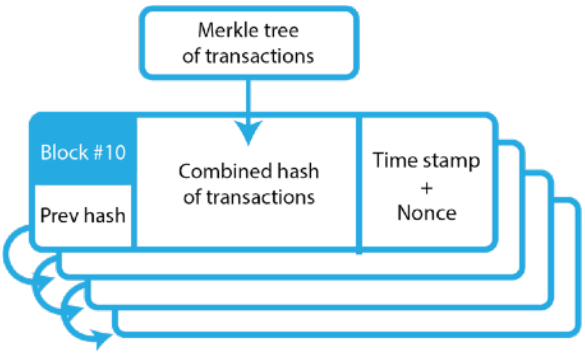

Figure 4.1 The content of a block header in a blockchain.

The proof-of-work also solves the problem of determining representation in majority decision making. This is crusial to ensure the unifrmity of all of the blockchains in the system. If the majority were based on one-IP-address-one-vote, it could be subverted by anyone able to allocate many IPs. Proof-of-work is essentially one-CPU-one-vote. The majority decision is represented by the longest chain, which has the greatest proof-of-work effort invested in it. If a majority of CPU power is controlled by honest nodes, the honest chain will grow the fastest and outpace any competing chains. To modify a past block, an attacker would have to redo the proof-of-work of the block and all blocks after it and then catch up with and surpass the work of the honest nodes. It will require an impossibly large amount of computational work, therefore ensuring the integrity of the system. The next section will show that the probability of a slower attacker catching up diminishes exponentially as more and more blocks are added to the system.[1]

In practice, to compensate for increasing hardware speed and varying interest in running nodes over time, the difficulty to generate a proof-of-work is determined by a moving average targeting an average number of blocks per hour. If they're generated too fast, the difficulty increases and vice versa.[1]

C. *The Protocol*

Based on all the technologies explained in previous sections, protocols are set to enable the distributed blockchain system to work. These rules are needed to keep the integrity of the network intact therefore allowing a completely trustless transaction system. These rules are set in the original Bitcoin paper in 2008. The rules for each node of the distributed blockchain network (each blockchain) are as follows:

1. New transactions are broadcast to all nodes.
2. Each node collects new transactions into a block.
3. Each node works on finding a difficult proof-of-work for its block.
4. When a node finds a proof-of-work, it broadcasts the block to all nodes.
5. Nodes accept the block only if all transactions in it are valid and not already spent.
6. Nodes express their acceptance of the block by working on creating the next block in the chain, using the hash of the accepted block as the previous hash.

Nodes always consider the longest chain to be the correct one and will keep working on extending it. If two nodes broadcast different versions of the next block simultaneously, some nodes may receive one or the other first. In that case, they work on the first one they received, but save the other branch in case it becomes longer. The tie will be broken when the next proof-of-work is found and one branch becomes longer; the nodes that were working on the other branch will then switch to the longer one.[1]

New transaction broadcasts do not necessarily need to reach all nodes. As long as they reach many nodes, they will get into a block before long. Block broadcasts are also tolerant of dropped messages. If a node does not receive a block, it will request it when it receives the next block and realizes it missed one.[1]

V. Security and Transparency

A. *Overview*

The traditional centralized transaction systems achieve a level of security by liming access to the ledger whereby only a handful of people besides the owners have direct access to manipulate the transaction database. While transparency is achieved by revealing the trasaction data whenever an authoritative body requests it. As you can see, a lot trust must be put in the mantainers of these centralized systems.

While this system has been working quite reliably in the last feq hundred years, it is still far from perfect. A centralized system like this concentrates weak points in the system to only a handful of locations and thereby giving attackers fewer variables to cope with.

The distributed system, on the other hand, doesn't require any centralized bodies to mantain the security of the system. A distributed system like the one explained in this paper can take care of the security of the system by itself. It is protected by the irreversabilty of the SHA256 hashing algorithm. In other words, it is protected by the principles of mathematics.

This way, you can see that in a distributed system powered by the security of the SHA256 algorithm, trust does not need to be put anywhere besides the principles of mathematics. Which is as secure as anything can be.

In terms of transparency, because the whole distributed ledger is publicly available to anyone, nothing can be hidden and every transaction is public. Regarding privacy, we can easily make the system to swap public keys regularly to keep the transacting parties' identities safe.

Also worth mentioning is the significantly lower maintenance cost needed to run this distributed network because there is no need to build any rigorously secure database to store all the transactions like in the case of a centralized system.

B. Calculations

The original Bitcoin paper from 2008 addressed any doubts regarding the security of the system against any fraudulent transactions. This security is achieved largely because the use of the SHA256 hashing algorithm that makes it impossibly hard to compromise the integrity of the system. The original paper calculates the security of the system as follows (with some edits):

Let's consider the scenario of an attacker trying to generate an alternate (fraudulent) chain faster than the honest chain. Even if this is accomplished, it does not throw the system open to arbitrary changes, such as creating value out of thin air or taking money that never belonged to the attacker. Nodes are not going to accept an invalid transaction as payment, and honest nodes will never accept a block containing them. An attacker can only try to change one of his own transactions to take back money he recently spent.[1]

The probability of an attacker catching up from a given deficit is analogous to a Gambler's Ruin problem. Suppose a gambler with unlimited credit starts at a deficit and plays potentially an infinite number of trials to try to reach breakeven. We can calculate the probability he ever reaches breakeven, in the case of blockchains, that an attacker ever catches up with the honest chain, as follows[5]:

p = the probability an honest node finds the next block
q = the probability the attacker finds the next block
q_z = probability the attacker will ever catch up from z blocks behind

$$q_z = \begin{cases} 1 & \text{if } p \leq q \\ (q/p)^z & \text{if } p > q \end{cases}$$

Given our assumption that p > q, the probability of the attacker catching up drops exponentially as the number of blocks the attacker has to catch up with increases. With this huge disadvantage against the attacker, if he doesn't somehow have an also huge head start early on, his chances become vanishingly small as he falls further behind. As this head start is practically not possible because new blocks keep being broadcast to the system, the possibility of a fraud node catching up the rest of the system is virtually impossible.[1]

VI. Social and Economic Implications

The trustless transation system powered by distributed blockchain technology has the power to completely rewrite the way societies carry on their economic and social functions. Its trustless nature, its security, its transparency, and its ability to let anyone register and transfer value and data and, allow us to use it to increase the efficiency and the efficacy of various economic and social institutions, both in the private and public sectors. Also, as explained in previous chapters, all of this is largely thanks to the irreversible propery of the SHA256 hashing algorithm.

The power of this system goes to the advantage of both those who are directly benefitting from those institutions, and those who are not. This is made possible by its reduction of the costs and risks associated with those institutions, to its increase of scalability and inclusivity.[6]

Moreover, the trustless transaction system also opens the door to new ways of recording and owning a value such as keeping track of land ownership and many other things. This also makes possible many new institutions which in the legacy centralized system are made impractical by their high costs and risks.

VII. Acknowledgment

I would like to thank Mr. Dr. Rinaldi Munir, Ms. Harilili M.Sc, and Mr. Dr. Judhi Santoso as the lecturers of this amazing class and also giving me this chance to explore the interesting applications of discrete math and learn new things along the way. I would also like to thank my families and friends to keep me motivated during this chaotic times. And I would also thank Paradox Development Studios for developing Europa Universalis 4 so that I can play that game to relieve my nerves during the stresses I experienced while writing this paper.

References

[1] Nakamoto, Satoshi. *Bitcoin: A Peer-to-Peer Electronic Cash System.* 2008.

[2] https://www.buybitcoinworldwide.com/price Retrieved December 2nd 2017.

[3] Mazonka, Oleg. *Blockchain: Simple Explanation*, 2017.

[4] Stallings, William (1990-05-03). *Cryptography and Network Security: Principles and Practice*. Prentice Hall. p. 165. ISBN 9780138690175.

[5] W. Feller, *An introduction to probability theory and its applications.* 1957.

[6] https://www.interlogica.it/en/insight/blockchain-socio-economic-effects/ Retrieved December 3rd 2017.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 3 Desember 2017

Faza Fahleraz 13516095