Hacking the Lightning Network - A protocol to scale Bitcoin [Draft]

Rene Pickhardt

January 15, 2019

Consider financially supporting my book writing effort!

Since this is an open source effort, I rely on the support of the community. In order to be able to work full time on this book, I need a budget of 21.21212121 BTC (with current exchange rates).

Donate using Bitcoin via: `https://tallyco.in/s/lnbook/` or fiat money at: `https://patreon.com/renepickhardt`

[1] `https://github.com/renepickhardt/the-lightning-network-book/LICENCE`

[2] `https://creativecommons.org/licenses/by-nc-sa/4.0/legalcode`

Contents

Chapter 1

Introduction

This book is a guide to understanding the Lightning Network as a technology to solve the scalability problem of Bitcoin. It will first introduce the basic concepts of the Bitcoin protocol. We will only make use of applied cryptography on a high level without the mathematical details. Therefore basically no cryptographic knowledge is needed to be able to follow through this book. For the sake of completeness some of the cryptographic details will be stated in the appendix. Thus the Bitcoin section should be understandable by any computer science (or similar) undergraduate student. As the Lightning Network is built on top of Bitcoin the rest of the book should be easily understandable by the same audience. We conclude the first chapter by looking at some fundamental issues with Bitcoin as a payment system and explain the necessity for the Lightning Network.

We continue by explaining the theory and construction of the Lightning Network. We will understand how to construct payment channels via Revocable Sequence Maturity Contracts and how to solve the routing issues via Hashed Time Locked Contracts. These two forms of smart contracts allow for the Blockchain to be transformed from a transaction layer for payments to an enforcing layer of those contracts. Similar to the real world, most contracts will never have to be enforced via a court ruling but are negotiated bilaterally between consenting parties. In the same sense, we will understand that the Lightning Network reduces the load of the Bitcoin Network significantly by transforming it to be a contract enforcing layer instead of a value transaction layer.

After we understand the core contracts that enable the Lightning Network, we will look at the technologies to define a properly working protocol. We will study the transport layer with its Sphinx mix format and Onion routing first and move on to the gossip protocol which enables the creation of a peer to peer network.

After we understand the technical foundations of the Lightning Network, a practical guide is given. We look at the current wallets, implementations and tools for developers. We create some small Lightning Network applications to demonstrate how easy it becomes with the help of the Lightning Network to accept Bitcoin payments.

Finally, we look at some of the more advanced topics of the Lightning Network. These are mainly relevant for researchers and developers who want to improve the protocol. We decided to include these topics because it seems plausible that users of the Lightning Network gain a clear picture of current trends and potential future enhancements.

One core principle of this book is that we introduce new concepts by formulating problem statements first. We will then try to understand why these problems existed and how the technologies in this book will be able to solve those problems. In many cases, we will derive more specific problems which we will tackle in a similar fashion.

Chapter 2

Bitcoin

2.1 Introduction to Electronic Cash

Problem: How can we create a digital form of cash? With the invention of the micro controller which lead to the development of computers, the Internet and smart phones, we have seen that many things that existed physically now exist in a digital version too. Examples could be:

- Photographs, video recordings and music tapes.
- Books, newspapers and other print publications.
- Postal services for sending mail.
- Archives and libraries.

Also, a lot of communication has become digitized with services like email or instant messaging. It seems that a huge reason for society to adopt a technological breakthrough is that in many cases digital goods are more convenient than their physical counterparts. Take paper publications as an example: While it might be very convenient to carry around one book, it is hardly practical to carry around your entire library. With modern electronic readers, it is possible to access the world's most extensive collection of textbooks on a single device. One can even easily store the entire Wikipedia - which as an encyclopedia is larger than any other encyclopedia that has previously existed - on a small nano SD card. Remember such an SD card is smaller than your thumbnail.

With regard to cash, we can see a similar trend. Due to its scarcity gold served as one of the first universally accepted materials to serve as a currency. However, using gold was not very practical as it is a heavy material and

cannot be split easily. Physical coins and bills (which in the beginning were backed by gold) emerged. This form of cash is still used in all major countries of the world.[1] Similarly to carrying around a single book, it might be highly convenient to carry around a 20 Euro[2] bill in order to buy some groceries. However we hardly see people carrying around 1000 times as much in order to buy a car or even more bills to buy a house. Physical money in large quantities seems to be too cumbersome. The traditional banking sector and financial industry have come up with solutions like wire transfer or cheques. Also, we can observe the emergence of online banking, credit cards and online money transfer services. From a perspective of convenience, these services can be seen as a digital form of cash and are similarly convenient for the user like a digital book.

However, we should state that while the digital solutions which the financial industry provided might be convenient, they cannot - in fact, they must not - be seen as a real form of cash. Physical cash in the form of coins and bills is supposed to be yours if you have it in your pocket.[3] The digital alternatives, however, only exist virtually on some computer. From a technical point of view they are just entries in a third party's database. Thus, they are based on trusting this third party. If your digital cash service provider decides to deny you access to your funds or run away with your money, there is basically nothing you can do.[4] Such a scenario can easily happen as we have seen with the bankruptcy of the famous bank Lehman Brothers in the summer of 2008. Many people that trusted their cash to this bank lost a fortune. While we have to be fair and attest that it was certainly not the fault of their digital and virtual cash systems that they filed for bankruptcy, we see that trusting a bank to have a digital form of cash removes the most important property of cash. The fact that you (and only you) will own it. But the reason banks existed was not only to be a provider of a digital form of cash. They already existed before the digital age because people did not want to carry around vast amounts of money.

So the question remains: Can there be a better form of digital cash in comparison to the solution provided by the traditional banking industry?

Similarly to our example of e-books and physical cash (which we carry

[1] although currency nowadays is no longer backed by gold.

[2] or whatever local currency you prefer

[3] Technically in most countries the notes and coins belong to the country which issued the cash. However you are entitled to the monetary value printed on those bills. As long as the value of the notes is preserved, you can be sure that you are able to spend your cash as long as others accept it.

[4] Of course in most jurisdictions there is the legal system preventing them from stealing your money.

around with us) this form of digital or electronic cash should be stored on the digital device of the user. Thus, it becomes just a piece of information. In this way the user is not dependent on a trusted third party.[5]

However, computers and digital services seem to have a fundamental property that might make them useless for electronic cash systems. Information can easily be duplicated by copy and pasting. We remember that gold was used as one of the first mediums of exchange since it could not be duplicated.

To emphasize this issue again: A physical bill is actually transferred when making a payment. In contrast when spending a digital coin - as a piece of information - it can just be copied and duplicated. The recipient has no chance of knowing that the sender actually deleted the coin. If the sender did not delete the coin, the sender would be able to spend the coin again by copying the information to another person. This process is called double spending. Obviously, double spending sabotages the property that cash should act as a store of value. When the supply is infinite, a single coin becomes worthless.

We can conclude that electronic cash is only useful if we solve the problem of double spending and create a limited supply of coins.

One obvious solution is a central authority or custodian who knows who owns which coins, and allows cash transactions only if the sender has enough cash. While this solution works and is already implemented by traditional banks, it does not solve our desire that we carry around our electronic coins in our devices and - even worse - we still need to trust third-party services with all the risks that we have experienced with the collapse of Lehman Brothers.

Luckily in late 2008 a person or group under the pseudonym Satoshi Nakamoto published the Bitcoin paper. This major technological breakthrough showed the world how it would be possible to create a decentralized form of electronic cash. in which every participant stores access to their own funds and copying those coins does not mean duplicating them (as only one copy can be spent).

2.1.1 Electronic coins as chains of digital signatures

Problem: How do you claim ownership and authenticity of a digital good or a piece of information? **TODO**: Definitely rethink the approach for this section. Also in the above chapter it might not have been wise to already introduce the double spending problem. At the end of the day cash

[5] Of course as Bitcoin and Lightning Network developers we should be aware of the fact that we still trust the hardware producers, the software of our operating system and the consensus mechanisms of the network and protocol.

is about ownership. Therefore, when creating an electronic cash system we have to be able to ascribe ownership of digital information. **TODO**: need to define if I want to talk about information or messages Another goal is to be able to verify the authenticity of electronic cash, as we do with bank notes in our everyday lives. As electronic cash is supposed to be a piece of information we can look in the field of computer science to see how ownership and authenticity of a piece of information can be claimed. The standard procedure would be to make use of Digital Signatures. Interestingly in the digital world it becomes significantly harder to fake an electronic coin in comparison to a physical bank note. [6]

Problem: How do digital signatures work? A well known method to create digital signatures is by using an Asymmetric Cryptographic System. Despite the fact that modern cryptography involves quite advanced mathematical theory and computer science, I want to encourage you to stick with me in this section. Here we will consider this powerful tool as a blackbox and describe its properties, which should be easy to understand even for people who have trouble with mathematics. In the appendix**TODO**: put in label we give some mathematical and technical background of elliptic curve cryptography, and explain how and why it actually works to provide secure digital signatures.

An asymmetric cryptographic system consists of a large set of pairs of keys. The keys of each pair are called the **private** and the **public key** by convention. From a computer science perspective each key is just a piece of information encoded as a bitstring, or from a mathematical perspective each key is just a number. Due to some amazing mathmatical properties these keys turn out to be extremely useful and have a (surprisingly?) asymmetric relation to each other. Hence the name asymmetric cryptographic system.

Assuming we take an arbitrary key pair from the asymmetric cryptographic system this pair will have three important properties. The third property might seem rather abstract and useless at first but it is the property which leads to electronic cash and and cryptographic currencies. [7] The three properties are as follows:

1. **Asymmetry:** We can easily compute the public key from a private key. However the other way around is extremly hard to compute. In this context **Computationally easy** means that there is a well known algorithm and method to do this computation and that the computation also takes very little time. In contrast, **Computationally**

[6] Unless of course one does not handle one's private keys - which are used to create a digital signature - properly.

[7] While there is a little bit more cryptographic theory involved in creating cryptographic currencies this one is probably the most important building block.

hard means that there might be a well known algorithm[8] to compute the solution. But even in that case it would not be feasible[9] to compute the solution with one computer or even a cluster of computers within a reasonable time interval.

2. **Encryption:** With the help of the public key one can easily encrypt a message. It is hard to decrypt the message with the public key. But the encrypted message can easily be decrypted with the help of the private key. Remember the private key is hard to derive from the public key. So we already see where the names come from. A person can publish their public key and everyone else can easily compute an encrypted message. This message can only be easily accessed (decrypted) if the private key is known.

3. **Signatures:** Given a piece of information and a private key one can easily compute a digital signature. Any person who has access to the piece of information, the signature and the public key can easily verify that this signature was computed with the private key from that particular piece of information. If only a single bit in the information was changed after the signature was computed, the verification can no longer be completed using the public key. As with the first two properties, digital signatures are asymmetric in the sense that it is easy to produce a valid signature for a message if the private key is known, but almost impossible to forge the signature if only the public key is known. It is considered safe to assume that the owner of the private key (and thus also the public key) is indeed the person authorized to have created that message.

TODO: INCLUDE GRAPHIC showing from private to public is easy and way back is difficult. Also show encryption and signature verifying mechanics

In the following we will look at some examples of these three properties to get a better feeling of what can be done with them. As one can already see they are being used to control a piece of information.

Example 2.1. Compute the public key from a private key **TODO**: Come up with the actual example. One idea: to use python code from an ecdsa library and refer to the appendix. With the help of the *ecdsa* library in Python we can generate a private key. If we print this key it looks like this: **TODO**:

[8]For example, brute forcing by systematically trying all values for a private key and computing the corresponding public key, then checking if it is the public key that we started with

[9]actually it would be more accurate to say that it is highly unlikely

enter key The object storing the private key has a method to compute the public key which will result in the following key: **TODO**: enter the public key

Example 2.2. Encrypt and decrypt a message Using the public key from the previous example we can encrypt a message. Let our message be *I am a proud supporter of the Lightning Network book*. The encrypted message will look like this: **TODO**: enter message We can use the private key from the previous example to decrypt the message. All of this can be seen in the following lines of Python code.

TODO: enter code here

It is suggested to keep the private key as private and as secret as possible. The great benefit of encryption is that it is sufficient to keep a small piece of information secure and secret (the private key) in order to keep large amounts of information secret. This is done by encrypting the information that is supposed to be kept secret. Since only people with the private key can access the original message we can safely share the encrypted file anywhere as long as we keep the private key secure.

Example 2.3. Lets assume we have the message: *I am a proud supporter of the Lightning Network book* I can use the private key of the Bitcoin address from the fundraiser of this book **1GZx8tWgDd21Rd8b1QdMrzdZGHgyfVkzaD** to compute a signature: **TODO**: do the computation!. Now with the Bitcoin address, which is closely related to the public key, we can verify that the signature was indeed derived from the message and the corresponding private key. This means that the owner of the private key was the authentic person to sign the message. If the message will be changed by adding the missing point at the end of message to *I am a proud supporter of the Lightning Network book.* it will produce the following signature: **TODO**: compute it!. We realize that it looks completely different in comparison to the first signature. This emphasizes the fact that a signed message cannot be modified. This is a strong indicator to the validity of the message together with the signature and the public key. If we use the private key for a different Bitcoin address - for example 1CWtSk58d3rN1R3oGwMobJ6Ets9E2gvPj5 - we can produce the signatures **TODO**: enter for the first and **TODO**: enter for the second message. The signatures again look very different since they now belong to a different public and private key pair. Still, with the second Bitcoin address anyone can verify them. This shows that digital signatures also provide a means of ownership besides the previously seen poperty of proving the authenticity of a message.

While this example is nice in the sense that one can certainly prove ownership of a Bitcoin address in most cases, one doesn't want to prove

that ownership explicitly. Actually, the opposite is true. Many people who use Bitcoin want to stay anonymous. However, we need to have a mechanism to prove ownership of coins in order for others to be able to check that they actually gain new coins if we claim to transfer them. This method will be described in the following example. It will make use of digital signatures to define coins and the process of transfering those coins.

Important! You should have a clear understanding of the following example as it leads directly to the definition of Bitcoin.

TODO: create pictures to depict the situation.

Example 2.4. Let us assume we have a very poor version of Bitcoin which we will call Renecoin. So Rene decides to issue coins in hopes that people believe they are valuable. If that would happen and Rene has the power to issue more coins at his will he would never have to work again.[10] He issues coins by creating the message *This is 1 Renecoin.* Since anybody could create such a message Rene decides to sign the message with his private key. He wants to give this Renecoin to Alice. He does this by extending the message. *This is 1 Renecoin. It is transferred to Alice.* Again he computes the signature to this message. Everyone who believes in Renecoin can now see and verify that this electronic coin really belongs to Alice by checking the signature with the help of the public key which belongs to Rene's private key. If Alice wants to transfer the coin to someone else she can extend the message and sign the extended message. *This is 1 Renecoin. It is transferred to Alice. It is transferred to Bob.* This new message is now signed by Alice's private key and everyone can verify that Bob is the real owner. We can see that there is a chain of owners {1. Rene, 2. Alice, 3. Bob} and a chain of digital signatures {1. Rene's signature, 2. Alice's signature}. The chain of signatures is always one element shorter then the chain of owners. Besides the last entry of the chain of owners, all other entries have to match exactly the entries of the chain of signatures. In this way we have created an electronic coin which can only be transferred to another person if the owner can provide a valid signature for the next transfer message. [11]

In fact if Mallory wanted to steal the coin from Alice before Alice transferred it to Bob this would not work. In order to change the owner, the message that Rene signed would have to be extended in the following way: *This is 1 Renecoin. It is transferred to Alice. It is transferred to Mallory.* This results

[10]Satoshi Nakamoto actually provided a system in which he did not have the direct ability to issue coins. Anyone could do this from the very beginning by participating in the process of Bitcoin mining. This was obviously a smart design decision to not premine coins.

[11]One can find a video explaining this in more detail at: `https://www.youtube.com/watch?v=TrF9RmfyLbw`**TODO**: create a command for video links

in a list of potential owners {1. Rene, 2. Alice, 3. Mallory} But due to the assymetric properties of public and private Keys, Mallory will not be able to provide a proper signature for the last transfer message. This means that we would have a chain of signatures that looks like this: {1. Rene's signature, 2. Mallory's forged signature} Since no one will accept the forged signature, we basically only have a chain with one element. This is just Rene's signature which only authorizes the first transfer to Alice.

If Mallory signed the transfer message with her own private key, the list of signatures would look like this: {1. Rene's signature, 2. Mallory's signature}. No one would accept another transfer message from Mallory because the transfer from Alice to Mallory was never authorized by Alice.

This system obviously requires all participants to keep their private keys secure. Anyone familiar with Bitcoin should already know that. If anyone has access to the private keys they can provide a signature to any transfer message and effectively steal the coins.

Being able to securely ascribe and transfer ownership of Renecoins, one might naively think that we have already created a secure and useful form of electronic cash. However just using the chain of owners and the chain of digital signatures is not sufficient and has a severe problem.

Problem: What happens if Alice also gives the coin to Charlie and not only to Bob? Earlier we saw Alice transferred her Renecoin to Bob. Now assume that Alice also produces a valid signature for the message *This is a Renecoin. It is transferred to Alice. It is transferred to Charlie.* By design of the system, she will be able to produce such a signature as she knows her private key. We would now have a chain of owners **TODO**: maybe introduce arrows here! {1. Rene, 2. Alice, 3. Charlie}. Additionally, there would be a valid chain of digital signatures {1. Rene's signature, 2. Alice's signature}. These two lists would exist together with the orginial lists in which Alice has spent her coins by transferring them to Bob. Remember that in the example **TODO**: work in reference the signatures provided by Alice would look differently because the transfer message differed.

In both cases Charlie and Bob can verify that they have been given 1 Renecoin from Alice. However Rene has only issued one coin. Alice could transfer her coin to even more people. Thus, Renecoins would never have the chance to be a store of value as the supply can be extended arbitrarily by any participant.

The described process is called the **double spending problem** **TODO**: add glossary entry. It can be easily mitigated by using a central authority - for example Rene - that signs all transfer messages. However that would make it risky for people to use Renecoin as everyone would have to trust Rene to act honestly. Luckily in 2008 Satoshi Nakamoto published his solution to the

problem which he called *Bitcoin: A Peer-to-Peer Electronic Cash System.*[12] In order to get rid of a central authority he used a concept called proof of work, which was based on Adam Back's work called *Hashcash - A Denial of Service Counter-Measure*[13].

2.2 Proof of Work and Blockchain

We have seen in the previous section that it is possible to create an electronic cash system in which ownership and validity of coins can be ensured by defining coins as chains of digital signatures. Yet the system was not sufficient since there was the chance for participants to double spend the coins. As this ability would remove a very fundamental property - namely being a store of value - from any cash system we have to solve the double spending problem.

Problem: How can we prevent double spending of electronic cash without using a central authority to sign all transactions?

TODO: get rid of all the structural ideas and produce some text 1. We start by timestamping transactions and publish all transactions to some decentralized storage. 2. We could assume that at a certain point in time a transaction was successfully done –¿ yields a problem: What if two versions of the transaction exist. Either at the same timestamp or at different time stamps. A previously signed and timestamped transaction would have to be rejected to be spent differently (So this is already the block chain) and the immutability of the blockchain comes from the proof of work.

TODO: Question: what about the technical level of detail here and notation for lightning network related stuff? Stuff like bitcoin script... two ideas: First do it at the end of chapter one in a section formal definitions. Or do it in an bitcoin appendix.

2.3 Scalability issues of blockchain technologies

[12] `https://bitcoin.org/bitcoin.pdf`
[13] `http://www.hashcash.org/papers/hashcash.pdf`

Chapter 3

Lightning Network

3.1 Basic properties of the Lightning Network

3.2 Payment Channels via Revocable Sequence Maturity Contracts

3.3 Enabeling Routing via Hashed Time Locked Contracts

3.4 Transport layer: Sphinx Routing to increase privacy

3.5 Peer to Peer layer: The gossip protocol

3.6 Payment requests and BOLT11 invoices

3.7 Basics of Lightning Technology

Chapter 4

Practical guide

4.1 Lightning Wallets

4.2 Lightning Nodes

4.3 Lightning Application Development

4.4 Useful Applications and Tools

Chapter 5

Advanced Topics

5.1 Channel management

5.2 Autopilots

5.3 Securing your Lightning Network node

5.4 Pitfalls with concurrent requests

5.5 Eltoo payment channels

5.6 Channel factories

5.7 RGB protocol and colored coins

5.8 Rendezvous routing to hide the recipient

5.9 Risks

5.9.1 Hot Wallet Risk

5.9.2 Backup dilemma

5.9.3 Denial of Service attacks

- Spam HTLCs to block channels

- Spam the blockchain

Chapter 6

Controversial topics

6.1 Consequences of decentralization

6.2 Custodial Services, Lightning hubs and banks

6.3 Privacy and regulatory challenges

6.4 Consensus, Forking and Altcoins

Appendix A

Cryptography

A.1 ECDSA

Once upon a time... This document shows how you can get ePub-like formatting in LaTeX with the `memoir` document class. You can't yet export directly to ePub from writeLaTeX, but you can download the source and run it through a format conversion tool, such as `htlatex` to get HTML, and then go from HTML to ePub with a tool like Sigil or Calibre. See `http://tex.stackexchange.com/questions/16569` for more advice. And they lived happily ever after.

.1 Contributers

In the order of merging the pull requests:

- vv01f
- adamjonas
- billygarrison
- gchaincl

.2 Donors

In the temporal order of donations:

- import data import from https://tallyco.in/a/1GZx8tWgDd21Rd8b1QdMrzdZGHgyfVkzaI

Some of the glossary terms are taken from the English Wikipedia and might be adopted and modfied. These termes are indicated by the proper attribution to the Wikipedia article as a link at the end of the respecitve entry these entries are - in opposition to the remainder of the book - licensed as CC-BY-SA-3.0. In case they have been modified by by there will be a star (*) at the end of the URL

Glossary

2nd stage HTLC . 21

Asymmetric Cryptographic System Asymmetric cryptography, or public-key cryptography is a cryptographic system that uses pairs of keys: public keys which may be disseminated widely, and private keys which are known only to the owner. The generation of such keys depends on cryptographic algorithms based on mathematical problems to produce one-way functions. Effective security only requires keeping the private key private; the public key can be openly distributed without compromising security.`https://en.wikipedia.org/w/index.php?title=Public-key_cryptography&oldid=877579180`. 9, 21

Basics Of Lightning Technoclogy . 21

bech32 . 21

BIP . 21

Bitcoin . 21

Bitcoin Script . 21

Bitcoin Mining . 21

Blockchain . 21

Breach Remedy Transaction . 21

c-lightning . 21

Coinbase . 21

Commitment Transaction . 21

Computationally hard A problem is considered to be computationally hard if no algorithm exists or is known that is able to compute the solution to the problem rather quickly.**TODO**: exchange this with a more formal definition? Maybe use polynomial runtime.. 9, 21

Computationally easy A problem is considered to be computationally easy if there exists an alogorithm that is able to compute the solution to the problem rather quickly.**TODO**: exchange this with a more formal definition? Maybe use polynomial runtime.. 9, 21

Contract A contract is a set of Bitcoin transactions which result together in a certain desired behaviour. Examples are Revocable Sequence Maturity Contracts to create a trustless, bi-directional payment channel to be built or Hashed Time Locked Contracts to create a mechanism to allow trustless forwarding of payments through third parties.. 21, 24

Digital Signature A digital signature is a mathematical scheme for verifying the authenticity of digital messages or documents. A valid digital signature gives a recipient reason to believe that the message was created by a known sender, that the sender cannot deny having sent the message and that the message was not altered in transit. They can be seen as cryptographic commitments in which the message is not hidden.`https://en.wikipedia.org/w/index.php?title=Digital_signature&oldid=876680165`. 9, 21

ECDSA . 21

Eclair . 21

Electronic cash . 21

Encoding . 21

Funding Transaction . 21

Gossip Protocol . 21

Hashed Time Locked Contract Additional outputputs in the commitment transactions which can be claimed by one node if a secret preimage to a hash is provided within the time lock after the commitment transaction is being mined by the bitcoin network.. 21, 23

Hashfunction . 21

HD Wallet . 21

Input Script . 21

Invoice . 21

lnd . 21

millisatoshi . 21

Onion Routing . 21

Output Script . 21

Payment Channel . 21

Pay to script hash . 21

Pay to pubkey hash . 21

Paymenthash . 21

Penalty Transaction . 21

Preimage . 21

Relative Timelock . 21

Revocable Sequence Maturity Contract This Contract is used to contract a payment channel between two bitcoin users who do not need to trust each other. The name comes from a sequence of states which are encoded as commitment transactions and can be revoked if wrongfully published and mined by the bitcoin netowork.. 21, 23

Revocation Key . 21

Segregated Witness . 21

Sha-chain . 21

Source Based Routing . 21

SPHINX Mix Format . 21

Transaction . 21

Transaction Malleabillity . 21

Transport Layer . 21

Unspent Transaction Output . 21