UMEÅ UNIVERSITY
Department of Computing Science
Master thesis report

Master thesis presentation document
30HP, 5DV143, VT19

Evaluating Cross-chain Settlement and Exchange in Cryptocurrency

Chapter 1

Abstract

To summerize the entire thesis I would say: "I have experimented with, and studied the aspects of different types of atomic swaps. Wth the goal of formalizing a proposal on how they should be standardized and handled in the future." You probably do not know what an atomic swap is yet, but I will cover that.

The general topics covered are:

- Cryptocurrencies and how they can be used in special use cases.
- Cryptography
- Smart/programmable contracts
- Payment channels and lightning network

Don't worry if you do nt know anything about thee topics as I will cover them in this presentation.

Chapter 2

Introduction & Background

Alright, so the thing that is on everybody's mind is probably: what is an atomic swap? An atomic swap is where two parties exchange assets atomically, which means that either the transaction takes place fully, or the state is reset to the pre- exchange state.

Read from oracle defenition

This is made possible by clever use of cryptography and programmable contracts on the bitcoin network and blockchain. So before we can go into more detail on this I should first cover the basics of Bitcoin.

So, most people, especially in computer science, have heard of Bitcoin, but I could almost count on one hand the number of people I have met that have more than a basic understanding of how it works. I could talk for hours about this subject, but sadly there is no time for that. So I will try to give you the shortest possible version where you can at least understand the rest of my thesis.

The simplest description of Bitcoin is a shared public ledger, that relies on proof-of-work to build network-wide consensus. First of proof-of-work is a way to prove mathematically that work was put into doing something. The most common way of doing this is via hashing of some datatype. The hash has to meet certain criteria to be accepted. There is no known way of producing a wanted hash, so the only way is to try different combinations until a good result is found. So if you have data that produces a certain hash that hash serves as proof that you put work into creating it.

Another thing you have probably heard about before is the blockchain, but just as with Bitcoin overall, people know little about what it actually is. A blockchain is basically a shared datatype. It is very reminiscent of a linked list, but allows for branching, meaning that two elements can link to the same parent. We will come back to this in a moment, but first, let's take a closer look at the blocks.

A block is a data structure that has a header and data. The header contains metadata about the block itself as well as a reference to the previous block in the chain. In Bitcoins case, the data in the block is just a list of transactions, but you could put anything you want into this field. The reference to the previous block is what forms the chain. You can from any block follow the references all the way back to the original block. Anyone can add a block to the chain. But it has to meet the proof-of-work criteria. The Bitcoin network independently calculates something called mining-difficulty. This is represented by a large 256-bit number. For your new block to be accepted the header of the block has to produce a hash that is strictly smaller than the difficulty number. You produce unique hashes by changing a field in the header called nonce, this process is what is referred to as mining.

The mining-difficulty is set so that the sum of all participant's hash-calculating power, or hashrate, will produce new block on average every 10 minutes. The difficulty is adjusted every 2016th block.

So how does proof-of-work ensure that the shared ledger stays consistent? This is where concepts like longest chain comes in. The Bitcoin network only accepts the longest chain as truth, in other words the chain with most accumulated proof-of-work. This works as long as the majority of participants is honest. However due to the probabilistic manner of how new valid blocks are found there is still a chance for contention even if all participants are honest. For example what will the network do in the case where two different blocks are found at almost the same time, in different parts of the network? Now there are two chains of equal length. So how does the network decide which one is correct?

Explain pictures

2.0.1 Elliptic curve cryptography

For cryptographic purposes Bitcoin uses a elliptic curve, known simply as elliptic curve cryptography or shortend ecc. I will not cover the exact details of the maths, it can be found in details in my report, basically all you need to know is that it is an assymmetric cryptography, meaning it has private and public keys.

Though encryption and decryption is possible with ecc it is not the purpose it serves for Bitcoin. What is used is signing. This is a concept that all computer scientists should be familiar with but basically what it means is that you can sign a string of data with a private key, and that signature could be independently verified by someone who has the data prior to signing and the public key belonging to the private key, all without revealing the private key.

Clasically signing is used to prove identity. For example a website could sign a document to prove that it truly was sent by the server it purports to be. This is a very useful feature, but in the world of smart contracts the signature has more significance.

Signatures tend to hold the following significance:

- **Identity** - A signatures serves as proof that it came from the holder of the private key.
- **Immutability** - A signature makes whatever was signed immutable in a way. If the data is manipulated after the signing the signature will no longer be valid.
- **Agreement** - A signature can serve as proof that the holder of a private key agreed or approved to whatever data was signed. For example signing a digital contract could be seen as agreement to the conditions in the contract. The agreement can't be withdrawn at a later date as the signature proves mathematically that you signed it. There is of course the possibility that the private key was stolen. Just goes to show how important it is to keep the private key private, especially in Bitocin.

Keep the 3 above points in mind. As they are absolutley vital for the functionality of smart contracts.

2.0.2 Script & Transactions

Alright, now that we got consensus and the cryptography down let's take a closer look at how programmable contracts are made possible over the bitcoin network.

Built into Bitcoin there is a programming language called Script. Not the best of names, but let's not think about that for now. Script is the code that defines exactly what a transaction can and cannot do. It is a very bare bones language, with byte sized instruc-

tions, no heap memory, and only a very limited stack memory, that strictly operates like a stack. Very reminiscent of the Forth language.

A piece of Script code can be defined as either valid or invalid. The only way a script could be valid is if at the end of executing the last instruction there is only one value on the stack, and it equates to true. Anything else will make the script invalid, in fact under certain circumstances the script could be marked as invalid before the execution is even finished.

Script has one major limitation compared to other programming languages, it is by design not Turing complete. Basically meaning that it cannot do anything. What is missing from the usual programming language repertoire is loops. Script has no construct that allows for repeated instruction execution. Instead all Script programs are completely linear and always execute in a limited time. The reason for this has to do with the halting problem and a bunch of anti-spam measures. Even with this limitation Script has a very wide use case as you will soon see.

Transactions in Bitcoin are not as straight forward as you might expect. It's not just a matter of moving from one attached name to another. A transaction in Bitcoin is a datatype consisting of meta data, a list of inputs and a list of outputs. An input is a reference to a previous output from an earlier transaction, an output could be seen as a destination, it holds the amount sent and the "destination". Because of the references to previous transactions in the input you can follow transaction all the way back to the generation transaction when the coins were created in the first place.

The inputs and outputs are where Script comes into the picture. A output contains a partial script. A input contains the complementary part forming the entire script. This is what makes Bitcoin transactions so versatile. You could make the output script be anything that is describable in Script.

For example *Show simple script on screen*. This script requires the redeemer to give the answer to this equation to spend the money in the output. *Show redeeming input script*.

Obviously most Bitcoin transactions are not this basic. The most common script type has a name, P2PKH, Pay to public key hash. In this script the output is given a hashed public key, and the one who wants to spend the output has to provide a signature from the private key related to the given public key. *Show script and redeeming script*

A slightly newer development in Bitcoin is P2SH, Pay to script hash. If the previous example can be seen as paying to a bitcoin address, this one can be seen as paying to a script, or a contract. In this case anyone who can provide the contract together with a partial script that makes the contact valid, is allowed to redeem. The output paying to the script contains only the hash of the script, as the name implies.