CSC2125: Homework #8

Due on November 19

Fan Long

Thomas Hollis

Problem 1

The timelock parameter in a Bitcoin transaction specifies a specific block number for the transaction to be unlocked. Why not use a timestamp directly? What could go wrong with using timestamp instead of block number?

Solution

The Lightning Network is a layer of off-chain Bitcoin processing which allows an almost unlimited transaction number at negligible cost, all while benefitting from Bitcoin security.

This Lightning Network uses Multisig (multiple signatures required to receive/send a transaction) and Timelock (transactions only become available after a certain time using CSV or at a certain block height using CLTV).

Indeed, the Lightning Network uses Hashlocks and Timelocks rather than timestamps because using a timestamp directly would be very silly. This is due to the fact that timestamps in Bitcoin can be spoofed because they are based on a loosely synchronised clock which only keeps approximate time. Indeed, the Bitcoin network's loosely synchronised clock is simply not good enough for the Lightning Network.

What could go wrong with this is that certain malicious nodes could spoof their timestamp leading to a break in the protocol as certain channels would time-out at unexpected times (e.g. prematurely due to spoofed timestamps) thus we could not guarantee a fair off-chain transaction processing scheme.

Problem 2

Suppose a simple lightning network with three nodes, A, B, and C. There are two payment channels, one between A and B and another one between B and C. Now suppose A wants to send 10 BTC to C via this network. How many BTC, B has to have in order to proceed this transaction off-chain?

Solution

If A wants to send 10 BTC to C via the network outlined above containing a middle-man B, then we will need two channels: one between A and B as well as another between B and C. The network is therefore as follows: A - B - C. This is achieved using Hashed Time-Locked Contracts (HTLC) in the Lightning network.

This means that to set up these two channels, B will need to open a channel with A and B will also need to open a channel with C.

For the first channel between A and B, only A will commit 10 BTC into the opening Multisig transaction while B will commit 0 BTC. Similarly, for the second channel between B and C, B will commit a minimum of 10 BTC into the opening Multisig transaction while C will commit 0. Therefore, in total A will need to commit 10 BTC, B will need to commit 10 BTC and C will need to commit 0 BTC.

This means we can conclude that B will need to own 10 BTC in order to proceed this transaction off-chain. Indeed, if B is a large merchant he will need to be quite rich to keep many channels open. Note that all figures above excluded fees for sake of simplicity.

Problem 3

Now suppose the above scenario, what would happen if the payment channel between A and B expires before the payment channel between B and C? What could go wrong?

Solution

In this case the A - B - C network becomes: A   B - C.

Once this happens, i.e. as soon as the A - B channel expires before B - C, then B can no longer receive the payment from A. This is because the A - B channel is no longer active as it has been timed-out and the transaction has been executed on-chain. However, C can still take the committed payment from B and execute it on-chain. Indeed, this would lead to B losing money without any possibility of recovering these funds.

It is precisely for this reason that the time lock for channel B - C must expire (as per the Lightning Network protocol) before the time lock for channel A - B expires.

Problem 4

Do you think it is possible to support complicated smart contract transactions with the idea of lightning network on top of Ethereum? If so, how? (Open Question)

Solution

I believe that it depends by what we mean by complicated.

If by 'complicated' we mean programmatically complicated (i.e. many rules but that are strictly defined in code), I think that this is technically feasible in the lightning network but simply highly impractical as many adaptations need to be made. Indeed, this is partly because Ethereum and Bitcoin have very large number of core differences in their design. In terms of how to implement this, it is worth noting here that Ethereum already has its own version of an off-chain 'Lightning' solution called "Raiden". Raiden is still running on its test-net with an upcoming release due anytime soon. The implementation of Raiden is similar to Lightning Network as it uses channels of state instead of channels of payments to minimise the load on the Ethereum blockchain.

On the other hand, if by 'complicated' we mean human-level complicated (i.e. a divorce settlement or conditional bank contract), then smart contracts are almost unfeasible without a central party. Indeed, this is because we would need the entire justice system of all respective countries to be capable of operating on blockchains. This is only possible by having a complicated network of Oracles that observe legal proceedings and automatically react in consequent. However, these Oracles need to be able to independently parse legal text with full comprehension and reliability. Unfortunately, the field of Natural Language Processing (NLP) has not yet reached such a level of reliability and sophistication.

Indeed, this is the principle subject of the book "Attack of the 50 Foot Blockchain" by David Gerard and I believe that this is the main reason why cryptocurrencies will never entirely take over traditional currencies but instead will simply co-exist side by side (or get incorporated within traditional banks like Ripple is in the process of doing).

Problem 5

People claim that off-chain solutions like lightning network will make the blockchain systems increasingly centralized. Do you agree or disagree? Why? (Open Question)

Solution

I mostly agree with this claim.

One leading cause of centralisation in Lightning is its scalability issues. One of the creators of Lightning, Rusty Russell, wrote on Reddit that the protocol is facing scaling and implementation issues. He states "its likely to get painful somewhere between 10 000 and 1 000 000 channels". Indeed, this corresponds to around 1 000 to 100 000 users (since each user is recommended to have 14 open channels to retain decentralisation). Page 48 of the Lightening Network whitepaper indeed presents a small paragraph revealing these scaling issues due to routing. However, the whitepaper simply says it is "theoretically possible" to overcome this but it does not propose any solution! As a consequence, "Flare" proposed an approach to routing in the Lightning Network through the release of a new seperate white paper. However, when reading this whitepaper, we can see that once again that the proposed solution only scales to around 100 000 users. Indeed, it seems the main way to scale to more users in Lightning would be for each user to have significantly less than 14 open channels which inevitably leads to more centralisation as it violates the decentralisation postulate of having at least 14 open channels.

It seems that the Lightening Network is not standing up to all its promises. Indeed, after being told for multiple years that scalable off-chain Bitcoin using Lightning is just around the corner, my faith is decreasing.

Another piece of evidence to support my belief comes from Jameson Lopp, a former BitGo engineer, who compares the Lightning network as being similar to mining. He says that all networks, no matter how grassroots they are in the beginning, will eventually give way to specialists who are simply better at offering a more reliable service at a more affordable cost. I actually agree with this perspective for the Lightning Network, especially in light of the routing issues. The entire network is constantly readjusting and keeping track of open channels so it is in the interest of the user to pick optimal and reliable channels, which will eventually become 'popular hubs'. These will effectively be operated for profit opening up the Lighting Network to the same curse of centralisation that mining is currently facing. In addition, these 'popular hubs' will get access to a large amount of data that other users will not see. This essentially means the blockchain becomes opaque, with a lot of transactions being executed off-chain (hidden from the rest of the world) with only a few centralised entities knowing them.

The most compelling counter-argument to these criticisms is presented by Thaddeus Dryja, MIT researcher and co-author of the original Lightning Network whitepaper. He argues that, although there will be hubs, since the hardware requirements are low any user can easily close a channel with a big player who becomes unpopular and simply open up a new channel with someone else. While I agree that the switching cost is low and that the Lightning Network is still somewhat appealing, this only confirms that the Lightning Network is destined for centralisation, albeit a highly democratic centralisation (in a suspiciously similar fashion to existing centralised banks and free-market FinTech alternatives e.g. Monzo).

Problem 6

Can an attacker cause problems for a decentralized storage network that implements Proof-of-Retrievability but not Proof-of-Replication? Why?

Solution

Proof-of-Retrievability (PoR) and Proof-of-Replication (PoRep) are two separate proofs that are needed during the operation of Filecoin, which is built on top of the Inter-Planetary-File-System (IPFS).

PoR ensures that a server (a.k.a. storage miners) S claiming to be storing data D of client C, is able to retrieve D when prompted. Rather than having to send the entire D to prove retrievability (highly inefficient), S can send only a random sample of D chosen by C to prove he has D. Indeed, this is a probabilistic proof that S is storing D for C and is able to retrieve it when prompted.

PoRep, on the other hand, ensures that if an S is claiming to be storing D of C then he can cryptographically prove (using polynomial-time algorithms) that he has replicated D to his uniquely dedicated physical storage.

A decentralised storage network (DSN) that implements PoR but not PoRep is vulnerable to three main types of attacks: Sybil attacks, Outsourcing attacks and Generation attacks. This is because PoRep is the protection required to avoid all three of these. To explain this, we must explain each attack.

- Sybil attacks in the context of a DSN would be where malicious miners pretend to store many copies of a particular piece of data by creating multiple Sybil identities each claiming to have access to the data but only storing the data once. If a DSN is not implementing PoRep than S cannot send a PoRep to prove that he has his very own copy. This means Sybil attacks would indeed be possible without PoRep.

- Outsourcing attacks in the context of a DSN would be where malicious miners commit to store more data than they can physically store by fetching data from another provider. Again, a DSN without PoRep would have no way for S to prove that he has his own copy making it vulnerable to Outsourcing attacks.

- Generation attacks in the context of DSN would be where malicious miners claim to store large amounts of data but instead are efficiently generating the data on-demand using a program. If the program is smaller than the data this inflates the miner's likelihood of winning a block reward which is unfair. DSNs without PoRep would again be vulnerable to these types of attacks.

Hence, we can confidently conclude that an attacker could definitely cause problems for a DSN that uses only PoR but not PoRep by exploiting Sybil, Outsourcing or Generation attacks to gain an unfair advantage over his peers while compromising the reliability security of the DSN (in this case IPFS).

Problem 7

The introduced proof of replication includes a Seal operation as part of the setup. Why do you think this Seal operation is required?

Solution

The PoRep concept is to be implemented using a practical approach that can be run on any non-specific hardware without trusted parties. In order to achieve this, Seal-based PoRep is implemented which uses a Seal operation as part of the setup. The Seal operation is a very slow sequential computation which must be performed to generate a replica for the PoRep.

The way that Seal works is that it forces a storage miner S to store a pseudo-random permutation of D unique to his public key such that committing to store n replicas would require the disk space for n replicas to successfully respond to the PoRep within the allocated time. If D is not replicated locally, the computation would take much longer. Indeed, if storage miner S is unable to respond to the PoRep challenge in this tight time frame by using the Seal, it is an indication that he cannot prove that he has replicated the data himself.

Indeed, this Seal operation is therefore required to allow a practical and feasible implementation of a step of PoRep without needing particular hardware or trusted third parties.

Problem 8

Is it possible that a piece of data can get lost? How and what happens then? Do you think this is a problem?

Solution

Of course, this is possible in theory. Data gets lost all the time for a variety of different reasons and the IPFS storage miners that are acting as servers are no exception, regardless of the Filecoin protocol running on top.

However, Filecoin does provide useful protocols to make sure clients C are aware that one of their server S nodes is no longer holding a particular piece of data D. In addition, Filecoin aims to financially refund clients C if the data they attempted to store on the distributed Filecoin network is lost. Indeed, one of multiple scenarios could arise.

If S loses D and wants to try to pretend he still has it, he could try to guess the proof but this will always be recognised as an invalid proof if verified by the network.

Alternatively, if S loses D he could pretend he still has it by not sending any proofs and pretending the network is down. This will eventually cause a timeout and result in a missing proof.

Regardless of if the proof is randomly sent (invalid proof) or not sent at all (missing proof), Filecoin will make this obvious to the network quite fast. Indeed, all the storage allocations are public to all participants so at each block the Filecoin Network checks if the required proofs for each assignment are present and if any single proof is missing or invalid the network penalises the storage miners S by taking part of their collateral. If a large amount of proofs is missing or invalid then the network considers S to be faulty and settles the order as failed before reintroducing a new order for the same D into the market. If all storage miners S are faulty, the piece is lost and the client is refunded.

This is achieved by the Manage Protocol in the Filecoin DSN. More specifically, `AssignOrders()` and `RepairOrders()` are the main functions relied upon. Indeed, since all the proofs are stored on the blockchain this is publicly auditable at any time which is why this is not that much of a problem.

This is because although it is theoretically possible for a particular piece of data to be lost forever this is highly unlikely as it would require every single storage miner holding that data to either go offline or lose the data, at a financial cost to them. Indeed, a client should give his data to more than a dozen storage miners to ensure dropped nodes can be replenished when needed, achieving reasonable protection against data loss (compared to hard drive failure rates).