Açai: a backup protocol for Lightning Network wallets

Abstract— **Nowadays, Bitcoin is considered the worlds most widely used and valuable digital currency. As all new technologies, Bitcoin is not perfect immediately, and a long work has to be done to make this technology more solid and a possible candidate to become the main payment system in the next future. The focus of this project is on Lightning Network, which is the second layer payment of Bitcoin and permits fast payments transactions inside a network of participants without the use of the Blockchain for each transaction. The main problem of Lightning Network, from the user point of view, is the absence of a recovery mechanism for the unspent bitcoins in case of a wallet failure. Our proposed solution, called Açai Protocol, is aimed to adopt the mechanism of Watchtowers, already introduced in the Lightning Network to monitor the channels when a node is offline, as a possible mechanism of backup.**

I. INTRODUCTION

Bitcoin can be defined as a collection of concepts and technologies that are the basis of a digital money ecosystem. The term Bitcoin, that is both the name of the technology and the name of the units of currency, is used to store and transmit value among the participants in the Bitcoin network [1].

Every transaction inside the Bitcoin is stored on a single, distributed public ledger, called Blockchain. All confirmed transactions are included in a so-called *block*, that it is broadcast to the Blockchain, so all users are aware of each transaction. This prevents stealing and double-spending, that happens when someone spends the same currency twice [2].

Bitcoin presents some limits on the number of transactions that the network is able to process. This is related to the fact that the blocks are limited in size and frequency, and the Bitcoin network is able to handle maximally around 7 transactions per second. Compared to a centralized payment system like Visa, which can handle up to 45,000 transactions per second, this is an obstacle if we want to extend the use of Bitcoin on a large scale [3].

This question called also as the Bitcoin Scalability problem refers to the discussion concerning the limits on the number of transactions the Bitcoin network can process. This has created a bottleneck in Bitcoin, causing transaction fees and delayed processing of transactions that cannot be fit into a block [6]. Various proposals have emerged on how to scale Bitcoin, and a contentious debate has resulted, that is characterized as an ideological battle over Bitcoins future [4].

One of the proposed solutions to the Bitcoin scalability problem is Lightning Network technology, introduced in the paper by Poon and Dryja [5]. This technology permits the users to have transactions without publishing all of them on the Blockchain, but just the initial and final status. This does not increase the capacity of the Bitcoin system and allows that more transactions can be done.

The Lightning Network consists of interconnected payment channels. A payment channel is a one-to-one channel allowing two participants to exchange funds between each other. A network is created by connecting multiple channels with different users, making transactions possible across multiple channels. For instance, Alice may pay Bob without having a channel directly connecting them, but using intermediary channels [7].

The develop of Lightning Network technology is still in progress, and like all emerging technologies, there are different challenges to face to ensure that the technology is successfully adopted on a large scale. Nevertheless, this technology is already very utilized and on 1st January 2019, the number of Lightning Network Channels were 16,872 [8].

One of the problems of Lightning Network is not offering a mechanism of backup if one of the nodes loses accidentally all transaction data, stored inside the confidential status channels. As a result, if a wrong channel status is sent to another peer, the node could be considered malicious and be punished with a Lightning Network penalty, which imposes the loss of own funds on the channel.

A. Contribution of the paper

The purpose of this work is analyzing this problem and finding a solution that can permit to have a recovery mechanism avoiding the usage of wrong channel states. This research is fundamental for all who want to make Bitcoin as the main payment system in the next future. The solution, defined as Açai Protocol, has to manage a backup mechanism to guarantee to nodes to recover own status if they accidentally lose all information.

II. THE PROBLEM DESCRIPTION

The problem we are focusing on is the recovery of the unspent bitcoins stuck inside the Lightning Network after a wallet failure (e.g. lost or corrupted transaction data put into the wallet storage).

The Lightning Network provides higher speeds and confidentiality for bitcoin transactions, but the absence of the underlying distributed ledger makes impossible the recovery of unspent transactions through the traditional cryptographic seed and the BIP32 address derivation.

A. Related Work

In our solution, we will adopt two new concepts of Lightning Network: Eltoo and Watchtowers.

Eltoo: we have seen an opportunity within the new mechanism Eltoo [9], which expects the two nodes of an open channel to share the same state(or commitments) for their unspent bitcoins. By leveraging Eltoo one of the two nodes could try to recover missing data from the counterpart since its unspent bitcoins are stored inside the same commitment. This could be a very useful solution, but, unfortunately, presents key vulnerabilities. The biggest one comes from the case that an adversarial node (Eve) does not respond with the latest channel state to the user (Alice), providing instead a previous one. The LN-Penalty fee, involuntary triggered by Alice, represents an economic incentive for Eve to act adversarially, increasing the risks for Alice to ask for help. This project aims to solve this particular issue, and to make the protocol less vulnerable to involuntary triggers of LN-Penalty fees as false positives.
Moreover, the idea of leveraging Eltoo to mitigate the issue is still missing remediation comparable to the mentioned seed + BIP32 recovery which is provided today by any Bitcoin deterministic wallet.

Watchtower: Further exploring upcoming Lightning Network features, the Watchtowers, as have been outlined so far, will provide a protection service to offline nodes, by storing their encrypted commitments. Watchtowers are designed to broadcast, on the blockchain, the latest channel state if an adversarial node tries to spend an old commitment, exploiting the offline state of its counterpart.
As an example, let's suppose to have a circumstance where Alice has open channels with Bob, Charlie, Eve and is using 3 watchtowers: W0, W1 and W2.
Eve, exploiting the situation of Alice being offline or unable to broadcast her commitments could try to broadcast an older channel state to the blockchain, essentially stealing the last payment done to Alice. In this event, a Watchtower is capable to identify this malicious transaction from the mempool and intervene in defence of Alice, even if she is offline.
Alice, in order to keep the watchtowers aware of the latest channel state, has to send a hint and a blob after every update to her commitments with Bob, Charlie or Eve.
For example, the sequence of events after a Lightning payment between Alice and Bob is:

- Change of the channel state between Alice and Bob (a new Eltoo commitment is signed).
- Alice calculates the variable hint by truncating the commitment hash (which is also the txid): hint = tixd[:16].
- Alice calculates the payload blob by encrypting the commitment itself, using the hash/txid as the symmetric key: blob = Enc(data, txid[16:]).
- Alice sends (hint, blob) to one of the watchtowers W0, W1 or W2.

Therefore, some of the Watchtowers are storing the information that Alice needs in case of failure, without the adversarial risks from asking this data to Bob, Charlie or (worse) Eve.
If we imply a process where Alice can probe Watchtowers availability, by randomly retrieving her blobs through the list of hints, we only need a way to backup and restore the list of all of the hash/txid still unspent inside the Lightning Network.
Açai Protocol stores this list of commitments inside a properly crafted blob, using hints as unique data pointers and the Watchtowers for the backup itself. The key to decrypt this special blob is a derived BIP32 public-key, generated by using the latest block height as a rough timestamp. This final step enables the advantages of BIP39 passphrases to recover funds stored also inside the Lightning Network.

III. AÇAI PROTOCOL

This solution, named Açai Protocol after the controversial berry fruit, aims to use the watchtowers not just for monitoring the channels, but also as a backup service. This protocol adopts the same concepts of txid, hint and blob. In order to distinguish the two types of data payload, we will use a different notation: txidç, hintç and blobç.
Therefore:

- txid, hint and blob represent the standard format used by Alice to interact with the watchtowers
- txidç,hintç and blobç represent the format used by Alices wallet to store unspent txid inside an Açai backup.

Technically, Açai Protocol leverages the same endpoints:

- hintç = txidç[:16]
- blobç = Enc(dataç,txidç[16:])

In the Açai protocol, dataç contains an array of txid: [txid_Bob, txid_Charlie, txid_Eve], where (e.g.) txid_Bob is the hash of the Eltoo commitment between Alice and Bob. Therefore, once Alice is capable to retrieve and decode her Açai blob, she can also identify the list of txid stored inside the watchtowers. Every time that a channel changes its state, Alice sends to one of the three watchtowers W0, W1 or W2 (randomly chosen) two different payloads: the channel state information (hint, blob),and the Açai blob(hintç, blobç) containing the updated list of txid. Note that the watchtowers will store both the standard blobs (hint, blob) and the Açai blob (hintç, blobç), without being able to distinguishing them. We can imply that, an Açais blobç has length and size comparable to standard Watchtowers blobs.
In the following sections, we are going to list all the steps needed by Alice's wallet to send commitments data to the watchtowers and how she can request the backup.

A. Send the backup to the watchtower

Scenario: after a payment to Bob, the channel state changed, so Alice has to send the new channel information (hint, blob) and the new backup (hintç, blobç) to watchtower W0.

Steps:

- To generate txidç, which represent the key to identify and decrypt blobç, Alice uses its deterministic wallet address function, applying the current Block height as the derivation variable to generate the pub-key. For example:

pub-key= m/108/0(mainnet)/(accountnumber)/0/ Current_Blockheight

  Where 108 is an arbitrary purpose number, and account number is needed to match the wallet subject to restore process.
- Calculate the txidç_n. Since it is possible for Alice to perform many payments in the same Blockheight, we must enumerate each new state of the Açai backup inside the same Blockheight as: txidç_0, txidç_1, txidç_2, txidç_3...
  Therefore, we can assume that we need to change the hint while keeping the same Blockheight in the BIP32 derivation function, so:

txidç_0= 2SHA256(pub-key)

  As soon as Alice needs to update her Açai backup, but the Blockheight is not yet changed, we can apply the following rule: *the txidç_n is calculated as the hash function of the previous txidç_n-1*.
  For example:

txidç_1= SHA256(txidç_0) txidç_2= SHA256(txidç_1) txidç_3= SHA256(txidç_2) .....

- Truncate the txidç_n into hintç_n= txidç_n[:16] and encrypt blobç_n= Enc(dataç_n,txidç_n[16:])
- Send hintç_n and the blobç_n to watchtower W0, with the information of the new channel status (hint,blob).

B. How to request backup to the watchtowers

Scenario: Alice can request the backup when she has lost all her data accidentally or otherwise she can simply request the backup to the watchtower every time she is online. In this way, she can check that the watchtowers are storing real data and they are providing the promised service.

Steps:

- Ask to the connected nodes the current Blockheight. Note that the nodes cannot cheat, otherwise they could be excluded by the network.
- Use the Blockheight and the seed to calculate the deterministic pub-key:

address= m/108/0(mainnet)/(account number)/0/Current_Blockheight

- Calculate txidç_0 = 2SHA256(pub-key) and the hintç_0=txidç_0[:16]
- Ask to W0, W1 and W2 watchtowers to retrieve hintç_0.

  case a: If one of the watchtowers contains the hintç_0, it could also contain hintç_1 (case of more than one transaction in the same Block). Therefore, Alices wallet calculates txidç_1=SHA256(txidç_0) and asks the watchtowers if one of them contains the hintç_1. If one has the hintç_1, calculate txidç_2 and hintç_2 and so on. If no one has hintç_2, we can assume that txidç_1 contains the latest channel state so Alice can decrypt blobç_1 and extract the list of txid.

  case b: If no one of the watchtowers provides hintç_0, we can assume that Alice does not have any transactions at the current Blockheight. In this case, Alice's wallet generates a new pub-key, decreasing the blockheight by one: pub-key = m/108/0(mainnet)/(account number)/0/(Current_Blockheight-1). Once generated, Alice's wallet calculates hintç_0 and proceed as in the case a.

- When Alice's wallet finds her hintç_n, she requests to the watchtower to send the correspond blobç_n. The blobç is composed by dataç and txidç[:16], where dataç=[txid_Bob, txid_Charlie, txid_Eve]
- From data, Alice can recover the txid_Blob. then she can calculate the hint_Bob=txid_Blob[:16] and ask the watchtowers which one has that value and requests the corresponding blob_Bob. Since Alice knows txid_Blob[16:], she is able to get the data inside blob_Bob and recover the status channel with Bob. The same happens with the recovery of the status channel with Charlie and Eve.
- At this point, Alice is able to check that her data matchest the ones stored in the watchtowers, or otherwise recovery her data.

IV. Assumptions in the solution

To achieve this solution, we assumed :

- Fees: Watchtowers model still needs a form of trustless payment to be economically viable, e.g. to cover the extra storage and bandwidth for the service (both for the normal service and the Açai backup)
- Memory: The Açai data stored in the watchtowers are not deleted or substituted/tampered.
- Açai Protocol does not cover the case where Alice has no idea of which watchtowers she used and when.

V. EVALUATION

This Section will attempt to give an assessment to the suggested Açai protocol, analyzing the main results, how it might affect the Lightning user and a possible drawback.

A. Main results

The Açai Protocol allows having a mechanism to backup data for a Lightning Network wallet. First of all, it maintains the decentralized nature of Bitcoin: in fact, it does not rely on any on-cloud or any other external service. It safeguards the anonymity since its mechanism is not based on the use of public keys, which might identify the user. Through the use of the SHA256 hash function, it maintains the integrity and confidentiality of the stored data. Furthermore, through the Watchtowers, it is able to guarantee an immediate recovery solution.

B. Game Theory, adversarial thinking and economic principles in the solution

This solution also respects the Nash equilibrium, in fact, we do not have a cryptographic solution to mitigate the case in which the Watchtower is not cooperative with the user, for example affirming to not contain a specific hint. On the other hand, game theory principles may be reasonably used to expect Watchtowers service to work. Watchtowers are paid to offer the backup service, through micro-fee every time that a channel is closed, so they are economically incentivized to continue to guarantee the service. Moreover, the watchtowers may not distinguish the different users, so they are not able to ban Alice and the Açai Protocol since it is indistinguishable from normal transactions.

C. Açai Protocol: for a frequent or sporadic user?

One possible criticism of our protocol is that it is more indicated for frequent users, with high use of Lightning network, for example, minimum one or more transactions for every one-two blocks. In fact, for a normal daily user is high power consumption to calculate the txid_ç using the value of the block height. Since there is a new block every 10 minutes, it is quite unlikely that a user has his/her last transaction every time in the most recent block. On the other hand, in case of a big company, for example, an e-commerce company, the probability to have transactions in the last block is higher. To customize the proposed protocol for sporadic users, it would be useful using as the last parameter in the Derivation Path an integer, with the functionality of a counter. The incrementation of this counter permits to have a binary research counting and on a channel that lives more than 32 blocks, this solution permits to do fewer steps.

Derivation Path= m/108/0(mainnet)/ (account number)/0/counter

where counter=0,1,2...n

VI. CONCLUSIONS

The aim of the project was to analyze Bitcoin and Lightning technology, their mechanism and their main features, ending up with a technically feasible solution to mitigate the recovery of unspent Bitcoin after a Lightning wallet failure. Accordingly, the main goal of the work may be considered successfully fulfilled, offering a solution that allows recovering the status channels in case of wallet failure. We have proposed Açai Protocol: a solution that permits to recover the status channels in case of Lightning wallet failure. More precisely, it bases its mechanism on using the Watchtowers not just for monitoring the channels, but also as a backup service. It respects the decentralized nature of Bitcoin, the anonymity of users, integrity and confidentiality of data and allows to have an immediate recovery service.

A. Future works

Bitcoin and Lightning are quite new technology and a lot of work has to be done, to permit these two technologies to improve continuously. In this section, we propose some possible Future projects, that might extend further the work to include new interesting features.

Watchtower service: This protocol may be proposed from wallet providers and e-commerce companies, which they might manage a backup service through watchtowers and offer the solution to their own clients, in charge of a fee.

Lightning Wallet including the Açai Protocol: There are not applications that may offer a backup solution without utilizing an on-cloud service or centralized systems. Therefore, it would be very useful to realize a Lightning wallet that permits to guarantee a backup service through the Açai Protocol, maintaining a decentralized nature, typical of Bitcoin.

Introduce IPFS system in the Açai Protocol: InterPlanetary File System (IPFS) is a protocol and network designed to create a content addressable, a peer-to-peer method of storing and sharing hashed hypermedia files in a distributed file system [10]. A possible idea is creating a shared hashed file where different watchtowers may store all data information for back up. In this way, a Lightning node may ask to one of all watchtowers, that can access the IPFS file, the information to recover its own status channel. Moreover, this permits to solve the problem of memory storage in the watchtowers. In fact, in this solution, it is assumed that data stored in the watchtowers are not deleted or substituted/tampered. In real life, this is causing a waste of memory and having a solution, which includes IPFS, might be helpful.

REFERENCES

[1] Andreas M. Antonopoulos. Mastering Bitcoin: Unlocking Digital Crypto-Currencies. 2014.

[2] The Economist. The great chain of being sure about things. Oct 31, 2015. url: https://www.economist.com/briefing/2015/10/ 31/the-great-chain-of-being-sure-about-things.

[3] Jonathan Chester Contributor. Your Guide On Bitcoin's Lightning Network: The Opportunities And The Issues. Jun 18, 2018. url: https : / / www . forbes . com / sites / jonathanchester / 2018 / 06 / 18/your-guide-on-the-lightning-network-the-opportunities- and-the-issues/#5012c11e3677.
[4] Oscar Williams-Grut and Rob Price. A Bitcoin civil war is threatening to tear the digital currency in 2 heres what you need to know. March 26, 2017.
[5] Joseph Poon and Thaddeus Dryja. The bitcoin lightning network: Scalable off-chain instant payments. 2016.
[6] Jordan Pearson. Bitcoin Unlimited Hopes to Save Bitcoin from Itself. Oct 14, 2016.
[7] Lightning NetworkAuxledger. Blockchain Scalability Hindrance and How to Overcome It. Nov 18, 2018. url: https://medium.com/auxesis/blockchain-scalability-hindrance-how-to-overcome- it-54378166b804.
[8] Bitcoin Visuals. Lightning Network Channels. Jan 1, 2019. url: https://bitcoinvisuals.com/ln-channels.
[9] Christian Decker, Rusty Russell, and Olaoluwa Osuntokun. eltoo: A simple layer2 protocol for bitcoin.
[10] Kurt Finley. The Inventors of the Internet Are Trying to Build a Truly Permanent Web. June 20, 2016. url: https://www.wired. com / 2016 / 06 / inventors - internet - trying - build - truly - permanent-web/.