Secure Password Storage on the Bitcoin Blockchain

Nina Polshakova

Abstract

Modern password managers require both security and persistent storage, but often require a user to trust third-party providers and pay maintenance costs. A blockchain-based password manager would not only provide a secure and persistent way to store credentials, but it would also eliminate the need to trust third-parties with sensitive information and only require a minimum setup cost. A password manager was built on the Bitcoin blockchain to explore the costs and trade offs of different blockchain-based storage methods.

1 Introduction

Persistent storage is the goal of any data storage system, but few systems can maintain persistent storage with minimum setup cost and zero maintenance cost. The problem of low-cost persistence is particularly applicable to password managers. Most modern password managers depend on maintaining a database of credentials on servers, the cloud or in hardware memory. This forces a client to either trust the external provider with their sensitive data or to put their trust in their own local database not getting corrupted. A trusted, incorruptible distributed data storage method would be an better alternative as a password manager than current alternatives.

The Bitcoin Network has been established to be reliable, secure and here to stay. Global acceptance of Bitcoin has increased rapidly since the digital currency first appeared in 2009. The future of the Bitcoin Network as exclusively a ledger to record transaction versus a decentralized data storage platform is still an open question. While the Bitcoin blockchain may have been envisioned as a way to record financial transactions, the applications built on top of the blockchain have promoted Bitcoin and are what ensure its relevance. Although the Bitcoin Network is not built for data storage it has been used for storing text, links and images in the blockchain. The Bitcoin Network provides several benefits to data storage. No account is necessary for a user to create to store data in the Bitcoin Network unlike password storage with a cloud provider. Bitcoin provides a way to store data that is anonymous, persistent and incorruptible. A Blockchain-based password manager also provides permanent data storage that will be persistent over time which cannot be edited by an adversary. In addition, unlike many modern password managers that have monthly maintanence fees, a Blockchain-based password manager has zero maintenance cost. These factors support exploring if a password manager which stores credentials on the Bitcoin blockchain is feasible in terms of actual implementation, Bitcoin cost and search cost.

A report made by LastPass in 2017 shows that the average business employee must keep track of 191 passwords. A Dashlane analysis of data from more than 20,000 users in 2015 found that the average user has 90 online accounts (Lord 2018). In the United States, Dashlane found that there are an average of 130 accounts assigned to a single email address. Storing over 100 password credentials requires a password manager that is not only persistent and secure but easy to search. It also gives an estimate of how many passwords a Blockchain-based password manager should be able to handle on average. The cost of storing data on the Bitcoin blockchain depends on the Bitcoin transaction fee which has remained relatively stable over time. With a focus on the Bitcoin cost of storing data securely, different storage methods and data structures need to be analyzed when saving credentials on the blockchain.

The cost of the Bitcoin transaction fee should be relatively inexpensive for a user to store a password. This cost depends on different storage methods and data structure when saving credentials in the blockchain.

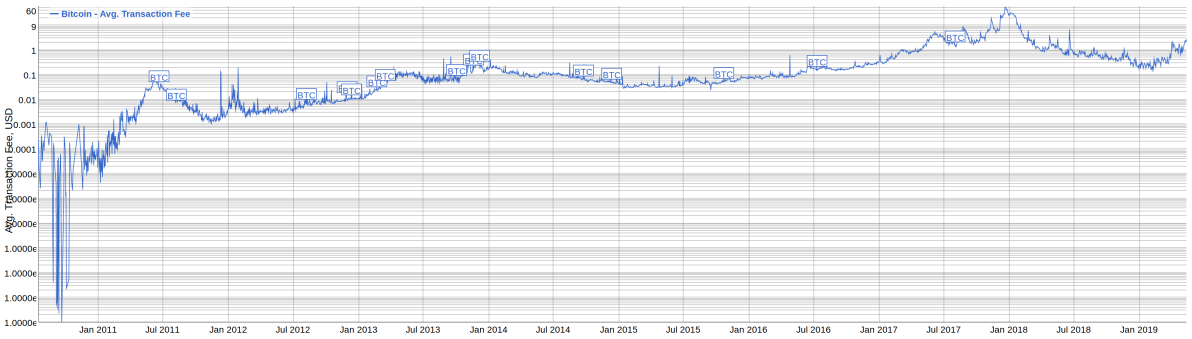

Figure 1: Average Bitcoin Transaction Fee Over All Time (Bitinfocharts n.d.)

The cost of storing credentials using a blockchain-based password manager should also be comparable in price to modern, cloud-based password managers. Two factors determine the cost of storing data in the Bitcoin blockchain: the minimum non-dust limit and the transaction fee. Bitcoin dust is the amount of Bitcoin which is lower than the minimum amount for a valid transaction. A transaction can not include outputs that receive less than a third Bitcoin as it would take to spend it in a typical input. Currently dust limit is currently set to 546 satoshi for a P2PKH or P2SH output on Bitcoin Core with the default fee (Guide 2019).

Looking at the average Bitcoin transaction fee in the past couple years in Figure 1, the cost has been relatively stable over time. Although was a spike in average Bitcoin transaction fees in late 2017 to 55.16 USD, the average transaction fee cost quickly reverted to pre-spike costs which on average are less than a dollar per transaction.

2 Design

The Java library Bitcoinj's API was used to store a user's credentials in the blockchain using Pay-to-Fake-Key data storage. Similar to conventional password managers a user enters their credential's domain name, username and password as plaintext. This information is encrypted and labeled and then stored in the blockchain as unspendable transactions to fake addresses. Project code and high level data structure design is available on a Github repository at: `https://github.com/npolshakova/pswd-manager`

2.1 Client-Side Storage

The client is required to store several things to recover credentials. The client needs to store the HMAC key in order to recreate the IDs to search. The client must also store one encryption key which is responsible for the encryption and decryption of the stored data. Finally, to recover the data structure, the client must a Bitcoin transaction hash. This transaction hash will serve as an ID to locate the transaction with outputs of fake addresses which store the user's data. The data the user needs to store locally can be stored password-encrypted on a user's device.

2.2 Blockchain Storage

Bitcoin is a cryptocurrency not designed for storing data. Data storage on the network incurs costs on the whole network forever. However in recent years there has been an increase in applications that use the blockchain to store data. The Bitcoin Network already has some built-in ways to store data. Coinbase data, the input of a generation transaction, is used by miners to insert ASCII strings such as short messages up to 100 bytes in size or the name of their mining pools. However, coinbase data is not often employed by users

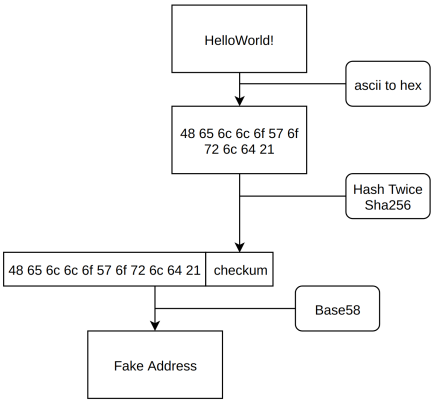

Figure 2: Process of Generating a Fake Address. Image generated by draw.io

to store data. There are several ways to store data in the blockchain as a user which have different trade-offs in terms of Bitcoin cost, permanent storage and future support.

Pay-to-Fake-Key-Hash (P2FKH) stores the data as a public key hash in PubKeyHash field of script and creates unspendable UTXOs (unspent transaction output). This allows a user to store data in 20 bytes per transaction. This creates the most unspendable UTXO bloat as Bitcoin miners and developers have no way to know if the hash corresponds to a real public key that someone owns or not. This requires miners to keep track of these fake addresses, storing the data in the blockchain forever. This method is currently used by Apertus.io and has been used to store relatively large data such as an image of Nelson Mandela (bitFossil 2013). Unlike the OP_RETURN it also ensures that a user will be able to access the data regardless if they are using a full node or API queries to the network.

The Pay-to-Fake-Key method was used in the password manager implementation to store data. This method was chosen mainly due to the multiple methods of access issue to ensure a user would be able to recover credentials regardless of how they chose to search the blockchain. Pay-to-Fake-Key also provides a slightly larger amount of data storage compared to Pay-to-Fake-Key-Hash (33 bytes vs. 20 bytes). Given an input credential string, the ascii string is turned into hex, a checksum is generated and the value is base58 encoded. From the base58 encoding a Bitcoin address is generated. Data using the password manager can also be stored using the Fake Key Hash, but would require a higher cost. In order to store data in the fake key hash, the credential is stored in the base58 encoding itself instead of the raw key. This requires more payments to fake addresses to store data than the Pay-to-Fake-Key method.

3 Data Storage Cost Analysis:

Analyzing a set of fake credentials, the cost, time and search times were analyzed and recorded by storing data on the Bitcoin Testnet. Estimated costs were made based on the expected no-fee minimum cost for the different data structures to insert the nth value into the structure. The calculated costs were based inserting a set of fake credentials using the password manager implementation and calculating the total cost. Since Bitcoin fees fluctuate the calculated costs are not guaranteed to be using the exact same fee, but the fee for each test time was on average comparable based on data from Bitinfocharts (Bitinfocharts n.d.). Estimated costs per data structure can be found in the results section of the full report on the github repo.

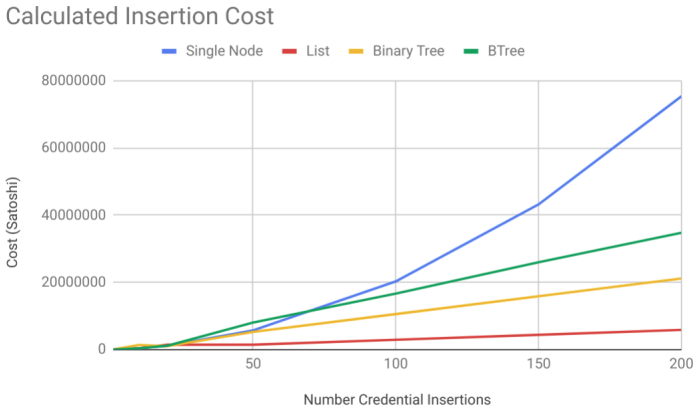

Figure 3: Calculated Cost versus Number of Credentials Stored

Number Credentials	Single Node	List	Binary Tree	BTree
1	23046	23046	41692	23046
10	394130	409030	1381752	409030
20	1154360	1472120	1059384	1169260
50	5721150	1472120	5293800	8074432
100	20307300	2944240	10587600	16668320
150	43202550	4416360	15881400	25992840
200	75341400	5888480	21175200	34765640

Figure 4: Table of Calculated Costs

3.1 Calculated Cost

To calculate the actual cost of storing a credentials on the Bitcoin blockchain, the Bitcoin Testnet was used to calculate the real storage costs for different structures. When calculating the estimated cost, transaction fees were ignored in the calculations and then estimated with a fixed cost of 0.001 BTC/byte. The calculated costs reflect the actual costs of inserting credentials at fixed points (1,10,20,50,100,150,200 credentials). Using the Bitcoinj Java library, fake credentials were inserted in sequence in each structure. Testnet bitcoins were generously obtained from the Bitcoin Testnet Faucet `https://testnet-faucet.mempool.co/` and the change in wallet balance for storing the fake credentials was recorded.

As expected, the single node cost was the highest as the number of insertions increased. As with the estimated cost, the linked list had the lowest calculated cost out of all the different data structures. The binary tree and modified btree calculated costs were relatively high and although they may perform better for larger quantities of credentials, this shows they are not cost-effective for this use case. The average cost of updating a credential in the linked list structure was 7,740,936 satoshi (about $443 USD as of May 5th 2019) for one hundred passwords. This cost is relatively high compared to alternative modern password managers, even though there is no maintenance fee. Cloud based password storage services such as Dashlane Premium cost about $39.99 per year and provides an unlimited number of password storage space. The Bitcoin-based password manager would only be cost-effective in the long-term. Assuming an average user has about 100 online accounts with unique passwords and does not change these passwords frequently, the Bitcoin password manager would only be cost effective after ten years. Alternative data storage methods should be further explored if the feasibility of using the Bitcoin blockchain to store credentials is to be seriously considered.

4 Future Extensions

Secret sharing schemes work best when storing highly sensitive or important information. Since a user wants both confidentiality and reliability when storing credentials, the secret sharing scheme address both

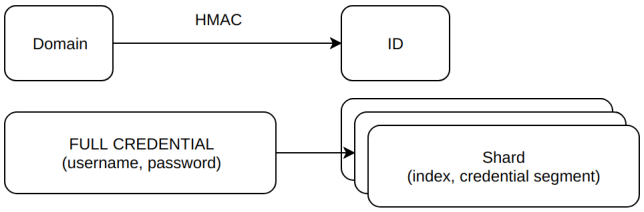

Figure 5: Shard Construction. Image made using draw.io

of these issues by allow arbitrarily high levels of confidentiality and reliability to be achieved. Credentials can "shredded" and stored in multiple data structures to increase security. This approach is explored as a theoretical future extension to the Bitcoin network password manager project. A credential (domain, username, password) can be split into smaller parts called "shards" and reassembled when a user needs to retrieve a password. Instead of storing an entire credential in a single node, list or tree structure, a user can store credentials in multiple separate structures. This would involve a user storing several transaction hashes locally and a way to reassemble the credential shards.

Although this has a higher Bitcoin cost as well as higher storage cost and search time, there are arguably security benefits to shredding credentials before storing on the blockchain. Creating shards of credentials provides a method of secret sharing by distributing the credentials among separate addresses. Unlike third-party cloud-based storage providers, a blockchain-based password manager allows secret sharing to be easily implemented by the user themselves by duplicating the normal storage method of credentials described above for each "shard". Each of these shards needs to be kept highly confidential, but it is also critical that each shard should not be lost, otherwise the user loses their credential. Similar secret sharing schemes are already used in cloud computing environments where a key can be distributed over multiple serves and then reconstructed when needed.

5 Discussion

As the number credentials a user has increases and the ease of access become more important, storing credentials on the blockchain provides a secure alternative to third-party cloud storage. Storing information on the Bitcoin blockchain provides distributed, persistent storage with a minimum setup cost and no maintenance cost unlike other services that charge monthly fees. In addition to no maintenance fees, an account is not necessary for a user to create to store data in the Bitcoin Network unlike most password storage with a cloud provider. A client no longer needs to trust and external provider with sensitive data or use up memory in their own local storage.

The Bitcoin Network has been established to be not only reliable and secure, but also permanent enough for a user to ensure long-term use. Although the Bitcoin Network is not built for data storage, users have been stored text, links and images in the blockchain. A password manager was built using the Bitcoinj Java library to store data in Bitcoin transactions. Analyzing the cost of different storage structures, a linked list structure out-performed the other structures in terms of Bitcoin cost and number of search queries. However the cost for storing even one hundred passwords on the Bitcoin blockchain is still relatively high compared to alternative password manager costs. Future exploration of alternative data storage methods should be explored to see if costs can be lowered. In addition, it may be beneficial to look into using altcoins with lower transactions fees and minimum non-dust costs.