Distributed backup scheme for non-custodial wallets using Shamir Secret Sharing

Anant Tapadia and the BitHyve team
Varunram Ganesh, Arpan Jain, Meer Ali, Parshva Jain

April 2019

Abstract

Shamir Secret Sharing (SSS) is a well-studied threshold signature scheme to store parts of a secret in a distributed way. In this paper, we propose a solution that uses SSS for a distributed backup scheme for non-custodial bitcoin wallets. It offers different security guarantees and protection against attack vectors as compared with other backup schemes. It uses a multi-platform, multi-device, and multi-contact approach that minimizes the trust that needs to be placed in a single entity. The solution also considers specific concerns with such a scheme and includes ways to address them in its construction. Which includes a robust health-check mechanism, introduction of checksum for verifying individual shares, and use of memorable encryption key. The paper further explores how such a distributed trusted network can be used to back up wallet meta-data securely. A model implementation of the same is done with Hexa Wallet.

1 Introduction

The real value of Bitcoin and other such crypto-currencies is that it allows network participants to exchange value with peers without involving a third party making the system censorship resistant, permission-less, immutable and border-less. It is also the reason a hard and deterministic monetary policy can be enforced as there is no central controller.

Users access bitcoins through their private keys held in pieces of software called wallets. A seed is used to derive these keys acting as a backup mechanism. Non-custodial wallets allow users to have full control over their money. But with such ownership comes the responsibility of managing these keys and the backup to guard them against both loss and theft. A study by Chainalysis in 2017 [3] estimated that twice the amount of bitcoins were lost as compared to the amount stolen (4 million and 2 million BTCs respectively. Funds are considered 'lost' when the user has no way to access them, i.e. the backup seed is also inaccessible or unusable. Since there is no way to get back the funds once they are lost, this is a significant deterrent when it comes to mass adoption. In the last ten years, advances have been made for the backup mechanism of wallets with existing solutions offering a specific set of security guarantees. This paper describes a backup scheme that offers security guarantees that are different from the existing solutions. It increases the resiliency of the backup scheme making it much more resistant to loss. Concepts discussed in this paper may apply to other crypto-currencies but have been explicitly tested for Bitcoin wallets supporting BIP 32, BIP 39, and BIP 44 [2].

2 Preliminaries

2.1 Bitcoin Wallets

A bitcoin wallet is a collection of bitcoin private keys that can be used to prove ownership of Unspent Transaction Outputs (UTXOs) on the Bitcoin network [4]. Classified based on custody, these are of two types of wallets - custodial and non-custodial. Custodial wallets are ones where a third party holds private keys and the user has no access to them. While easy to handle and create, these create a central point of failure, which can be exploited by malicious parties to gain access to users' bitcoins. Non-custodial wallets give the user control over his private keys, requiring that she maintain and store them in a safe and secure location. While this provides increased autonomy over bitcoins for users, it carries the risk of loss and theft. The proposed solution aims to develop a non-custodial wallet solution that merges the best of both worlds.

2.2 Seed

A seed is a byte sequence of length 128-512 bytes from which multiple Bitcoin public keys can be derived using the deterministic wallet [5] generation standard as proposed in BIP 32. This standard is widely adopted by all popular Bitcoin wallets as of writing and provides a set of 12 / 24 words (referred to as a "seed phrase") as a backup.

2.3 Seed Phrase

A seed phrase (or mnemonic code) is a set of 12 or 24 human-readable words which can be used to recover a user's seed in case the user loses access to his bitcoin wallet [1]. BIP 39 defines a common standard for Seed Phrases and ensures interoperability between different mnemonic based wallets.

2.4 BIP 39

BIP 39 describes the implementation of a mnemonic sequence and defines a simple word list for the generation of deterministic Bitcoin wallets. To get a binary Bitcoin seed from the mnemonic, the PBKDF2 function is used with a mnemonic sentence as the password and the string "mnemonic" + passphrase as the salt. The length of the derived key is 512 bits.

2.5 Restoring Seed Phrases

To restore a Bitcoin seed from the seed phrase, the seed phrase is converted back into the integer used as the seed to the deterministic wallet generator. Most bitcoin wallets provide a user-friendly UI for the input of a seed phrase and users are shown their balances if successful at seed recovery.

2.6 Backup Schemes

A backup scheme is a method that can be used to retrieve user access to their bitcoin wallets. This may include physical action like retrieving different parts of the private key or may involve mental action like remembering a password. [6] explores the different means of securing private keys.

2.7 Threshold recovery schemes

Shamir's Secret Sharing used in this paper is a type of Threshold Recovery Scheme. Adi Shamir created the cryptography algorithm.

It is essentially a form of secret sharing, where a secret is divided into parts, giving each participant their unique part. To reconstruct the original secret, a minimum number of parts is required. As an example, a secret can be split in 5, and each part is given to 5 different people in such a way that 4 of them have to come together to recreate the secret. This can be any 4 of the 5 people.

2.7.1 Definition

Let m and n be positive integers, $m \leq n$ and X be a secret. An (m,n)-threshold scheme is a method of sharing a secret X among a set of n participants in such a way that any m participants can reconstruct X, but no group of $m-1$ or fewer can do so and it is computationally infeasible to generate a secret s given any other secret s'

There are multiple threshold recovery schemes such as Shamir's scheme, Blakley's scheme, Mignotte's scheme, and Asmuth Bloom's scheme. Of these, we focus on Shamir's scheme.

2.7.2 Shamir Secret Sharing

The idea of Shamir's scheme is that you need m+1 points to define a curve of degree m (i.e.) to uniquely define a line, we need two points, to define a parabola we need three points and so on. Lets assume we want to share a secret m in a finite field Z_p where p is a prime number. We assume the distributor of shares calculates

$$f(x) = a_0 + a_1 x + \cdots + a_{t-1} x^{t-1} (\mathrm{mod} p) \tag{1}$$

and let $a_0 = m$. The distributor chooses a random x_i and computes $y_i = f(x_i)$. Each of the participants is given a point (a pair of x_i and the output y_i) and given any subset of t of those pairs, we can compute the coefficients of the polynomial. The secret $a_0 = m$ can be computed as:

$$a_0 = a(0) = \sum_{j=1}^{\infty} y_j \prod_{\substack{1 \leq k \leq t \\ k \neq j}} \frac{x_k}{x_k - x_j} \tag{2}$$

Several k of n threshold schemes have been proposed after Shamir's initial construction, each with its own set of optimizations, particularly in the field of verifiable secret sharing schemes. (ref. [26], [27], [28])

3 Security Guarantees of a backup scheme

While building a robust system for backup, it is essential to map out the security guarantees of a system that manages access to funds. [7], [8], [9], [10], [11] explore backup mechanisms in various scenarios. Taking reference from the above, we conclude that there are four desirable characteristics:

1. Ease of Use - The ease with which a user can use the proposed solution
2. Resistance Against Loss - Resistance against loss of access to funds

3. Resistance to Theft - Resistance against physical or cyber theft resulting in an unintended transfer of funds

4. Ease of Inheritance* - The ease with which a user can pass on control of funds to future generations

*Though inheritance is an important consideration, the ownership of an asset is defined by law and merely having access to it does not make one the owner (For more, see [12] for a formal definition of Asset Ownership). As such traditional methods like writing a will along with a private way to pass on the access are required. This is outside the purview of the discussion here but can be used alongside the scheme defined.

Ease of Use and Resistance to Loss have a proportional relation with the frequency of use:

- Something that is used less frequently (e.g., a bar of gold in a bank) is difficult to lose
- Something that is used more frequently (e.g., car keys) will be easy to lose

The best way to arrive at a common ground is to separate both mechanisms - One mechanism which is accessed frequently (wallets) and one which is accessed rarely (backup schemes). This distinction is essential to make since it would be impossible to construct a system addressing all four security guarantee requirements simultaneously.

From our argument above, it is quite clear that for a regular access solution, the desirable characteristics are

1. Resistance to Theft and
2. Ease of Use

Similarly, it also becomes clear that for a backup scheme, the desirable characteristics are

1. Resistance to Theft and
2. Resistance to Loss

Note that we do not focus on Resistance to Loss for regular access since there is a backup scheme in place which can be used in case the regular access to funds is lost (not stolen). Also, note that we do not focus on Ease of Use for backup solutions, we assume it is used infrequently and only in the case when regular access is inaccessible.

The topic of this paper is backup schemes and how SSS can be used to offer different security guarantees. There are many advancements in the wallet space (like multisig) which provide additional security for operational use (use of keys). Other than being able to support them, they are outside the remit of the solution proposed herein.

4 Evaluating existing schemes

The below section explores some constructions that have been proposed and used in the past as a backup scheme for Bitcoin wallets.

4.1 Backing up keys in a Digital File

A file involves creating a digital backup file with keys on it and storing it securely on a computer or a digital device.
This was the first backup scheme for bitcoin [13] where users were required to store their wallet file on a hard disk or similar hardware. Restoring access to a person's account is easy since this only involved placing the backup in a specific directory. This ranks slightly better than a paper wallet since its a bit more difficult to lose a hard disk than a piece of paper and its less likely for someone not to notice a hard disk's theft.

4.2 Paper Wallet

A paper wallet involves taking a printed representation of a private key on a piece of paper [14].

Users can restore their accounts by simply scanning their private keys with the help of a smartphone / dedicated hardware. This ranks low on both scales since it is easy to steal (physical theft) and easy to lose (fire hazard, water hazards, misplacement, among others).

4.3 Brain Wallet

A brain wallet [15] is a phrase that is remembered by the user of the wallet. Several schemes have been introduced to improve the brain wallet like in [16], but none of them caught on popularly, and hence we take the standard implementation of the brain wallet into consideration while mapping advantages and disadvantages. More work into exploring brain wallets has been done in [17].

4.4 Multi-signature scheme

A multi-signature wallet (theorized in [19]) involves an m of n signature scheme where a total of m signatures are required from n parties in order to be able to move funds from the wallet.

Multi-signature schemes are beneficial as they can be used for both operational and backup purposes (by allowing the threshold to be tweaked). However, this does not give the flexibility of using any other mechanism at the wallet level like a single signature scheme, which limits the ways a wallet can be used.

4.5 Mnemonics

A mnemonic (first introduced by Electrum [18] as "brain wallet", this later came on to be improved as "mnemonic" by BIP 32) is an easily memorable set of alphabetic daily use words which can be used to retrieve the seed from a wallet. Different wallets implementing mnemonics can be inter-operable as long as they follow the standards defined by BIP 39.

This scheme is backward compatible and can be easily stored since the words can be written down and stored safely in a secure location. However, if mnemonics are lost, there is no recourse, so they do not offer any resiliency.

4.6 Using Salt

A mnemonic can optionally be combined with a user-provided salt or passphrase as described in BIP 39.

Using salt is more resistant to theft than using just mnemonics, but it is less resilient as losing either of the two would mean a permanent loss of access to funds.

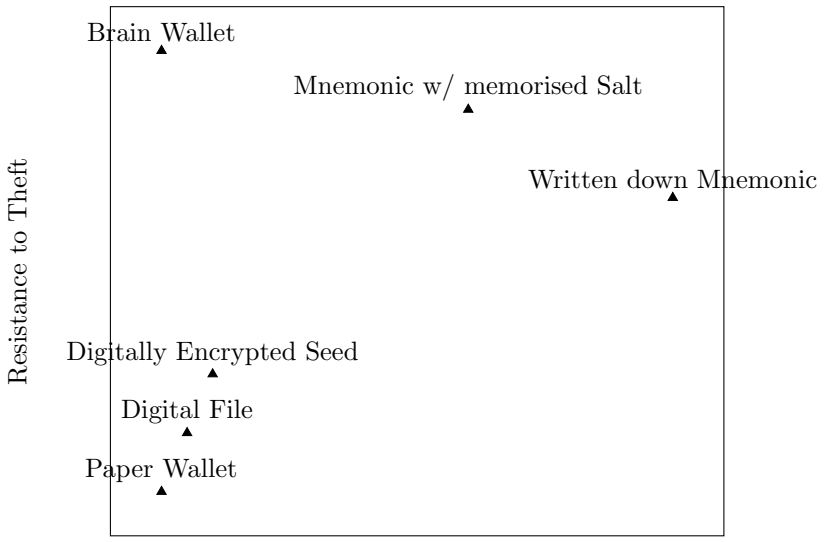
4.7 Using encryption

Any backup method, e.g., Seed can be additionally encrypted (e.g., the scheme described by [20]). Though this gives an additional layer of security, the encryption key used needs to be stored and backed up.

This is more resistant to theft than using just a seed but less resilient as losing the encryption key would mean permanently losing access to funds.

4.8 Summary

We plot the relative effectiveness of these backup schemes for Resistance to Loss and Resistance to Theft. These can be used in different ways changing their properties somewhat, but for mapping them, some assumptions have been made about how they have been used.



Comparison of backup schemes (Fig 3.3A)

From the above, we conclude that Mnemonics (with and without Salt) offer the best tradeoffs.

5 The Hexa Backup Scheme

5.1 Solution Brief

It is important to note that the solution described herein is to be implemented on a local device, e.g. a mobile phone which is in total control of the user and not on a remote server. The server does not even know where the parts of the secret are stored.

The solution involves using SSS for splitting the seed phrase of a wallet into n parts for an m of n scheme. A checksum is then applied to these parts to ensure they are verifiable independently. This is now encrypted with a key that the user will remember easily. These encrypted parts are then enriched with additional data that help identify the type of data it is and the wallet it is associated with. This is particularly helpful when sharing different types of data across a network of wallets. Additional information from the wallet (other than the primary seed phrase) can also be added to these parts which will ensure complete restoration of the wallet as it was.

The solution ensures that these parts are kept in diverse sources, which makes it very difficult to collude or hack. It uses a multi-platform, multi-device, and multi-contact approach. A regular health check of these parts is performed to ensure that the access to these is not lost. It is imperative as the need to restore may not arise for a while, even for a few years, and it is very easy for the parts to have become inaccessible in the meanwhile.

When there is a need to restore, the user will collect the minimum number of parts, decrypt them with the key and combine them to get back the seed phrase and any other data stored along with them.

The solution design goals of the solution and the construction have been described in detail in the following sections.

5.2 Design Goals

5.2.1 Privacy

The privacy of the user regarding PII (Personally Identifiable Information) should not be in a position of compromise. Any third party server should be avoided to avoid information leakage (also to avoid hacks like described in [21]).

5.2.2 Security

The proposed solution must be resilient against an individual or a colluding group of individuals (collusion to retrieve the wallet, not to collude against the consensus mechanism itself like described in [22], [23] and [24])and nobody should be capable of stealing funds or related metadata (not to be confused with transaction metadata described in [25]).

5.2.3 Self-sovereignty

The proposed solution should support the non-custodial nature of wallets.

5.2.4 Standardisation

All standards adopted by the majority of the community in the form of BIPs should be supported. The proposed solution must be defined as an easily implementable standard to enable all wallet providers to adopt it.

5.2.5 Operational flexibility

The backup scheme should not impose restrictions on the way a wallet can be used while following high security and usability standards. This specifically applies to signature schemes - it should support single signature and multi-signature wallets.

5.2.6 Open-source

Any code used to demonstrate the solution or as a reference should be made publicly available unless in some way it compromises the security of the product.

5.3 Construction

5.3.1 Splitting Shares

The Seed of the wallet is split into n shares for an m of n SSS scheme as described below:

Figure 1: Multi-platform, Multi-device and Multi-contact approach

5.3.2 Selecting m and n

1. 3 of 5 scheme: A good trade-off between usability and security will be with m = 3 and n=5.
2. 2 of 3 scheme: 2 of 3 scheme: Other schemes could also be used if the required level of collusion protection is lower.

5.3.3 Encrypting Shares

The encryption key is a single point of failure if the user forgets what it is or is a natural point of attack if it is noted down. Therefore for encrypting the shares, a set of secret questions are used instead of a password. A password has more entropy and is less vulnerable to enumeration/dictionary attacks but is not memorable over a longer time frame ([29], [30]). Therefore, choosing the right question set is very important:

1. Safe: cannot be guessed or researched
2. Stable: does not change over time
3. Memorable: easy to recall and remember

4. Simple: is precise, easy, consistent
5. Many: has many possible answers making it difficult for someone to guess a specific answer
6. Unrelated: when more than one question is asked, they need to be unrelated or of a different category.

5.3.4 Adding meta-data to shares

The shares are then enriched with additional meta-data to help them be identifiable without any PII information. If shared over the air, an additional level of encryption with an OTP (One Time Password) is done.

5.3.5 Distributing Shares

Successfully distributing the shares to independent and unrelated sources is the key to the security of this scheme. Several commonly available sources have been proposed, and the user has to choose a subset of them. Limitations for each have also been identified in order to reduce potential attack vectors and to educate the user about the benefits and drawbacks.

1. Social Contacts - Contacts are trusted by people to hold certain information. In this context, a set of social contacts (¡ m) can be chosen to store shares. This can be implemented by the receivers wallet app, which can also act as a way to receive shares.
2. On Email - Emails are readily accessible and reasonably secure if used with the right authentication. As such, one share per email provider can be stored safely. This can be done in the form of a PDF file locked with the answer(s) of the first security questions.
3. Other Devices - Secondary devices are used frequently as an authentication mechanism, e.g. for 2FA. Similar to social contacts, an app on these other devices can be used to store secrets. These should be limited to one share per device.
4. Local Storage - People also prefer manual storage of secrets such as keys to their apartment, official documents. This provides the highest degree of privacy among all proposed solutions since the user does not have to trust anybody but himself in the process. This includes keeping a local print or a digital pdf file.
5. On Personal Drive - People store various items such as photos and documents on the cloud provided by services like iCloud and Dropbox. Similar to emails, these can be used to store one share of the secret per service provider. This can be done in the form of a PDF file locked with the answer(s) of the first security questions.

All these sources have different properties and different attack surfaces. So if combined such that no specific attack vector has a chance to gain the minimum shares needed, then we make the system highly resistant against all attack types. Specific scenarios may best illustrate the use of sources.
Example 1: Alice distributes the shares of her wallet seed to the below 5 sources

1. Her mom (social contact)
2. A close friend (social contact)
3. The tablet at home (Other devices)
4. On her Google email (Email)

5. On Dropbox (Cloud)

Example 2: Bob distributes the shares of his wallet seed to the below 5 sources

1. His wife (social contact)
2. A local print copy (local storage)
3. The second phone (Other devices)
4. On his home mac (local storage)
5. On proton email (Email)

5.3.6 Verifying a share

As we split the secret into multiple parts, it becomes necessary to verify the validity of each share when we send them out, and when we receive them from a remote peer. For this, we define a checksum that we append to the end of the share.
Checksum: Shamir's secret sharing in its natural form does not do any form of data verification (also has other disadvantages described in [31]), meaning that it is trivially easy for an attacker to pass on a fake share as a regular one. We can add a checksum [32] in order to prevent this and provide an easy way of ensuring the variability of data provided to the sender of shares.

5.3.7 Assimilating Shares

A total of m of the n shares are brought together through the same channel as it was distributed out. For example, using a link from a contact or a scan of the QR code locally stored. These are verified with the checksum and wallet ID before combining it back using SSS to create the secret (Seed).

5.3.8 Health Indicator

One of the issues with backup schemes is their infrequency of use. It may be years before the backup is needed and so it is possible that the user does not remember how to access the backup when needed. This becomes more important when the backup is based on shares kept in a distributed way. Any of these shares may become corrupt or lost or damaged without the user realizing it. This is the reason a health indicator is critical. There may be different ways of achieving this based on the type of share.

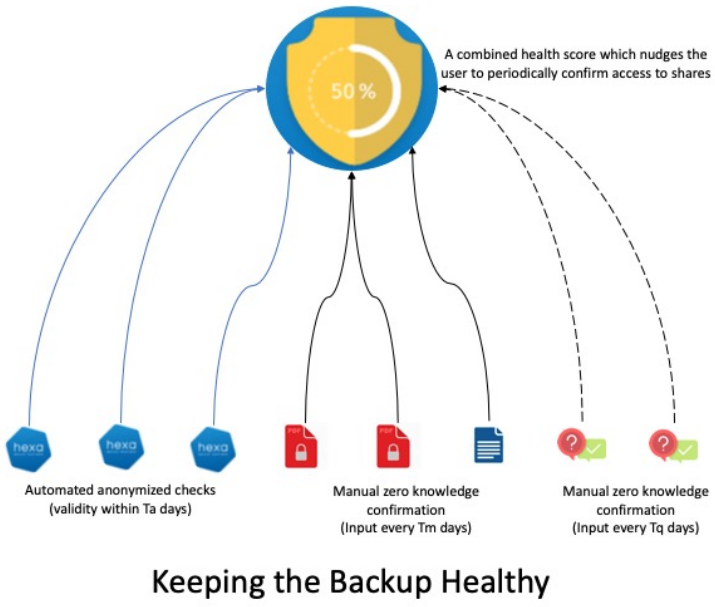


Figure 2: Health check

- For social contacts and app-based shares, it can be automated using a simple ping test to check if the app is live and functioning.
- For manually kept shares it can be something that the user has to confirm once in a specific period.
- The same can be applied to the questions to make sure the user remembers them.
- **Zero Knowledge Confirmations:**To increase security and privacy, Zero-Knowledge techniques can be used when confirming the shares or questions.
- **Anonymous Pull Confirmations:** For automated checks, a simple flag can be placed on the server every time the share receiving app goes online. This can be pulled by the share originating app without revealing anything to the server. The mapping between the apps (Wallet IDs) is maintained locally within the individual apps.

5.3.9 Independent recovery

Finally, it is essential to construct a solution that is not dependent on the app platform, and the funds are not lost if the app cannot be used anymore for any reason. For this, an open-source tool with code that can be readily executed on any platform is available. This code will mainly assimilate the shares to produce a mnemonic which can be used to reproduce the wallet and get access to the funds. Alternatively, the user can decide to hold the mnemonic along with using the SSS scheme, which will give her a standard way to recover funds without needing such a tool.

5.4 Security against attack vectors

Let us consider some potential attack vectors and how secure the system is against them.

- Social collusion: No number of social contacts can collude to generate the seed as social contacts together also hold less than the minimum number of shares needed.
- Hacking a platform: No single platform (email or cloud or otherwise) should hold more than one share, preventing any chance of them being able to get hold of the minimum number of shares.
- App provider going rogue: If the app provider goes rouge, the system should be designed in such a way that even temporarily the minimum number of shares don't touch the back-end. This can be achieved by ensuring that most of the shares are sent through a private mechanism like using QR code.
- Malicious developer: If every upgrade to the app goes through the open source route, a malicious piece of code will be caught in the review process.
- Device stolen: Funds on the device are accessed through keys in the app. The app is generally protected by the device pin and/ or bio-metrics. Additional mechanism like 2 of 3 multisig accounts help secure funds even when the device is stolen and the app is somehow accessed.
- Physical theft: Local copies of shares do not form the minimum number of shares needed.
- Network sniffing: sharing of shares in an encrypted manner (with OTPs) or using QR codes guards against this kind of attack.
- App platform blacklisting: If the app gets blacklisted and removed from the platform providing the app, their may be a potential issue in reconstructing the secret using shares. It should be noted here that even when an app is no longer available for install, old apps still function as intended and so can put the shares back together. Also an open source tool which is platform independent can be used in case the app is no longer usable.
- Multi platform attack: The solution might not be immune to conditions where coordination between different companies or platforms is warranted. E.g., a court order to different companies enforced by law. Though this is possible but very hard to pull off. It would mean identification of most of the sources needed, then hacking into all these sources or collusion across them and finally guessing the secret answer which only the user knows. All this without user realising the attack and not moving all the funds to a new wallet.
- IP tracking and privacy: Chain analysis done on the Bitcoin chain or advanced IP tracking is outside the purview of this solution.

6 Comparison with existing solutions

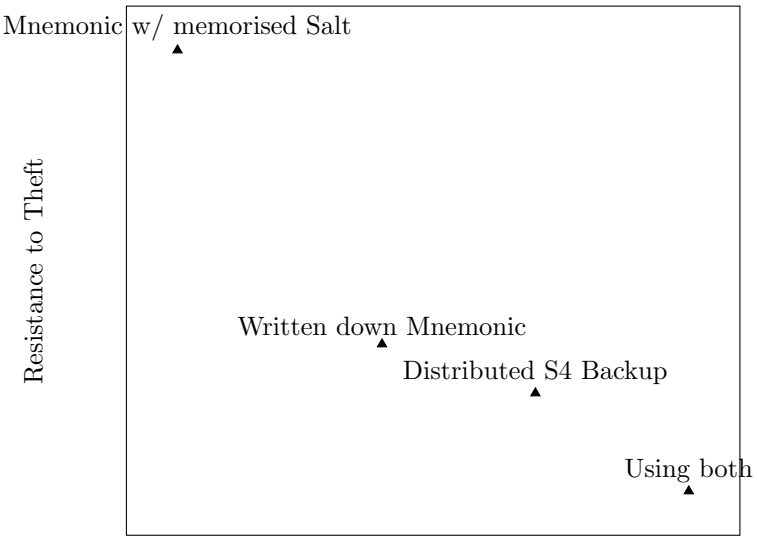
If we compare this distributed backup scheme with the existing solutions (described in [33], we can see that we get a different set of security guarantees which may be better suited for people with different requirements. It adds resiliency to the backup with minimal increase

in the attack surface.

One could also have two backups by combining:

1. Distributed S4 Backup scheme (In App - used as the first level of backup)
2. Written down Mnemonic (Stored in a bank vault - used as a second level of backup)

This gives the whole combined backup scheme a different set of security guarantees. Making it even more Resistant to Loss while increasing the attack surface a little and so reducing the Resistance to Theft by a little.



Distributed backup scheme vs. Mnemonic based scheme (Fig 3.4)

7 Additional Benefits

The solution here establishes a trust minimized distributed network of sources for backing up a wallet seed. The network not only offers high resiliency but can also help backup other data the wallet may need. This can enable many features in a wallet that were not possible previously with only the seed. Example:

1. Different accounts (address groupings) within the wallet
2. Joint accounts between wallets
3. Backing up latest payment channel state (Lightning Network)
4. Trusted contacts (in-app address book) and their PayNyms
5. Other preferences or settings

8 Conclusion

The solution proposed here if implemented with the design goals (like privacy) offers security guarantees with much more resiliency without too much increase in the potential attack surface. Losing one of the shares is not the same as losing the whole backup meaning there is much more resiliency and resistance to loss as compared to storing mnemonic. Also, as the trusted sources are:

1. not known by anyone other than the user,
2. highly distributed across platforms and geographies and
3. not associated with the user wallet even on the server

any attack trying to steal the seed/ mnemonic is highly unlikely.
Different user may use this scheme with slightly different configurations. With their feedback and further research, the system can be further improved.

9 References

1. https://en.bitcoin.it/wiki/Seed_phrase
2. https://github.com/bitcoin/bips/blob/master/bip-0032.mediawiki,https://github.com/bitcoin/bips/blob/master/bip-0039.mediawiki,https://github.com/bitcoin/bips/blob/master/bip-0039.mediawiki
3. http://fortune.com/2017/11/25/lost-bitcoins/
4. https://en.bitcoin.it/wiki/Wallet
5. https://bitcointalk.org/index.php?topic=19137.0
6. https://bitcoin.org/en/secure-your-wallet
7. https://eprint.iacr.org/2017/704.pdf
8. http://www.icommercecentral.com/open-access/blockchain-bitcoin-wallet-cryptography-security php?aid=86561
9. http://ijns.jalaxy.com.tw/contents/ijns-v21-n4/ijns-2019-v21-n4-p852-0.pdf
10. https://eprint.iacr.org/2014/998.pdf
11. http://www.scitepress.org/Papers/2017/62700/62700.pdf
12. https://www.investopedia.com/terms/a/actual-owner.asp
13. https://bitcointalk.org/index.php?topic=921
14. https://bitcointalk.org/index.php?topic=74978.
15. https://en.bitcoin.it/wiki/Brainwallet#Low_Entropy_from_Human-Generated_Passphrases
16. https://bitcointalk.org/index.php?topic=311000.0
17. http://www.jbonneau.com/doc/VBCKM16-FC-bitcoin_brain_wallets.pdf

18. https://bitcointalk.org/index.php?topic=51397.0
19. https://bitcointalk.org/index.php?topic=75481.0,https://gist.github.com/gavinandresen/830ca16758fb9ad496d7
20. https://bitcointalk.org/index.php?topic=17240.0
21. https://bitcointalk.org/index.php?topic=576337#post_toc_21
22. https://www.cs.cornell.edu/~ie53/publications/btcProcFC.pdf
23. https://btc-hijack.ethz.ch/files/btc_hijack.pdf
24. https://arxiv.org/abs/1805.08281
25. metadata
26. https://www.eit.lth.se/fileadmin/eit/courses/edi051/lecture_notes/LN8.pdf
27. https://en.wikipedia.org/wiki/Shamir%27s_Secret_Sharing#Mathematical_definition
28. http://www.cs.columbia.edu/~tal/4261/F16/secretsharing.pdf
29. https://ieeexplore.ieee.org/document/1341406
30. https://pdfs.semanticscholar.org/242e/0eb5e161ef1e3722a613b5bd3d6a32ba8c83.pdf
31. https://ethresear.ch/t/security-considerations-for-shamirs-secret-sharing/4294
32. checksum
33. https://blog.keys.casa/the-evolution-of-bitcoin-key-management/