Delft University of Technology, Bachelor Seminar of Computer Science and Engineering

Towards Data Resilience for Fully Distributed Self-Sovereign Identity Managers

Kalin Kostadinov , Martijn de Vos , Johan Pouwelse
Delft University of Technology

Abstract

Self-Sovereign Identities provide a solution to the identity crisis as their goal is bringing back control over identities to their owners. Nonetheless, currently deployed SSI managers lack data resilience. Consequently, one's identity is lost if the device holding it becomes inaccessible.

We achieve data resilience through identity backups. Unfortunately, there is no research on the matter. Thus, we discover that traditional backup systems need eight additional requirements to become suitable for identity backups. Then we describe two existing solutions and introduce a novel one designed by us. A comparison between them follows, from which we conclude that our novel solution satisfies the most requirements. We then extend an existing SSI manager with a proof-of-concept implementation of our solution.

Our implementation consists of three main components. The first one takes care of identity recovery. The second one allows verifiers to check whether identities and their backups are consistent before verifying their attested claims. And the last mechanism takes care of access revocation.

1 Introduction

Every person on the Internet uses at least one digital identity. Their purpose is to establish trust between service providers and their users. They also serve as a mechanism to distinguish between users, utilized by services for providing an experience tailored to customers' expectations. Unfortunately, we currently are in the middle of an identity crisis [2] since the Internet architecture is missing a unified identity layer. Thus, applications running on top of the Internet need to handle authentication and authorization themselves, explaining why every application has at least one identity management system. As a result, those systems control users' identities, so identity owners cannot administer their data.

In recent years, identity management has become a key concern for governments. That has led to a large number of regulations in the field [3]. There is a need for a novel identity management system, and its formal description stands in the middle of all the work [5]. It promises to Internet users total control over their identities and achieves this by satisfying the requirements of Self-Sovereign Identities (SSI) [1]. SSI allows every identity holder to store and manage their data. For that, users need to use resources under their jurisdiction.

There are already several implementations that cover part of SSI's properties [16], and they have matured over the past couple of years. However, the biggest obstacle preventing them all from going mainstream is the problem of adoption [7]. For the utilization of these implementations, they need to fulfill a long list of real-world usage requirements. One of them is storage, since every digital identity is just a data repository of its owner's claims, and existing solutions are using two approaches to satisfy it.

The first one requires the existence of a single data structure (blockchain). It contains information about all transactions in the network and chains them together. Thus, committing to this data structure requires time and computing resources before data propagates to all nodes. Unfortunately, most real-world identity use cases require high throughput and low latency. Also, in some cases, there needs to be support for offline transactions. And again, having a global data structure stands in the way because transactions need to happen online. Thus, this group of identity managers is not well suited for solving the problem of adoption.

The second group of identity managers uses a local data structure per node in the network. These implementations generally satisfy real-world requirements for throughput and latency. Such systems are also fully distributed [1], thus allowing offline transactions. They have superior functionality to the first group, but they have no data resilience. The problem arises from the fact that such identity managers keep all data in one physical place. And in the case that an identity owner loses access to his identity manager, the identity gets lost irrevocably. For example, implementations that work only on mobile devices are vulnerable to physical damage, theft, and loss.

Solving the data resilience problem of fully distributed SSI management systems could prove crucial. With such a solution, there will be no need for a single distributed or central-

[1]Throughout this paper, the term "fully distributed" would mean a network of nodes, each having an identity management system and a local data storage for the node's identity.

ized data structure. Thus, throughput will increase, latency decrease, and offline transactions will be possible while at the same time having data resilience.

We are the first to acknowledge the importance of data resilience for fully distributed systems since we did not found extensive research on the subject. Our work tries to fill a gap in existing research of fully distributed SSI managers. It will also be a step forward to solving the problem of adoption. Thus, data resilience as a sub-problem of adoption is a research area that is worthwhile exploring. The following research question is at the center of this work:

> ***How to make fully distributed SSI management systems data resilient?***

In the remainder of this paper, we formally specify the underlying problem by defining requirements for solving it (section 2). Then, we assess the positives and negatives of two existing solutions and a third one created by us (section 3). Next, the technical details of implementing our novel solution follow (section 4). We also go over the ethical aspects and reproducibility of the conducted research (section 5). Finally, we discuss results, draw the main conclusions, and suggest ideas for future work (section 6).

2 Requirements for Data Resilience

Data resilience means that data must never get lost. Thus, an identity holder must always be able to access his identifying information. Redundancy is usually the main component for achieving data resilience. The need for redundancy comes because, in fully distributed networks, nodes have vulnerable storage [9]. To make a system resistant to data loss and corruption, a protocol keeping identities in at least two separate locations should solve the problem.

An implementation supposedly looks like a backup system. Nevertheless, SSI managers store identity data that is very sensitive. This data comes in the form of claims. They usually contain signatures of both the identity holder and an attester that guarantees the trustworthiness of the claims. Next to that, SSI managers also hold users' private keys, used for signing transactions with other parties. Thus, we consider the requirements satisfied by traditional backup systems [2] as insufficient for SSI management, resulting in us inventing a few additional ones.

Since an identity backup would contain the whole transaction history of a user, the principles of SSI [1] served us as a source of inspiration. However, not all apply to a backup system, so we have left some out and added new ones. The following requirements for SSI managers' backup systems arise.

1. **Control.** It is of great importance where identity backups are stored because identity owners need to have full control over their data, and this includes their backups. Backup protocols must use storage that is under the jurisdiction of the identity owner. That has never been an issue of traditional backup systems. Even though they deal with identity information, it is usually not part of an SSI. Thus, users of traditional backup systems do not need and have never had direct control over the backups of their identities.

   Furthermore, encryption of backups must always take place for privacy and security reasons. Without having strong security and privacy, identity owners could suffer from identity theft, misuse, or unauthorized modifications.

2. **Availability.** Emergencies within a network of mobile devices that each host an identity manager could happen at any time. And identity owners must be able to recover their identity, no matter when or where they need to do so. Thus, identity backups have to be always accessible to their owners. As a consequence, high availability is a goal. Machines with constant access to electricity and the Internet are a must for backup storage. However, traditional backup systems usually do not need to provide high availability. For them, it is enough that data is persistently stored, sometimes even entirely offline.

3. **Transparency.** Backup protocols need to be transparent. Users must be explicitly aware of how their data is getting processed and where it is stored. If any such detail is unclear, we must not expect trust in the protocol, and regulatory agencies must prevent the adoption of those technologies. There already exist plenty of good examples when closed source systems have secretly stolen or misused identities [8]. Unlike identity backup systems, the management of traditional ones is in the hands of service providers. And next to the fact that identity management systems are not transparent [2], backup systems are not either.

4. **Persistence.** Both identity and traditional backups need persistence to some extent. However, identity backups must reside in storage with a probability of loss or corruption close to zero. If a user loses access to their identity, and its backup gets corrupted, the identity gets irrecoverable, destroying their reputation. Then, the user will have to start collecting attestations about their claims from scratch. On the flip side, persistence in traditional backup systems is desirable but optional. Service providers have access to seemingly unlimited resources, and their backup systems usually do not have to be further replicated.

5. **Portability.** Identities must be able to exist without reliance on any third party. Thus, backups must be transferable to different backup systems or different instances of the same system. In some cases, a user could lose their identity if impossible to move a backup to a different environment. For example, when users transfer their identity to a new SSI manager. Like persistence, portability in traditional backup systems is voluntary and depending on the use case. Usually, service providers do not transfer backups outside their closed source backup system deployments because they already rely just on their systems and no third parties.

6. **Usability.** Adding a backup mechanism to any system introduces increased complexity and some over-

[2] In this paper, traditional/conventional backup systems are backup systems deployed by service providers for the protection of their identity management systems.

head. The design of identity managers usually considers different types of users. For instance, such systems are supposed to be used by a whole nation. Not all identity holders have the technical knowledge about managing their identity backup system. Thus, seamless integration within the identity manager itself is a must for backup protocols. Backups should happen discretely, but users must know that they can rely on a backup, and it is always available and up to date.

7. **Consistency.** With existing backup systems, there might be discrepancies between the data and its backup. In terms of SSI, backup and storage have to be always synchronized. Transactions need to be stored at their backup location first before considering them valid by verifiers. If there is no synchronization, some transactions might get lost. And since at least two users are involved in a transaction, those users might have different knowledge about the current state of the identity.

   Traditional backup systems do not need to be consistent with their source. But identity ones need to since adversaries in the network could exploit inconsistencies to manipulate someone's identity. For instance, Alice makes a claim that gets attested by Bob as part of an agreement between them. Then, Bob causes Alice to lose access to her identity before their transaction gets to Alice's backup. He also hides his transaction with Alice. Consequently, Alice will not be able to prove anymore that she complied with her agreement with Bob.

8. **Access Revocation.** Users must be able to access and restore from their backups through multiple devices. Thus, there needs to be an access revocation mechanism that prevents rogue devices from reaching identity backups. Such devices are, for example, lost or stolen identity holders. Although the implementation of access revocation for traditional backup systems is possible, it is not a hard requirement since backups serve as just a measure for preventing data loss most of the time. They need rarely be accessible online, and they usually reside within private networks.

The following section lays out two existing solutions to the problem at hand and describes a novel one. For all three of them, there is an evaluation, whether they comply with the abovementioned requirements.

3 Achieving Data-Resilience in SSI

In the previous section, it became apparent that data resilience is achievable through the addition of redundancy. And although a traditional backup system might partially solve the problem at hand, no known conventional backup protocol satisfies all requirements. Thus, in this section, two existing solutions and a novel one are evaluated based on the extent to which they meet the previously introduced requirements.

A relevant piece of related work that we were able to find was a paper by Bokkem et al. [16]. It contains an overview of multiple SSI managers and assesses their implementations. Thanks to it, we learned which solutions are lacking data resilience. Some systems already implement identity recovery solutions, and here, we evaluate two of those and propose a novel solution that supposedly has fewer disadvantages than the others.

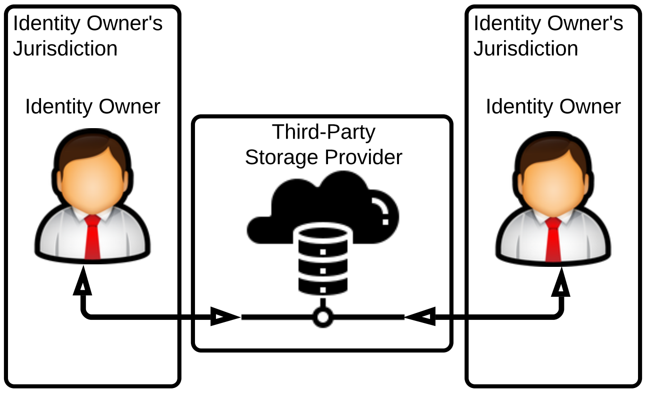
Identity Owner's Jurisdiction
Identity Owner
Third-Party Storage Provider
Identity Owner's Jurisdiction
Identity Owner

Figure 1: Users store backups with third-party storage providers.

3.1 Third-Party Storage Providers

The first solution comes as a proposal for a solution to the data resilience problem of IRMA [4]. In his research, Derksen refrained from specifying where backups should be stored. However, in his future work suggestions, he argues that users could store backups with services like Google Drive or iCloud. For this evaluation, we assume that users utilize third-party cloud storage as a backup space for identities. It is the most user-friendly solution because cloud owners are managers of the resources. Users save time and money since they do not have to deal with data loss and corruption - company operators handle disk failures. Operators also replicate the data enough times to ensure persistence. Figure 1 illustrates how this solution works.

Third-party storage usually sits near the backbone of the Internet. Thus, backups are easily reachable from any point in the network. This solution offers very high availability. Also, costs are low since the infrastructure is used efficiently by multiple users. However, cloud storage is vulnerable to cyber-attacks. Although security measures are state of the art, one server is responsible for the data of many users. Attackers have a better reason to target cloud services instead of single users. And in case of a breach, identity backups might disappear or get stolen.

Usually, cloud storage providers use proprietary closed source software. Thus, it is not always clear how they manage users' data. Identity owners should review such technical details before choosing a backup service. Using proprietary software also hinders portability. It makes it difficult, if not impossible, to move a backup from one cloud storage provider to another. Also, backups reside at a multitude of locations across several countries. However, some governments do not allow for identity data to cross borders for legal reasons. Often, there are local cloud service providers, and their use solves the previously discussed issue.

Access revocation is another problem. Anyone with credentials will be able to access a user's backup and steal one's

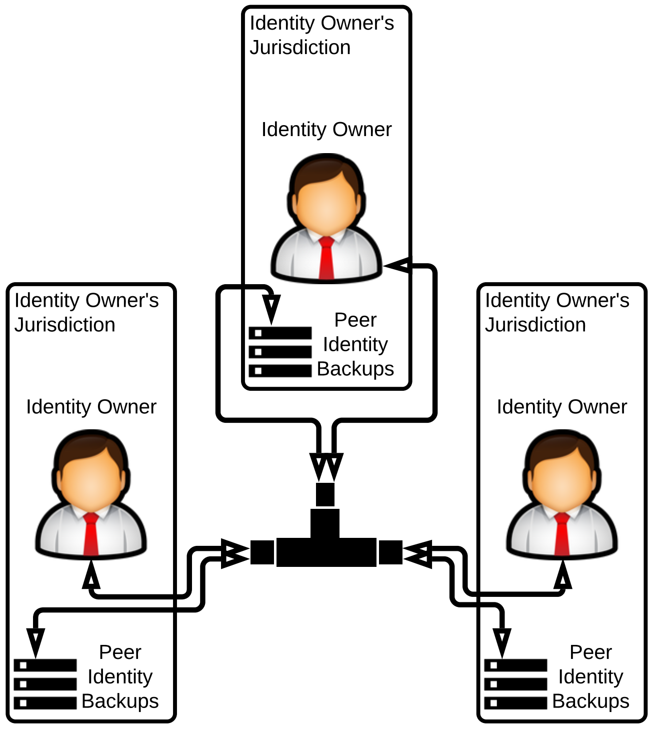

Figure 2: Peers in a network of identity holders store backups on behalf of each other.

identity. There should be extended security measures before authorization. Also, the backup system should prevent a device from backing up data if the identity owner loses access to the device. The work of Derksen introduced a separate mechanism for access revocation. It uses a central key-share server that knows the current authorized device for the management of identities and could revoke access at any time with the user's consent.

Identity owners have command over their backups through a management system, so they have no direct connection to their data. Also, cloud providers often require users to pay for services and apply different policies, which allows them to deny access to services in case of a policy breach. Thus, identity owners do not have complete control over their backup. In case of both service denial and data loss, the damage might become irreversible.

The last problem that we are going to discuss here is consistency. If some transactions do not get stored in the backup system and then the identity gets lost, after recovery, those transactions are not going to be part of the identity, thus, allowing for cheating the system. As a result, transactions first need to be brought to backup before using them in the process of verification. When using third-party storage for identity backups, users need to be online for transactions to move to backup. Thus, with cloud storage, offline transactions are not possible without additional mechanisms.

3.2 Peer-to-Peer Backup

The second solution, implemented by Sovrin [6], is called "social recovery". The idea behind it is to reproduce the blockchain from the knowledge of trusted users about the lost identity. This approach is closely related to the current developments in the field of peer-to-peer backup systems [13]. The concept is to share encrypted pieces of a user's identity with trusted peers in the network so that the probability of ending up with an unrecoverable identity is as low as possible. Figure 2 gives an overview of the system.

With this solution, users again have very little control. On the one hand, the identity manager constructs backups as a snapshot of one's identity and then requests peers to store it. On the other hand, when users want their identities destroyed, they have to ask their peers to do so. Users also need to keep track of peers who store their backups. If this list of network participants gets lost, identity owners immediately lose control over their backups.

Availability and persistence are other issues of this solution. There needs to be an algorithm that keeps track of where backups are stored. It should also request backup replication with enough nodes, so there are always enough available peers for identity recovery. At no point in time, a user should not be able to recover his identity. Unfortunately, identity networks are very dynamic. There might be some offline users during the rebuilding process. Also, some might not be honest about backup versions, and others might not even exist anymore. As a result, the algorithm would generate lots of network traffic and use a significant amount of computing resources to provide availability and persistence. For example, phone storage devoted to backups will become unusable to the owner of the mobile device. Also, there will be a decrease in battery life.

Network participants usually run the same version of an identity manager. However, peers are independent, and some might decide to run modified versions of the code. Thus, a peer-to-peer backup system is not very transparent. Without the consent of the identity owner, peers might decide to sell, destroy, or modify one's backup. Privacy is also a concern in this instance. It is not desirable to keep identity information, even if it is encrypted, on untrusted nodes. Although current encryption techniques are secure enough, in the future, there might be a powerful enough computer that could break the used encryption.

This solution is highly portable. The main algorithm can move backups between nodes. Furthermore, the actual backup gets created by the identity manager. Thus, it can produce backups for different backup networks. The algorithm resides within the identity manager. So, the whole process around the creation and distribution of backups is automated. It is perfect for users since they do not need to do anything to manage their backups.

As with the first solution, consistency is a significant concern. Again, transactions might get lost, allowing users to destroy their identity if they want to hide a specific transaction. Distributing backups with the latest version of one's identity among pees is a must before verifiers start accepting the attestations.

Lastly, there is a need for a revocation mechanism. Sovrin

	Control	Availability	Transparency	Persistence	Portability	Usability	Consistency	Access Revocation
Third-Party Storage Providers	- -	+ +	- -	+ +	- -	+ +	o	+ +
Peer-to-Peer Backup	- -	o	-	- -	+	+	o	+ +
Identity Owner as Storage Provider	+ +	o	+ +	+	+ +	+	o	o

Table 1: A comparison of the three solutions for data-resilience in SSI, in the context of our eight requirements.

has a separate procedure for handling revocation. In this case, each device needs authorization for managing identities. If the device gets compromised in any possible way, users can disallow this device from interacting with their identities.

3.3 Identity Owner as Storage Provider

The third idea that is novel and developed in this paper is to keep the SSI management system on a server under the jurisdiction of the identity owner. Users manage their identities through mobile devices like smartphones, and the need for another machine, controlled by the identity owner, will add some unwanted overhead. It hinders further usability because users need to have the technical know-how to operate the server. Nevertheless, they will have the most control over their identities. Also, this solution achieves transparency since users are solely responsible for their data. Figure 3 shows the novel solution.

An idea to improve usability is to integrate the protocol within routers. The identity backup system will then still be under the authority of its owner. However, the management of the system will happen automatically. Another advantage is that the identity backup will reside on the edge of users' home networks. It will improve accessibility. But, exposing machines to the Internet poses a security threat.

When a server only takes care of one identity manager, there is generally a very low probability of identity data getting corrupted or lost. It is because identity managers do not have computationally intensive processes, and storage requirements are low. Therefore, there is no need for further replication of data from the backup system. Of course, it holds only if the server has uninterrupted access to both electricity and the Internet. However, the problem of a disaster striking the server arises. Perhaps, snapshots of the backups should be stored securely at another location to guarantee that recovery is always possible.

This solution is highly portable as well. Users have to install a fresh version of their identity manager on a new server and recover the identity from the old one. Moving the identity manager to a new server is also transparent to the mobile device used to control the manager.

However, users cannot expect high availability. Depending on where identity holders are when they try to access their identity manager, the server might not be reachable. In this case, transactions happen offline, so there is a need for a mechanism that ensures transaction replication in the backup. This algorithm should account for consistency as well. Thus, transactions should only be considered valid by verifiers after their synchronization with the server.

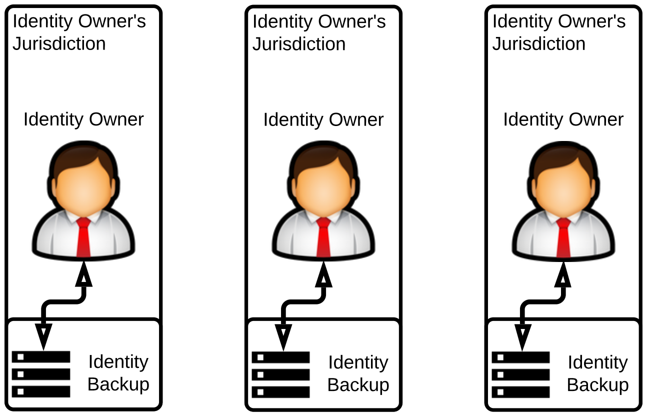
Identity Owner's Jurisdiction
Identity Owner
Identity Backup
Identity Owner's Jurisdiction
Identity Owner
Identity Backup
Identity Owner's Jurisdiction
Identity Owner
Identity Backup

Figure 3: A network of identity holders where each user handles their own backups.

Mobile devices can easily get lost or stolen. Thus, there should be a mechanism that can prevent the before-mentioned devices from controlling the identity manager. Access revocation functionality is easily integrable with this solution since a simple "blacklist" can help the manager deny access to specific devices.

3.4 Comparison

Given the three solutions and the requirements used to evaluate them, we created Table 1. It contains an overview of how far each of the solutions comes to meeting all requisites. We used markers ("- -", "-", "o", "+", "+ +") to show how solutions differ per requirement. They range from "- -" meaning the requirement cannot be satisfied - to "+ +" meaning it can be satisfied, with "o" meaning there needs to be an additional mechanism to meet the requirement.

Table 1 makes it apparent that the third solution meets most of our requirements. It tries to take the best of both worlds by standing in the middle between being centralized and fully distributed. The first solution relies on a third party to handle backups of multiple users, while the second one relies on peers for backup management. However, the third solution can be characterized as both centralized and fully distributed at the same time. From the user's perspective, there is only one machine that manages identity backups. But if we look at the big picture, users each have a personal identity backup manager making backups distributed.

Only with the third solution are users in total control of their identity backups, which is perhaps the most important

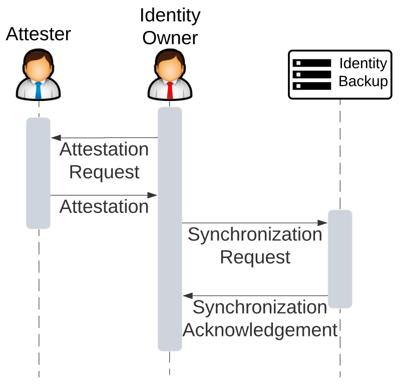

Figure 4: The attestation process after introducing backups.

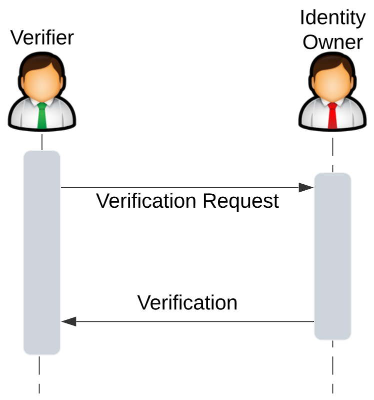

Figure 5: The interaction between the identity owner and the verifier does not change after introducing backups.

feature since the goal of SSI is to give command back to users over their identities. Next to that, it is also the most transparent solution since users manage themselves the software that takes care of their backups.

Persistence is tightly dependent on each use case, and if users replicate their backups multiple times, they can expect a lower probability of their data disappearing. User-managed data will always be highly portable. Moving an identity backup requires only the destruction of the old backup manager and the creation of a new one. Depending on whether the identity backup manager is pre-installed or not, usability can differ. For technical novices, there needs to be automatic deployment software.

In conclusion, since the third solution is missing some additional mechanisms that would make it satisfy all requirements, the following three questions need to be answered:

1. How to allow identity recovery?
2. How to make transactions consistent between the identity manager and its backup?
3. How to revoke access from specific devices when they become rogue?

The following section contains an implementation outline that uses the Trustchain Super App as a platform to develop an answer to the abovementioned questions.

4 Implementation Details

The Delft Blockchain Lab develops one of the SSI management systems, called IPv8 [12]. It is arguably the most sophisticated SSI management system. However, the issue with IPv8 is that it does not offer long-term data resilience, thus not offering a mechanism for recovery from lost access to the identity manager.

Within IPv8, every user has its blockchain, called TrustChain [10], for managing their identity. And Trustchain allows IPv8 to work as a fully distributed system. The idea behind this design decision is that users have more control over their own identity if they are the only ones physically possessing their data blocks.

Mobile applications are the most effective way of hosting an identity management system like IPv8. And an example of such an application is the Trustchain Super App [15]. However, mobile devices are not reliable enough. Thus, it is not clear how users should recover their identities when access to them is lost. IPv8 falls within the group of SSI managers that store identities locally. Consequently, it suffers from the problem this paper is trying to solve. That is why we are using IPv8 as a platform to implement the third solution from section three.

In our solution, every deployment of IPv8 should consist of two parts - an instance of IPv8 [14], running on a server under the user's jurisdiction as the identity backup system, and an instance of the Trustchain Super App as the mobile identity manager. The identity backup server represents the user's actual identity. Thus, its public key is the user's identifier. The mobile deployment is only the control device for the server. Now the answers to the three questions posed at the end of section three follow.

4.1 Identity Recovery Mechanism

Identity creation happens during the initialization of the identity backup system. It occurs when a user decides to create a new identity through the Trustchain Super App. During the initialization process, the mobile app generates a mnemonic phrase using a password provided by the user as a seed. This phrase allows the Trustchain Super App to regenerate the mobile device's public/private key pair.

From that moment, a user can use the mnemonic phrase and the public key of the backup system to restore their identity. And in the case that a user loses access to their current device, they can retrieve their data on a new one using the restoration mechanism. During recovery, the whole transaction history gets transferred to the new device. The identity backup manager notes the identity of the mobile deployment and adds it to a list of devices with access to the backup.

4.2 Transaction Synchronisation Mechanism

Each user has two pairs of public/private keys. The first one is owned and handled implicitly by the identity backup manager. And the user explicitly manages the second one through the Trustchain Super App. With its private key, the mobile device can sign transactions. The idea is to support online and offline attestation and verification. However, there is a difference between online and offline transactions. In online ones, both the attester/verifier and the user have access to the identity backup manager. But, in an offline setting, it is not accessible to anyone.

Each claim should be signed first by the mobile identity manager, then by the attester, and finally by the backup manager when the mobile deployment succeeds in connecting to it. The identity backup manager keeps track of its mobile deployments and signs only transactions having a signature from one of them. In Figure 4 there is a transaction diagram showing the process of attestation. The only difference compared to how the system looked before is the obligation of the user to synchronize their attestations with the identity backup.

Attesters are not interested in seeing the third signature. However, verifiers regard any transaction that lacks it as invalid. Transactions can be verified when they have all three signatures. The verifier checks whether the transaction has a signature from the identity backup system before verifying it. Figure 5 contains the process of verification. It is similar to the previous design of Trustchain. One of the main drawbacks to this system is that attestations become valid only after synchronizing them with the identity backup manager. To make this solution backward compatible, both attesters and verifiers will look for a marker that will lead them to use and expect the correct transaction structure.

To implement the abovementioned functionality, we need to add two new fields to the Trustchain transaction [11]. The proposed transaction structure is visible in Figure 2, and the fields are in positions seven and eight. The first one will contain the public key of the identity backup manager. It is the key that represents the real identity of the user. Verifiers will distinguish the actual identity holder using it. The third signature from the identity backup manager will reside in the second one. It will prove to verifiers the successful data synchronization between the mobile identity manager and the identity backup manager.

Number	Description
1	Requester public key
2	Requester sequence number
3	Responder public key
4	Responder sequence number
5	Requester previous hash
6	Signature
7	Backup manager public key
8	Backup manager signature
9	Transaction block size (n)
10	Transaction block

Table 2: Modified Trustchain Transaction

4.3 Access Revocation Mechanism

Access revocation will use the list of known identities and modify it to reflect changes. Such will occur when a device goes "rogue". Thus, it is no more accessible to the user, or the user switches his mobile device. The main drawback is that mobile deployments contain the whole transaction history, and adversaries could gain access to the identity if they successfully break the security measures.

The mechanisms outlined above show a possible interpretation of solution three. The approach can be considered generic and should apply to similar systems.

5 Responsible Research

We think that this research topic is part of the foundation of SSI. Thus, we tried to make our work first- and foremost relatable. That is why we decided to enumerate the most important requirements for an identity backup system. Our specification for the solution to the problem of data resilience will help other developers better and critically assess the quality of our contributions. Furthermore, given they agree with the requirements, they could create more elaborate interpretations that compete with the abovementioned one. That is important especially given the time constraint of this project. And since this project ran for only ten weeks, we could not conduct any evaluations. Thus, again we focused most of our efforts on eliciting the requirements. That helps other researchers follow our line of thought and run some experiments that will determine the quality of this paper.

In terms of reproducibility, in section four, we formally described an implementation of our solution with a currently existing system. Following the example, developers of other SSI managers should also be able to reproduce our work. The main takeaway from our implementation is that transactions should possess a signature from the identity backup system to regard them as valid, and verifiers should look for them during verification. Furthermore, backups should reside within the jurisdiction of the identity holder. Thus, users need to host their identity backup manager. And last but not least, identity backup systems must employ an access control mechanism that allows users to recover their identities and prevents unauthorized access to identities by possible adversaries.

Ethical aspects were another pillar of our efforts to provide an answer to the research question. Our solution takes into account the need for user control. Thus, we gave identity holders total control over their identity backups. Furthermore, since users administer their data, their privacy gets preserved. Also, we believe that transparency is of great importance, and we advocate that software should run on machines managed by the identity owner. Moreover, we took usability under consideration by requiring every identity management system to be seamlessly integrated with its backup system.

To conclude, this paper was able to balance between a concrete implementation and an abstract solution definition. Both of those are important for the further development of SSI backup systems.

6 Conclusions and Future Work

We now complete this work with conclusions and suggestions for future work.

6.1 Conclusions

This research paper outlines a general solution to data resilience that some SSI managers need. First of all, we began our work by devising eight requirements for identity backup systems. They help show what properties should be possessed by a system that takes care of identity backups. For them, we took inspiration from the ten principles of SSI [1].

We were not able to find prior research that concerns identity backups. Thus, previously there were no sources that specified the requirements of such a system. That makes our work novel and encourages other researchers and developers to adjust their solutions according to our contributions or question them.

Secondly, we tried to answer our research question by finding the solution that best fits our requirements. Unfortunately, the currently deployed SSI backup systems do not provide satisfactory results.

The first solution that we described uses third-party providers for backup storage. It achieves high availability and is the most user-friendly of the three. However, it was flawed because users can never have direct and total control over their data.

The second solution relied on peers in a network of identity managers to handle backups of users. It was motivated by the existence of peer-to-peer backup systems [13]. Its most major concern was that such networks are usually very dynamic, and excessive resources will get wasted to ensure persistence and availability.

Consequently, we were inspired to find an original answer to our research question. Our solution managed to satisfy most of our requirements. It demands users host their identity backup manager on a separate from their identity manager server. However, to satisfy all requirements, our solution needed further development. It was missing three main mechanisms:

1. Allowing identity recovery.
2. Making data consistent between the mobile identity manager and its backup system.
3. Revoking access to mobile identity managers that have gone rogue.

We used section four to describe a possible implementation of the solution and the three missing mechanisms. Trustchain acted as a platform for our development. Unfortunately, the length of the project allowed for only a partial implementation in code.

In conclusion, the elicited requirements and our novel solution should lead to the further development of identity backup systems that do not rely on centralized or global storage. With them, we take the first vital steps towards data resilience for fully distributed identity managers. Their need is part of the problem of adoption that keeps SSI managers from going mainstream.

6.2 Future Work

There needs to be an evaluation of the proposed solution. After its full implementation, developers should run several experiments to determine where availability stands and how resilient the system is to data loss. An experimental environment that simulates disasters in a network of identity managers could help in achieving this goal. Furthermore, researchers should conduct experiments with real users to measure usability, which we consider one of the most important aspects of any identity backup system.

Since in our solution, both devices are fully featured nodes in a network of identity managers, transactions done by the mobile identity manager are on behalf of the identity backup manager. Thus, we have touched upon transactions that happen on behalf of other peers. Consequently, the following question for further research arises: *Should users be able to make transactions on behalf of their peers, and how should those transactions happen?* Our current developments might prove helpful in answering that question.

Lastly, a further advancement that goes beyond the goals of this research project will be the creation of emergency access "terminals" that will be available at border control, for instance. They will allow someone access to their identity manager with restricted controls if their other SSI managers are not reachable. Those emergency "terminals" should only allow for verification of attestations. And with that mechanism, SSI managers will begin their journey towards disaster resilience.

References

[1] Christopher Allen. The path to self-sovereign identity. 2016. http://www.lifewithalacrity.com/2016/04/the-path-to-self-sovereign-identity.html.

[2] Gergely Alpár, Jaap-Henk Hoepman, and Johanneke Siljee. The identity crisis. security, privacy and usability issues in identity management. 2011.

[3] European Commission. Regulation (eu) 2016/679 of the european parliament and of the council. 2016. https://eur-lex.europa.eu/eli/reg/2016/679/oj.

[4] Ivar Derksen. Backup and recovery of irma credentials. 2019. https://privacybydesign.foundation/pdf/Backup-and-Recovery-of-IRMA-credentials-thesis.pdf.

[5] Md Sadek Ferdous, Farida Chowdhury, and Madini O. Alassafi. In search of self-sovereign identity leveraging blockchain technology. *IEEE Access*, 7:103059–103079, 2019.

[6] Daniel Hardman. What if i lose my phone? 2019. https://sovrin.org/wp-content/uploads/2019/03/What-if-someone-steals-my-phone-110319.pdf.

[7] Riley Hughes. 4 keys to self-sovereign identity adoption. https://trinsic.id/4-keys-to-ssi-adoption/.

[8] Jim Isaak and Mina J. Hanna. User data privacy: Facebook, cambridge analytica, and privacy protection. *Computer*, 51(8):56–59, 2018.

[9] Xueping Liang, Juan Zhao, Sachin Shetty, and Danyi Li. Towards data assurance and resilience in iot using blockchain. pages 261–266, 2017.

[10] Pim Otte, Martijn de Vos, and Johan Pouwelse. Trustchain: A sybil-resistant scalable blockchain. *Future Generation Computer Systems: the international journal of grid computing: theory, methods and applications*, 107:770–780, June 2020.

[11] Johan Pouwelse. Trustchain protocol. 2018. https://tools.ietf.org/id/draft-pouwelse-trustchain-01.html.

[12] Quinten Stokkink, Dick Epema, and Johan Pouwelse. A truly self-sovereign identity system. 2020.

[13] Rabih Tout, Nicolas Lumineau, Parisa Ghodous, and Mihai Tanasoiu. Backup scheduling in clustered p2p network. pages 185–193, 2008.

[14] Tribler. Ipv8. https://github.com/Tribler/py-ipv8.

[15] Tribler. Trustchain super app. https://github.com/Tribler/trustchain-superapp.

[16] Dirk van Bokkem, Rico Hageman, Gijs Koning, Luat Nguyen, and Naqib Zarin. Self-sovereign identity solutions: The necessity of blockchain technology. 2019.