Towards Data Resilience for Fully Distributed Self-Sovereign Identity Managers

Kalin Kostadinov, **Martijn de Vos**, **Johan Pouwelse**
k.k.kostadinov@student.tudelft.nl, m.a.devos-1@tudelft.nl
Delft University of Technology

Abstract

1 Introduction

Every person on the Internet uses at least one digital identity. And service providers rely on them for building trust with their users. Unfortunately, the creators of the Internet have not designed a unified identity layer. Thus, service providers need to handle authentication and authorization themselves [1] which explains why every service 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 big concern for governments which has led to a large amount of research and regulations in the field [2]. There is a need for a novel identity management system, and its formal description stands in the middle of all the work [3]. It promises to not take control over an identity from its rightful owner and achieves this by satisfying the requirements for Self-Sovereign Identity [4]. SSI allows every identity holder to store and manage their data. For that, they need to use resources under their jurisdiction.

There are already several implementations that cover part of SSI's properties [5], 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. For these implementations to be adopted, they need to fulfill a long list of real-world usage requirements. Solutions generally fall into two groups.

The first one uses global consensus and requires the existence of a single data structure (blockchain). This structure contains information about all transactions in the network. Unfortunately, most real-world identity use cases require high throughput and low latency. However, the global consensus needs time and computing resources until nodes in the network agree upon the legitimacy of transactions. Also, in some cases, there needs to be support for offline transactions. Again, global consensus stands in the way because it gets resolved online. Thus, this group of identity managers is not well suited for solving the problem of adoption.

The second group of identity managers uses either local consensus or no consensus at all. These implementations generally satisfy real-world requirements for throughput and latency. Such systems are also fully distributed, 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.

There is a need for a solution to the data resilience problem of fully distributed SSI management systems. It will 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 Self-Sovereign Identity management systems data resilient?***

The remainder of this article has the following organization. First, section 2 formally specifies the underlying problem. Second, section 3 defines three possible solutions and assesses all their positives and negatives. Section 4 consists of a recommendation about the best solution and the technical details of implementing it. Then, section 5 goes over the reproducibility of the conducted research. Finally, section 6 discusses results, draws the main conclusions, and suggests ideas for future work.

2 Requirements for Data Resilience

Data resilience means that an identity holder should always be able to access his identifying information. Since users of SSI managers are obliged to handle the resources for hosting their identity managers, redundancy is the main component for achieving data resilience. It comes as a consequence of the fact that in fully distributed networks, nodes have vulnerable storage. To make a system resistant to data loss and corruption, a protocol keeping identities in at least two separate locations could solve the problem. An implementation of this supposedly looks like a backup system. However, storing identities calls for additional requirements to the ones traditional backup systems are satisfying.

Since a backup contains the whole transaction history of a user, the principles of Self-Sovereign Identity should also

hold for identity managers' backup protocols. The following requirements for SSI managers' backup systems arise.

- **Control.** It is of great importance where identity backups are stored because identity owners need to have full control over their data. Backup protocols should use storage that is under the jurisdiction of the identity owner. Next to that, encryption of backups should 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.
- **Access.** 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. Emergencies can happen at any time. Thus, identity owners should be able to recover their identity, no matter when or where they need to do it.
- **Transparency.** Backup protocols need to be transparent. Users should be explicitly aware of how their data is getting processed and where it is stored. If any such detail is unclear, we should not expect trust in the protocol, and regulatory agencies should prevent the adoption of those technologies. There already exist plenty of good examples when closed source systems have secretly stolen and misused identities.

tity mis-

- **Persistance.** Identity backups should reside in storage with a zero probability of loss or corruption. Backup protocols aim at allowing users to recover from lost access to their identity. In the case a backup gets lost or corrupted, the identity is lost as well. Then, the user will have to start collecting claims about himself from scratch.
- **Portability.** Identities should be able to exist without reliance on any third party. Thus, backups should be transferable to different backup systems or different instances of the same system. Otherwise, any vulnerability of a particular backup system's instance will expose users to the potential of identity loss or theft.
- **Interoperability.** Identities consist of a multitude of different claims. Backup protocols must ensure that all supported claim types by identity managers can be replicated and stored safely. If a backup system does not know how to store a particular type of claim, then, later, when needed, this claim will not be recoverable.
- **Usability.** Adding a backup mechanism to any system introduces increased complexity and some overhead. The design of identity managers usually considers different types of users since 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.
- **Legality.** 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 in sync. Transactions need to be stored at their backup location first before completing them and considering them legal. 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 one's identity.
- **Access Revocation.** Users can 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.

The following section lays out three possible solutions to the problem at hand. For all three of them, there is an evaluation, whether they comply with the above-mentioned requirements.

3 Three Emerging Solutions

In the previous section, it became apparent that data resilience is achievable through the addition of redundancy. And although a backup system might partially solve the problem at hand, no known backup protocol satisfies all requirements. Thus, in this section, three solutions are proposed. In the end, there is a recommendation for the one that meets most of the requirements.

3.1 Third-Party Storage Providers

The first solution is to use 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.

Furthermore, 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 many users and one software manages multiple backups.

However, cloud storage is vulnerable to cyber-attacks. Although security measures are at state of the art, one server is responsible for the data of multiple 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.

Another issue is that there are multiple identity management systems. Storage providers must know how to store each type of backup. It calls for an unnecessary development of backup protocols for the same identity manager by different storage providers.

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 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.

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 legality. 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 considering them legally valid. 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.

3.2 Peer-to-Peer Backup

The second solution is to reproduce the blockchain from the knowledge of other users about the lost identity. This idea emerged from the current developments in the field of peer-to-peer backup systems. The concept is to share an encrypted version of a user's identity with enough peers in the network so that the probability of ending up with an unrecoverable identity is as low as possible.

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 the 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 is always at least one available peer 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 previous transactions, 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.

This solution is highly portable and interoperable. The main algorithm can move backups between nodes. Furthermore, the actual backup gets created by the identity manager. Thus, it can create backups from all supported transaction types. 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, legality is a big problem. 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 considering legal the latest transactions.

At last, there is a need for a revocation mechanism. It will rely on a distributed hash table algorithm that gossips information about devices with revoked access to a specific identity if those devices are lost or stolen.

3.3 Identity Owner as Storage Provider

The third solution is to keep the Self-Sovereign Identity management system on a master server, controlled by the identity owner. Users manage their identities through remote devices like smartphones. The need for a central node under the jurisdiction of the identity owner will add some unwanted overhead. Furthermore, it hinders usability because users need to have the technical know-how to operate the server. However, they will have the most control over their identities. Also, this solution achieves transparency since users are solely responsible for their data.

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 a 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 identities should be stored somewhere else to allow recovery. Replicating them will add persistence.

This solution is highly portable and interoperable as well. Users have to install a fresh version of their identity manager on a new central server and recover the identity from the old server. Again, it works with all types of transactions because of the backup system's integration within the identity manager.

However, users cannot expect high availability. Depending on where identity holders are when they try to access their identity manager, the master server might not be reachable. In this case, transactions happen offline, so there is a need for

a caching algorithm that holds transactions before committing them to the server. This algorithm should account for legality as well. Thus, transactions should only become legally valid after their synchronization with the master server.

Remote 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. Such an access revocation function can be easily integrated with this solution since a blacklist can help the manager deny access to specific devices.

After an evaluation of all solutions, the third one looks more suitable for providing data resilience. The reason is that a central node might run on a machine connected to the wall. Such a device does not rely on battery size and network coverage. Thus, compared to a smartphone, it seems to have unlimited resources.

In conclusion, to implement the third solution, the following questions need to be answered:

- How to allow transactions when there is no access to the central node?
- How to deal with cached transactions when they do not get committed to the central node and get lost?

4 Implementation and Results

The Delft Blockchain Lab develops one of the Self-Sovereign Identity management systems, called IPv8 [6]. 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 identity loss. Every user has its blockchain, called TrustChain [7], 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. However, mobile devices are not reliable enough. Thus, it is not clear how users are supposed to recover their identities when access to them is lost. IPv8 falls within the group of SSI managers that use local consensus. Consequently, it suffers from the problem this paper is trying to solve. That is why we are using IPv8 as a platform to develop a solution for my research question.

Since we are using IPv8 as a base for improvements, we have explored its implementation in Kotlin for the super app [8], which the Delft Blockchain Lab is currently working on. Our goal is to assess the application and find a suitable approach for integrating our implementation.

I was previously involved in the development of another application that also uses IPv8 for storing COVID-19 immunity certificates [9]. The main objective of that project was to create a user-friendly GUI. If I implement the remote server solution, I can use the COVID-19 project as an experimental environment for my findings since it already uses REST API to communicate with the IPv8 deployment. It also relies on React Native [10] for its GUI. That means IPv8 can get an easy implementation for iOS, which the lab is lacking.

Depending on the protocol's design, it might be helpful to keep the core back-end that is going to run on the mobile device in Python, as is the implementation of IPv8 [11] itself. There is still no publicly available tool that efficiently builds Python applications for Android, iOS, and other mobile or embedded operating systems. Therefore, I might make use of my development, called Porthon. Its goal is to make Python applications portable and resource-efficient. Currently, there is a working version of the IPv8 implementation in Python for Android, and one for iOS is underway.

5 Responsible Research

6 Conclusions and Future Work

A further development 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 available. Those emergency "terminals" should only allow for verification of attestations.

References

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

[2] European Commission. Regulation (eu) 2016/679 of the european parliament and of the council. 2016.

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

[4] Christopher Allen. The path to self-sovereign identity. 2016.

[5] Dirk van Bokkem, Rico Hageman, Gijs Koning, Tat Luat Nguyen, and Naqib Zarin. Self-sovereign identity solutions: The necessity of blockchain technology. 04 2019.

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

[7] 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.

[8] Tribler. Trustchain super app.

[9] Tribler. Immune: Building a critical infrastructure for the nation-wide identification of recovered covid-19 patients.

[10] Facebook. React native.

[11] Tribler. Ipv8.