Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 176 (2020) 1531–1539

Procedia Computer Science

www.elsevier.com/locate/procedia

24th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems

# Self-Sovereign Applications: return control of data back to people

Sînică Alboaie[ab], Nicu-Cosmin Ursache[a]*, Lenuța Alboaie[a]

[a]*Faculty of Computer Science, Alexandru Ioan Cuza University, Strada General Henri Mathias Berthelot Nr. 16, Iași 700259, România*
[b]*Axiologic Research SRL, Strada Costachi Negri Nr.39, Iași 700071, România*

## Abstract

This article presents new solutions for the difficult problem of data protection. The discussion in our approach will be based on the premise that data needs to be shared between the adequate number of entities under a set of strict rules. In this context, our proposal consists of a suite of privacy and data control focused approaches that become the building blocks for Self-Sovereign Applications (SSApps). SSApps are a new class of applications proposed for the first time in this paper and are aimed to ensure data ownership and proper data control for people and companies. The building blocks for SSApps consist of the insights that the role of blockchain should be reduced to ensure data anchoring of data shared between multiple participants. The blockchain ensures integrity and data provenance but should not be used for storing any private data. Therefore, the data is stored in off-chain storages that are structured in the form of a new approach called Data Sharing Units (DSU or Dossier). A method for performant implementation of SSApps, Dossiers and anchoring is proposed in this article.



*Keywords:* Self-Sovereign Application; SSApp; Privacy; EDFS; Dossier; DSU;

## 1. Introduction

We live in a world that is going through a continuous digitization process. Nowadays, the number of companies doing business online is steadily increasing and people spend more time on different types of online platforms or accessing online services. This process generates a big volume of data as a consequence of the rising number of online interactions between parties (people, companies all together). The volume of data generated by these interactions is collected and stored in public or private databases in order to facilitate or continue interactions and

* Corresponding author. Tel.: +40-740-662-863.
*E-mail address:* cosmin.ursache@info.uaic.ro

1877-0509 
10.1016/j.procs.2020.09.164

most of the time, the ownership of data is misunderstood or lost through the process of interaction. For this reason, each entity (people and also companies) should have at their disposal tools and means to control what is happening with their data. Everybody needs to treat data with great responsibility as confidential data is as significant as money.

Business models like Freemium[24] are popular choices for a lot of companies that do their business online because it presents safe nets. Based on a model providing something free to establish a future collaboration or sale, it attracts a valuable user group. The user adoption and the data collected is helping companies to develop a better monetization mechanism [20]. The usage of the model does not present guarantees regarding the success of converting the user base into a premium one. As such, the focus is moved over to data monetized as a form of anti-conversion [20]. Of course the ethics, morality, how and if people get affected by this kind of processes remain open subjects for internet gossip.

There is a need for mechanisms capable of dealing with data privacy and ownership. Everybody should start to find ways to control what is happening with their data. Currently, there is no clear solution available to help keep track of generated data. There is no transparent, trustful and at the same time respectful privacy procedure to verify and manage how data is used and by whom.

There are attempts to tweak the normal "centralized" way of application development with GDPR implementations (in the European Union at least)[8][1] or data encryption [23][26]. This direction has an advantage that comes from the fact that companies do not have to invest too much resources and users will have their data protected to some degree [22]. In theory, for example, GDPR should help users who want to resiliate an online contract with their service provider and request it to be forgotten. GDPR is only a set of rules developed to ensure a good level of uniformisation across a wide variety of online contracts in a bigger variety of software solutions. The challenge presents itself to those that need to control that GDPR was properly implemented and that there are no violations.

Decentralized Applications (DApps)[11] represent a solution to gain owner control over data. DApps have really interesting properties like: they are open source (based on the need for autonomy and unanimous consensus), decentralized nature, incentivization and strong algorithms (for consensus and not only) and because they are built over the blockchain technology they are able to notarize everything [3]. Nowadays, many resources are put into developing DApps, thinking that companies or users will support the extra cost of the infrastructure and code execution involved [14]. Besides the extra cost, other issues emerge like dealing with secrets in the Blockchain world. In order to occur, the consensus and validation processes put pressure to keep most of the data public. This requirement is difficult to meet when taking care of important data like trade secrets. No matter what anonymization or encryption strategies are applied over the data stored in Blockchain, there will always be possibilities to correlate the notarized data with external one and to create data leaks [4]. This is undesirable and represents an obstacle to the widespread adoption of DApps.

Our research team proposes an innovative idea synthesized in the concept of Self-Sovereign Applications (SSApps). The following chapters not only present the concept in detail, but also show an implementation of SSApps ready to be used on a large scale offering all the benefits of DApps without the drawbacks. SSApps are instrumental in solving confidentiality issues for all kinds of user categories, citizens and companies.

## 2. SSApps versus DApps

A SSApp (Self-Sovereign Application) aims at giving control of data back to the user. SSApps do not make any difference between user types: companies or regular users. As users want to protect their private data, also do the companies that want to keep in a safe place their application logic or trade secrets. An instance of a SSApp and the application data is solely under the cryptographic control of the user that has the key. The key of the SSApp is represented by a concept, that we called Recovery Seed or simply Seed. The Seed represents the secret information needed in order to identify and retrieve an instance of an eWallet. An eWallet is an implementation of a digital wallet [18] extended to store every important bit of user information and Self-Sovereign Applications code. Self-Sovereign Applications represent improved versions of DApps. DApps are commonly implemented using Smart Contracts in Distributed Ledger Technologies (DLT) (e.g. Ethereum[25], Hyperledger[6] etc.). SSApps are solving problems that are related to DApps code execution and infrastructure costs, complexity of consensus processes and

dealing with sensitive data. One of the important aspects that make companies reluctant to use DApps on a large scale is represented by the fact that, in most circumstances, data has to be available to all DLT members in order for the consensus to be possible. Data Pseudonymization and data encryption does not properly solve the confidentiality issues because the leakage in the metadata can be used to deanonymize the smart contracts execution. SSApps resolve this issue by design because they are built using Secret Smart Contracts technologies [2].

SSApps inherit DApps' advantages like decentralized nature, incentivization and strong algorithms (for consensus and not only) and are built over Blockchain technology with focus on privacy. On top of the inherited properties SSApps bring other ones such us:

- SSApps work with encrypted user data: SSApps and their data are stored in an encrypted archive and all reads and writes operations are done only in a trusted execution environment. Data is kept secret from any person (e.g. employees of companies that provide cloud storage), except the owner, even if it is stored in the cloud
- SSApps run only on user trusted devices: a user can interact with a SSApp only after the application code and data arrives from an encrypted decentralized storage and get loaded in the trusted execution environment
- Once the SSApps are loaded into the user controlled device, they do not need to interact with any machine or server when offering functionality to the user

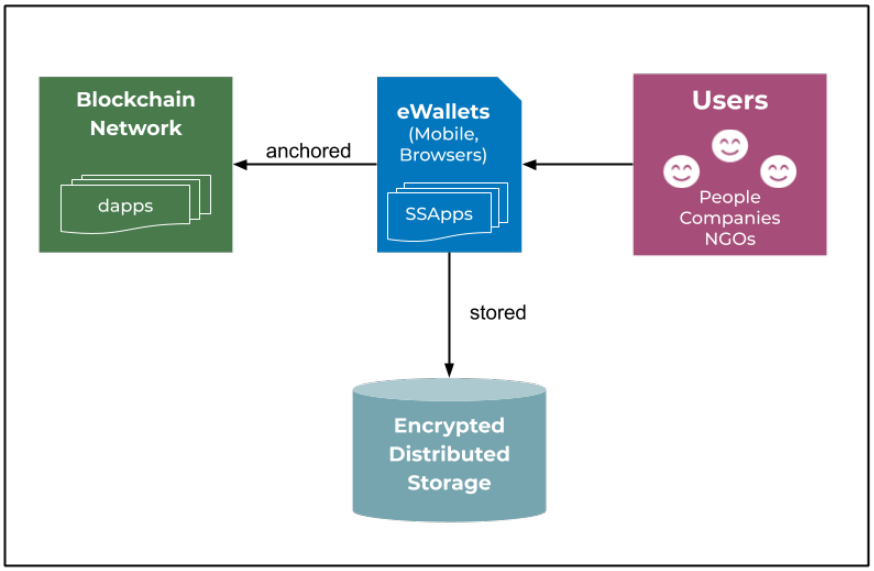


Figure 1. eWallets and SSApps interactions

eWallets can be anchored in private or public Blockchains(see Figure 1). As a result of the anchoring process, the eWallets and data stored in them can be notarized and at the same time data integrity can be ensured. An anchor can be identified by the Seed' hash. Technologies such as W3C Decentralized Identifiers (DIDs)[9] can be also used for identification purposes of anchors. As we will see further, SSApps implementations are possible due to a combination of technologies such as Blockchain, isolation and encapsulation execution mechanisms and encrypted distributed file systems.

## 3. SSApps - the underlying technologies and implementation

Self-Sovereign Applications development was made possible by using the technologies developed within the PrivateSky Research Project [16]. PrivateSky technology is capable of supporting SSApps development meant to be executed on client premises (mobile or web application) or on distributed computing infrastructures. As we will see in the next section, PrivateSky SSApp development framework offers instances of each component needed: encrypted distributed storage system, trusted execution environments and eWallets.

### *3.1. Encrypted distributed storage system*

#### *3.1.1. Storage mechanism using Bricks*

Our solution focuses on data privacy and for this reason it was needed to create a new decentralized file system that ensures data privacy by design. The backbone of our encrypted distributed storage system is represented by the implementation of a new type of archive, as an alternative to zip and tar archives, archives whose content is encrypted and stored in Cloud. This new type of archive is entitled BAR (Brick Archive). The main idea behind BARs is that each file has its content split in smaller fragments called Bricks. A Brick is a buffer (a collection of bytes compressed by a compression algorithm and encrypted with a secret encryption algorithm). BAR files are stored in the form of Bricks in a distributed storage system (any cloud provider infrastructure can be used for this). The Brick is referenced by the cryptographic hash of its compressed and encrypted content.

The file and folder structure of a BAR is stored in a BarMap (see Figure 2). BarMap is a special Brick containing a map (a reference) of the Bricks and the files from which these Bricks are part of. The BarMap also contains the Brick decryption keys needed to decrypt each Brick. When stored in the brick storage services, the BarMap looks like any other Brick, and for an intruder it is impossible to identify it. Each BAR file can be identified by a special key called Seed.

This ensures that data is stored in a secure and private manner. At this level each brick can be replicated on any number of nodes without the fear of somebody being able to understand what is stored in them. Also, by splitting the content we are able to retrieve composing bricks of files needed, on demand. These architectural decisions allow the possibility to load a high volume of data on mobile phones. For example, GDPR enforces patients control on medical data, but on the other hand the volume of data is not appropriate for mobile phones. The BAR technology finds the middle ground: the software treats the archive as a remote file system and will load only the necessary bricks.

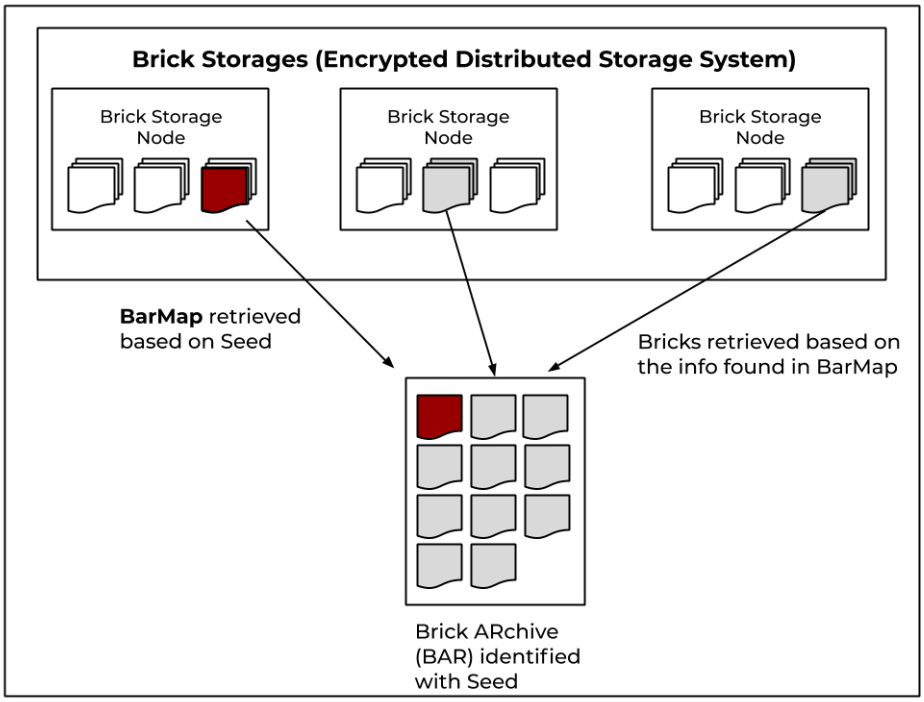


Figure 2. How a BAR gets reconstructed with Bricks from the storage system

Thanks to the way BAR was designed and implemented, it is impossible for someone to retrieve a BAR if one does not have the Seed. It is also impossible for an intruder to tamper with the content of a BAR and hide his tracks. From the developer's perspective, there is an API available that allows the following operations with the BARs: create a BAR, load/retrieve a BAR from the Brick Storage System, list files, write/read/extract files and folders from and to a BAR [5].

*3.1.2. Sharing data using Dossiers*

In the previous section we have presented a decentralized file system capable of ensuring data privacy. Next step is to offer mechanisms to share and reference data with stronger integrity properties. As a result, based on the BAR concept, the notion of Dossier was coined. A Dossier is a concrete implementation of a larger concept called Data Sharing Units (DSU). DSUs can be implemented in multiple ways and the Dossier concept is one such possible implementation. Our team is currently developing this concept further. At the implementation level, a Dossier is actually a BAR with a predefined internal structure (e.g. folders with defined purposes like storing blockchain data, Self-Sovereign Applications, user data et.al.). A Dossier contains a distributed ledger (PrivateSky blockchain [2]) that is used to validate the content of the Dossier, using secret smart contracts. Beside blockchain specific code, a Dossier contains the user interface that allows to modify the state of the Dossiers. This way, the user controls the whole application and the data. A different approach, but with a similar scope is Solid personal online data stores proposal [21].

Dossier can be viewed as a personal container, consisting of an application code and confidential data. A Dossier is a mobile container because it can be instantiated on Cloud or on edge devices (e.g. mobile phones, IoT, et.al.)

Data integrity and code integrity are assured by blockchain anchoring, making Dossiers a satellite blockchain described in [2]. Applications that run from Dossiers are called SSApp (Self-Sovereign Application) and represent a new breed of applications. From the technological point of view, SSApps are built as web applications, run without servers using Service Workers technology [19]. In a way, a Dossier can be imagined as a portable web server that can be instantiated easily on mobile devices and web browsers. In this setup, the Cloud servers are used only to store encrypted bricks and do not have access to the Dossier decrypted content.

Typically, a Dossier has a folder structure as follow:

- **Constitution**: contains a collection of code bundles that implements the business logic needed by the SSApp. It also includes code for integration with a public or private PrivateSky Blockchain deployment
- **Assets**: contains a collection of asset definitions needed by the internal blockchain
- **Transactions**: contains a collection of transaction definitions that implement operations over the available assets
- **Blockchain**: contains the history of the internal blockchain split in transaction blocks
- **Application**: contains the code of the Self-Sovereign Application View (HTML, CSS, JavaScript, images and other assets)
- **Data**: contains user data generated and needed by the SSApp

A Dossier inherits the list of APIs from BAR and on top of them it has the possibility of referencing to other Dossiers. This feature comes in handy when users share information among them. The referencing process is helping the owner of the referenced Dossier to maintain control over their data and to retract any rights granted to somebody else.

A Dossier can be anchored in a public or private Blockchain using a special DID Method [10] , called Anchor DID, in order to notarize every update and ensure data integrity. The Anchor will contain only public information about the Dossiers. Anchors can contain hashlinks [12], versions or timestamps.

eWalled is a special type of a Dossier. An eWallet is the starting point of navigating in the SSApps and Dossiers world (see Figure 3). Its security must be treated in a similar manner to the security of wallets for money or cryptocurrency.

The fact that a Dossier can have an internal private blockchain and can also be anchored in other public or private blockchains represents an innovative advantage. Basically, we can obtain hierarchical blockchains that will ensure data correctness and privacy [2]. For example, Dossiers can be used to export and manage private data from classic systems that need to get GDPR compliant. The internal blockchain can manage private data notarization by

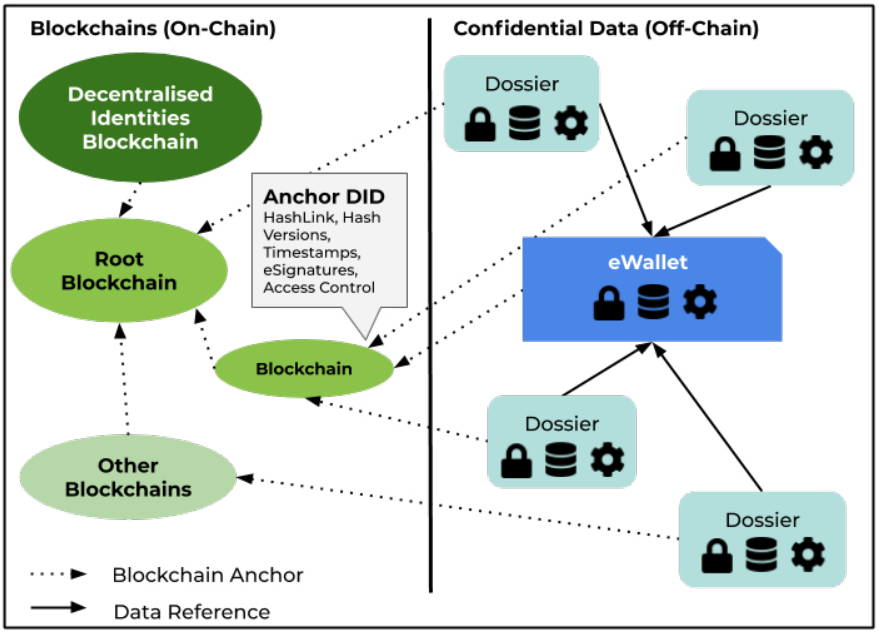


Figure 3. Dossier data referencing and anchoring

storing transactions that manipulate them and the public blockchain part can store information about the version index. The consensus process needed by the hierarchical blockchains is done by usage of the anchoring mechanism and only handles public data about each Dossier.

### *3.1.3. Encrypted Distributed File System (EDFS)*

Introducing a new type of archive, focused on the data privacy, presents some potential impediments to adoption. Inspired by other file systems that are in use nowadays we have designed and implemented the EDFS. EDFS is a file system designed to be used by BARs and Dossiers. When interacting with the APIs exposed by BARs and Dossiers, we actually interact with each internal EDFS instance. EDFS handles files, folders, bricks, BarMaps and the encryption/decryption algorithm of a BAR. Once a BAR or Dossier gets loaded into a trusted execution environment, the internal EDFS will work with the sensitive data and ensure that the privacy is assured. EDFS is responsible for starting transactions in the blockchains, where the Dossier is anchored, in order to update the anchors accordingly to the content changes.

Dossier refferenciation is handled at EDFS level through mount and unmount features. The mounting and unmounting processes found in EDFS are somehow similar to those found in Linux file systems. Mounting a Dossier into another one is implemented by referencing the internal EDFS of the targeted Dossier into the destination one. Each mounting point created during the mounting process is stored internally as a collection of information, composed by the internal destination path, name of the mounting point, the DID of the targeted Dossier containing the key that allows read and/or write operations. Once a Dossier is mounted, the internal EDFS file systems are “linked” and users can interact with data found in those two or more Dossiers seamlessly.

## *3.2. Trusted Execution Environment*

The security of Dossier and SSApps is based on the idea that a user will load these components in a sandboxed environment that we call Trusted Execution Environment (see Figure 4). Our current implementations are using isolates (a node.js technology, see section 3.2.2) and Service Workers (see section 3.2.1). Other implementations of the Trusted Execution Environment are also possible when the type of dossiers require different security or performance characteristics.

*3.2.1. Service workers*

Our web SSApp implementation proposal for a trusted execution environment aims at using the power and capabilities of service workers available in all modern browsers. Service workers' initial purpose was to ensure that users can interact with a web page even when the Internet is out of reach [19]. Service workers provide a Web Worker context that can be created by request at runtime. The implementation consists of creating an event-driven worker capable of handling navigation requests. When a worker is instantiated, it is registered with an origin and pattern. In this way, the worker is called each time a request is generated for that origin and matches the pattern. The communication with a service worker is done by sending and receiving events for each network request. In other words, a service worker becomes an intermediary actor for all the network requests. Service workers can reply with a response based on a cache system or can pass through the request to the network.

PrivateSky implementation of Self-Sovereign Applications is based on the features and capabilities of service workers. PrivateSky SSApp technology empowers service workers by allowing them to work with data stored in Dossiers and BARs. By doing that, service workers will act as full servers that run verified code stored in a Dossier's Constitution folder in a controlled and isolated environment.

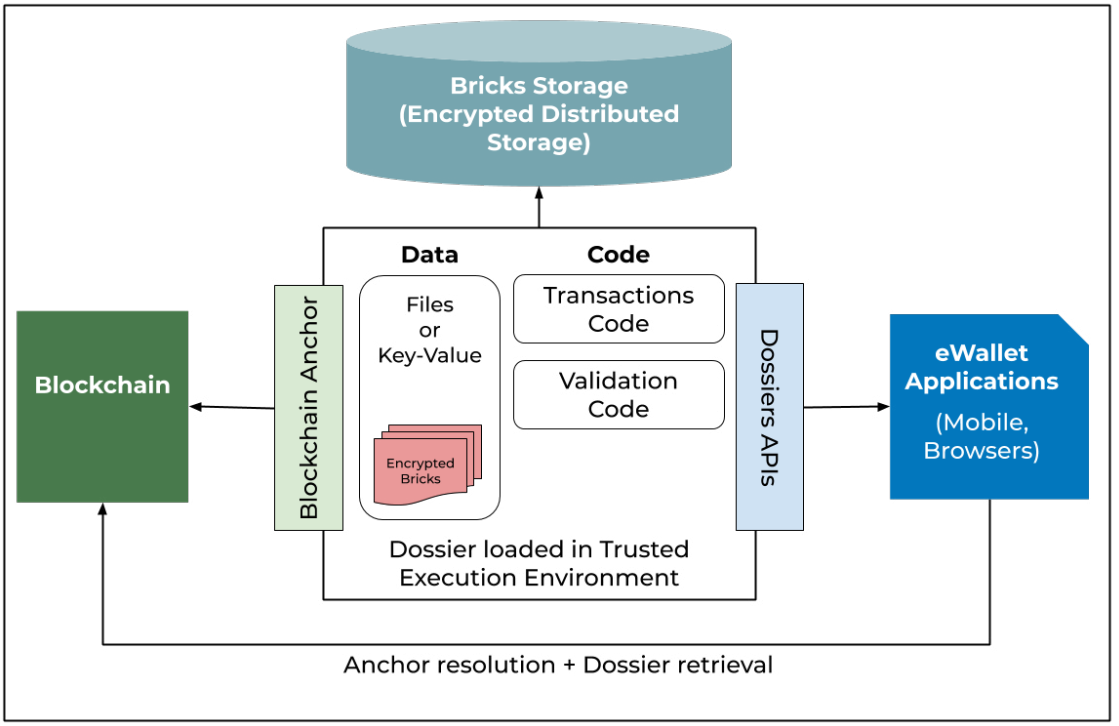


Figure 4. Dossier loaded in a Trust Execution Environment and interactions with Blockchain and Bricks Storage

*3.2.2. Isolates*

Self-Sovereign Applications can also be executed in server environments by using the isolation V8's capabilities to implement a trusted execution context. V8 engine on top which Node.js is built can execute code in an isolated context [15]. Isolate can be seen as a mechanism for v8 to create "sandboxes" and allocate memory to the code running inside them [13]. This is achieved by isolating an instance of the V8 engine, including garbage collection, code, context and state, and making it available to just one thread handler at a moment in time.

PrivateSky also provides the software infrastructure needed in order to be able to boot multiple v8's Isolates. Each Isolate has the capability to retrieve and load a Dossier from the Brick Storage System once it receives a Seed. There is a specific boot protocol followed that ensures the Isolate loads only the code available in the Dossier constitution. Once the booting process completes the Isolate runtime is available to be used by a Dossier's SSApp and can be executed.

### *3.3. Cardinal Framework*

The user interface of the SSApps can be created with HTML, Javascript and CSS technologies. Any particular single page framework (like React or Angular) could also be used. However, our consideration regarding security, made us create a light MVC framework that is called CardinalIJS(or simply Cardinal)[7] and uses Web Components. The usage of Web Components is trying to reduce the number of dependencies that a SSApp will rely upon and therefore reduce the attack surface. The HTML programming model provided by the web components technology offers the possibility of creating custom tags as reusable components. Cardinal has a collection of reusable components that cover the basics of web application development targeted for SSApps development.

Cardinal provides a mature and robust programming framework based on MVC and MVVM. In Cardinal the models are loaded and set up by controllers. If required, the Controller is setting up a View Model (an adapted model easier to connect to the View). The View part is accomplished by describing the pages or screens of the application by using the reusable components provided by Cardinal. It also offers a strong binding mechanism that can be used when trying to tie the View with the data models. To reduce the dimensions of the controllers hierarchy Cardinal framework is pushing the event driven communication model to be used between the view components.

Therefore, Cardinal is introducing a programming mode (it is a micro framework) oriented to allow the rapid development of WEB applications (RAD - Rapid Application Development or low code development). It can also be used in simple progressive web applications. Cardinal appeared as an alternative to other MVC frameworks that come bundled with too many dependencies and potential security issues.

## 4. Conclusions

Our team is currently working with seven companies to build applications based on the presented technologies [17]. The SSApp concept is very appealing for Software as A Service solutions because it offers a technical method to relieve the software providers from GDPR regulations and other privacy related issues.

In this paper we have presented our research results regarding Self-Sovereign Applications. SSApps represent an innovative and sustainable concept that will help people or companies to keep their data private without making any compromises when it comes to data sharing, data integrity, history notarization and keeping full control of how data is shared and used. We have presented how the suite of developed technologies is capable of supporting SSApps development from ground up, covering all the basics in order to ensure full data privacy at every level.

In this manner, after years of research and development, through our Self-Sovereign Applications proposal, the data ownership is not only at a declarative level and SSApps ensures that the control of data is given back to people.

## Acknowledgments

This work is partially supported by POC-A1-A1.2.3-G-2015 program, as part of the PrivateSky Research Project- “Experimental public-private partnership development to create local cloud platforms with advanced data protection features” (P_40_371/13/01.09.2016). Project co-financed from the European Regional Development Fund under the Operational Program Competitiveness 2014-2020.

This work is partially supported by the Ministry of Research and Innovation within Program 1 – Development of the national RD system, Subprogram 1.2 – Institutional Performance – RDI excellence funding projects, Contract no.34PFE/19.10.2018

## References

[1] Alboaie, Lenuța (2017), Towards a smart society through personal assistants employing executable choreographies, At 26th *International Conference on Information Systems Development*

[2] Alboaie, Sînică, Alboaie, L., Pritzker, Z. & Iftene, A. (2019) Secret Smart Contracts in Hierarchical Blockchains. In A. Siarheyeva, C. Barry, M. Lang, H. Linger, & C. Schneider (Eds.), *Information Systems Development: Information Systems Beyond 2020 (ISD2019 Proceedings)*

[3] Antonopoulos, Andreas M., and Gavin Wood. (2018) Mastering ethereum: building smart contracts and dapps. *O'reilly Media*

[4] Atzei, N., Bartoletti, M., & Cimoli, T. (2017, April). A survey of attacks on ethereum smart contracts (sok). In International conference on principles of security and trust (pp. 164-186). Springer, Berlin, Heidelberg.

[5] Brick Archive (BAR) (2020), https://github.com/PrivateSky/bar Accessed March 1, 2020
[6] Cachin, C. (2016). Architecture of the hyperledger blockchain fabric. In Workshop on distributed cryptocurrencies and consensus ledgers (Vol. 310, p. 4).
[7] CardinalJS, (2020) https://github.com/PrivateSky/cardinal Accessed March 1, 2020
[8] Cavoukian, A., and Jutla, D. (2014): Privacy Policies Are Not Enough: We Need Software Transparency
[9] Decentralized Identifiers (DIDs), (2019) https://www.w3.org/TR/did-core/ Accessed March 1, 2020
[10] DID Method Registry, (2020) https://w3c-ccg.github.io/did-method-registry/ Accessed February 29, 2020
[11] "Ethereum whitepaper", (2020) https://github.com/ethereum/wiki/wiki/White-Paper#applications Accessed March 1, 2020
[12] IETF specification for cryptographic hyperlinking, (2019) https://github.com/w3c-ccg/hashlink Accessed March 1, 2020
[13] Kelly, Nicholas. "Node. js Asynchronous Compute-Bound Multithreading."
[14] Luu, L., Chu, D. H., Olickel, H., Saxena, P., & Hobor, A. (2016). Making smart contracts smarter. In Proceedings of the 2016 ACM SIGSAC conference on computer and communications security (pp. 254-269).
[15] "Node Isolate", (2020) https://v8docs.nodesource.com/node-0.8/d5/dda/classv8_1_1_isolate.html Accessed March 1, 2020
[16] PrivateSky Research Project, (2016-2021) https://github.com/privatesky Accessed March 1, 2020
[17] PrivateSky Research Project contributors list, https://profs.info.uaic.ro/~ads/PrivateSkyEn/noutati/#tab-1-3 Accessed March 1, 2020
[18]Sahut, Jean-Michel. (2008). The Adoption and Diffusion of Electronic Wallets. *Journal of Internet Banking and Commerce*. 13.
[19] Service Workers, https://www.w3.org/TR/service-workers/ accessed March 1, 2020
[20] Seufert, E. B. (2013). Freemium economics: Leveraging analytics and user segmentation to drive revenue. Elsevier.
[21] Solid POD, (2020) https://solid.inrupt.com/ Accessed March 1, 2020
[22] Tankard, C. (2016). What the GDPR means for businesses. Network Security, 2016(6), 5-8.
[23] Voigt, P., & Von dem Bussche, A. (2017). The eu general data protection regulation (gdpr). A Practical Guide, 1st Ed., Cham: Springer International Publishing.
[24] Wilson, F. (2006). The freemium business model. A VC Blog, March, 23, 201.
[25] Wood, G. (2014). Ethereum: A secure decentralised generalised transaction ledger. Ethereum project yellow paper, 151(2014), 1-32.
[26] Zerlang, J. (2017). GDPR: a milestone in convergence for cyber-security and compliance. Network Security, 2017(6), 8-11.