applied sciences

MDPI

Article

Towards a Catalogue of Self-Sovereign Identity Design Patterns

Špela Čučko *, Vid Keršič and Muhamed Turkanović

Faculty of Electrical Engineering and Computer Science, University of Maribor, Koroška cesta 46, 2000 Maribor, Slovenia
* Correspondence: spela.cucko@um.si

Abstract: Self-sovereign identity (SSI) is a user-centric, decentralised identity approach that provides a means for identification, authentication, and authorisation without the involvement of external entities, responsible for identity provisioning and management in current centralised and federated approaches. In general, the basic building blocks of an SSI system include decentralised identifiers, verifiable credentials, identity wallets, a verifiable data registry, and three main actors: issuer, identity holder, and verifier. Even though the SSI field is dominated by proposals, SSI systems can be implemented in different ways, which is reflected in the absence of a well-defined architecture. Thus, the best implementation is still a matter of research, the requirements of the individual system, and its field of application. However, well-designed and implemented systems are crucial to avoiding failures, speeding up the development process, ensuring high quality, and the broader adoption of SSI solutions. Hence, the main objective of this study was to identify design patterns and good practices of the SSI ecosystems by reviewing and analysing the literature, technical documentation, and existing SSI implementations. Therefore, the study is built on existing knowledge, and presents a comprehensive catalogue of thirty-five SSI design patterns that can serve as a starting point for a possible SSI system design.

Keywords: self-sovereign identity; decentralised identity; design patterns; decentralised identifiers; verifiable credentials; blockchain; verifiable data registry; identity wallets

check for updates

Citation: Čučko, Š.; Keršič, V.; Turkanović, M. Towards a Catalogue of Self-Sovereign Identity Design Patterns. *Appl. Sci.* **2023**, *13*, 5395. https://doi.org/10.3390/app13095395

Academic Editor: Konstantinos Demertzis

Received: 7 March 2023
Revised: 27 March 2023
Accepted: 23 April 2023
Published: 26 April 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Self-sovereign identity (SSI) is a novel identity approach that presents a shift from current, centralised, and federated identity models to models following the principles of decentralisation and user-centrism. It provides a means for identification, authentication, and authorisation without the involvement of external entities, such as identity providers and certificate authorities, responsible for identity provisioning and management in current approaches. Therefore, users are central to digital interactions. They are responsible for obtaining, storing, managing, and distributing their identity data.

The concept arose with the growing interest in the shift towards decentralisation, specifically after the successes of blockchain technology, which was presented as a promising solution in the field of identity management for solving problems such as data centralisation, single point of failure, lack of data control, anonymity, privacy and security vulnerabilities, etc. Although the SSI approach solves many problems, new challenges have emerged that are nowadays being addressed by numerous researchers, projects, companies, and organisations, such as the World Wide Web Consortium (W3C), the Decentralised Identity Foundation (DIF), The Sovrin Foundation, and Trust Over IP (ToIP), which are striving to define the scope, standardise the protocols and artefacts included in the SSI ecosystem, and validate it through proofs-of-concept (PoC). Even though the concept is gaining traction and interest in academia and industry, the SSI field is dominated by proposals. While the idea is well known, it can be implemented in different ways, which is reflected in the absence of a well-defined architecture. Therefore, the best implementation is still a matter of research, the requirements of the individual system, and its field

of application. Consequently, many studies propose and present systems/solutions, IT architectures, and frameworks, and address authentication, security, trust, and privacy. On the other hand, there are almost no studies addressing usability, user experience, patterns, and best practices [1]. There is a lack of systematic architectural design for SSI systems that can support methodical development, which can potentially lead to security risks and vulnerabilities in the systems [2]. Well-designed and implemented systems are crucial to avoid failures and to the broader adoption of SSI solutions in general. Hence, the main objective of this study was to identify design patterns and good practices of the SSI ecosystems. The study presents a catalogue of thirty-five SSI design patterns that can serve as a starting point for a possible SSI system design. The catalogue is based on a literature review, as well as a technical documentation review, which focused on existing SSI implementations and best practices.

The remainder of this paper is structured as follows. Section 2 outlines the background and related work and provides knowledge on the main concepts and technologies needed for understanding the work. It includes several sections describing SSI concepts such as actors, process flows, building blocks, etc. (Section 2.1) defining the concept of software design patterns (Section 2.2), and presenting related work on design patterns that formed the starting point of our study (Section 2.3). Section 3 presents the main contribution of the research. It defines the objectives (Section 3.1) and the research methodology (Section 3.2). It presents a summary of the identified design patterns and their classification (Section 3.3), while also including a representative example of a chosen pattern (Section 3.4). Finally, Section 4 provides a discussion, and Section 5 presents the conclusions, possibilities for extending and improving the study, and opportunities for future work.

2. Background and Related Work

2.1. Self-Sovereign Identity

Self-sovereign identity (SSI) is a decentralised, user-centric approach to digital identity management that enables the actions of identification, authentication, and authorisation [3], without the need for external intermediaries [4] responsible for identity provisioning and management in the current centralised and federated approaches, where the power of control resides at the service or identity providers, as presented in Figure 1, where the blue colour represents an entity with control in a particular interaction. On the other hand, within the SSI, this control is shifted to each involved entity, while interactions can be based purely on peer-to-peer connections between the involved entities, who are the only ones accountable for generating, storing, and managing identifiers and correlated cryptographic keys, as well as acquiring, storing, and sharing their credentials that contain identity attributes that other entities attest to. The main advantage of the approach is that it enables entities (individuals/users and organisations) to self-create, control, and manage their identifiers and associated attributes, i.e., identity data [5]. At the same time, it allows controlling the flow of these data during digital interactions, increasing security and privacy by using cryptographic mechanisms.

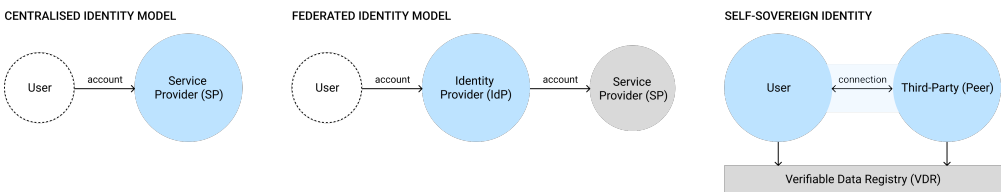


Figure 1. Identity models [6], representing the shift of control (e.g., data, process).

2.1.1. SSI Actors and Process Flow

Within the SSI approach, users do not need to create an account with each service provider (SP), nor is there a need for an identity provider (IdP) to provide the user with an identity and its management, as identity data are stored in the user's domain in digital wallets that are under the user's full control. Similarly to other identity models, we can

differentiate between three actors, namely, the *credential issuer*, the *identity holder*, and the *credential verifier*, as depicted in Figure 2. Each entity can act in a different role, depending on the context. The issuer attests to particular attributes of the holder by issuing and signing a verifiable credential (VC) digitally, containing one or multiple attributes or claims related to the holder responsible for retrieving, storing, managing, and presenting the acquired identity data with the verifier, who usually requires proof of identity to identify, verify, and ultimately allow access to the desired services. Therefore, the holder is at the centre of digital interactions, deciding when, with who, and what data he/she is willing to share. Thereby the verification process does not require the involvement of the issuer, as trust can be established by the utilisation of *decentralised identifiers (DIDs)* and *DID documents* holding the public keys of the holder, cryptographically signed *verifiable credentials (VCs)* and *verifiable presentations (VPs)*, *verifiable data registry* (VDR), and *identity wallets*, which are essential components of SSI architectures.

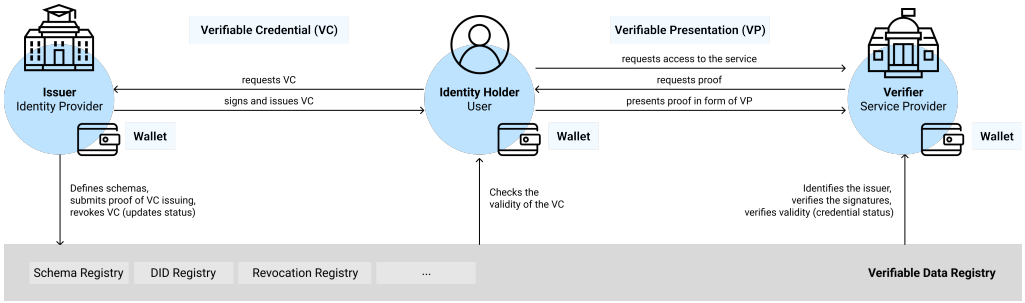


Figure 2. Interactions and data flow between issuer, holder, and verifier.

2.1.2. SSI Building Blocks

A *decentralised identifier (DID)* is a globally unique, persistent digital identifier, also classified as a uniform resource name (URN), that enables a cryptographically verifiable, decentralised digital identity. It refers to a DID subject as determined by a DID controller. Each DID is represented as a URI (did:<DID method>:<method-specific identifier>) that associates a DID subject with a DID document, outlining the DID subject, cryptographic public keys, authentication mechanisms, service endpoints, and other mechanisms used to verify the ownership of the DID. DIDs are entirely independent of centralised registries, external entities or IdPs, and allow registration (create), modification (update), resolution (read), and revocation (delete) operations to be carried out independently, without centralised registration, authentication, and authorisation. Thereby, CRUD operations depend on the use of a specific DID method. They enable the creation of unique, private, and secure connections between entities. Each entity can create, control, and manage any number of DIDs fully (with the same or different DID method), and use them separately in different digital interactions and contexts, preventing data correlation [7].

A *verifiable credential (VC)* is a standardised data model for cryptographically-verifiable digital credentials as defined by the W3C verifiable credentials specification. It contains metadata (such as issuer, credential type, credential limitations, etc.), identification information, a set of assertions or attributes describing the credential subject, and information on the mechanism of verification and revocation. The VC is signed digitally by the issuer, whose signature confirms the authenticity of the data in the VC. The use of cryptographic mechanisms protects the VC against tampering and enables cryptographic verifiability [8]. VC holders can generate *verifiable presentations (VPs)* from one or multiple VCs, and share them with verifiers to prove the identity and identity attributes. Thereby, privacy can be maintained using methods such as selective disclosure and zero-knowledge proof (ZKP). Selective disclosure enables the construction of a VP containing only a subset of attributes from one or more credentials, while ZKP allows attribute proofing without disclosing the values.

Digital identity wallets are a means for users to store, control, and manage their digital identities and identity attributes. They represent an essential user interface between end

users and the decentralised infrastructure, and play an important role in user identification, authentication, and authorisation. Wallets are portable, secure personal repositories, usually in the form of a mobile, desktop, or cloud wallet. All of these are based on software applications and an encrypted database where users (identity holders/controllers) store identifiers, cryptographic material (private keys), digital credentials, and other sensitive, private data. In addition to storage, such wallets usually support data acquisition, verification, management, and sharing [9], while recovery and backup represent a desired functionality. Various companies and governments endeavour to develop SSI wallets, and many are already available, e.g., VIDwallet, Trinsic Wallet, esatus Wallet, Connect.Me, Gataca, Lissi Wallet [10], SSI Snap [11], etc. However, they are not mature yet, and differ in implementation, user experience, and supported functionalities. On the other hand, the concept of digital wallets is not new. For many years they have been a secure means of digital payment, and, consequently, a replacement for physical wallets [12]. They are primarily financial applications for storing payment information securely (e.g., credit and debit card or bank account details) and assets, carrying out transactions, and tracking payment history using portable devices such as smartphones, tablets, and smartwatches. As the technology has evolved, digital wallets for the management of cryptocurrencies or cryptographic tokens [13] and, later, digital identity wallets, have emerged as a means for users to control and manage their digital identities.

The *verifiable data registry (VDR)* acts as the backbone of a decentralised public-key infrastructure (DPKI), and establishes trust between different entities within the SSI ecosystem. It functions as the single source of truth, eliminating dependencies on centralised registries, and can be implemented using blockchain, distributed ledger technology (DLT), or other decentralised systems. However, it should be noted that nothing prevents service or identity providers from proposing centralised systems (e.g., GitHub), as long as the entities (i.e., individuals or organisations) are willing to use such centralised registries within a decentralised-based ecosystem. In the context of SSI, VDR is utilised commonly for storing (i) public DIDs, (ii) VC schemas and definitions, (iii) VC state information, and (iv) evidence of interactions between entities (e.g., evidence of VC issuance and acceptance). The VDR can serve as a replacement for the centralised registration authority in traditional identity management systems, providing a DPKI [14] and storing the mapping between an identifier and an authentication method [15], as multiple DID methods rely on the VDR to resolve DID to DID documents [16].

2.2. Software Design Patterns

In software engineering, patterns are tools or means for architects and engineers to design robust and well-designed systems or applications [17]. In general, they are reusable solutions to commonly occurring problems within a given context in software design, captured in a way that allows their successful reuse. They are not a specific piece of code, but, rather, are a general concept of solving a particular problem. The principle of patterns originated as an architectural concept proposed by Christopher Alexander et al. [18]. They described in detail several patterns that serve as generic guiding principles for architectural design, and suggested documenting architectural plans, from regional plans to interior design [19]. They defined a pattern as a description of a problem that occurs repeatedly in our environment, accompanied by a description of the solution to that problem in such a way that this solution can be used a million times without performing it the same way twice [18]. Their book, entitled *A Pattern Language*, is also credited with inspiring the development of object-oriented programming (OOP) languages [19] and advances in software engineering, where patterns emerged in 1987 when Kent and Cunningham presented the idea of applying patterns to OOP using Smalltalk [20]. The rise of the patterns approach in the field of computer science arose after 1994, when a group of researchers, Gamma et al. (usually called the Gang of Four—GoF), published the book *Design Patterns: Elements of Reusable Object-Oriented Software*, defining a collection of design patterns for developing object-oriented software [21]. Since then, the number of design

patterns, their popularity, and their application has increased drastically, as software engineers have recognised their advantages [22], which are outlined as follows. (i) Existing patterns present a toolkit of solutions for solving common design problems. They are usually tested extensively and used widely by other developers. Therefore, they provide tested and proven paradigms. (ii) They can be reused in new designs as the best possible solution for a particular problem, as they capture best practices and lessons learned during software development, speeding up the development process. (iii) They define a common language between developers, as they are defined by an expressive name allowing effective communication [17,22]. (iv) They are crucial in forward engineering, where their use increases quality, readability, and documentation, and in reverse engineering, where they help to understand the software and the rationale behind solutions [22]. However, despite these benefits, their use must be evaluated carefully and not taken for granted. Using the wrong pattern, inappropriate for a specific use case, can be detrimental and increase the complexity of the software [17].

Patterns can be grouped into three categories: (i) architectural patterns, (ii) design patterns, and (iii) idioms. (i) Architectural patterns define the general structure of applications, such as elements and the connections between them, and represent the highest level of abstraction. (ii) Design patterns define the way modules, classes, or components are organised to solve problems, while (iii) idioms represent code-level solutions to problems related to a specific programming language [17].

Patterns are formalised best practices, described in a way that allows common design problems to be solved without reinventing the wheel. They are documented in a way that allows their reuse in numerous situations, often using a pattern template. However, various template formats have been used for defining patterns by different authors, the most prominent being the GoF pattern format proposed by Gamma et al. and the template proposed by Alexander et al. Both approaches encapsulate an expressive *name*, a *context*, and a recurring *problem* that the pattern solves. Additionally, Alexander's template includes a *solution*, *forces* affected by the pattern, *examples* of application, the *resulting context*, a *rationale* for deep or complex aspects, *related patterns*, and *known uses*. On the other hand, the GoF format encloses noncompulsory *classification*, an attribute called *"known as"* in case the pattern has various names, *motivation* for use, *applicability* to encompass situations in which the pattern can be applied, *participants* and their *collaborations*, the *structure* of the pattern, *consequences* of its use, an *implementation* part describing the code example and *technical aspects* that have to be considered, *known uses*, and *related patterns* [17].

2.3. Related Work

Numerous researchers have proposed design patterns for various application domains, such as web development, game design, microservices, the Internet of Things (IoT), blockchain, etc. Gašparič et al. [23] identified 12 existing catalogues of architectural and design patterns applicable to blockchain-based applications, obtained through a literature review, and presented their catalogue proposal. They categorised 97 patterns into five categories, namely, data, transactions, security, architecture, and business logic and management. Six et al. [17] identified 120 unique blockchain-related patterns collected from a systematic literature review of twenty research papers. They proposed a pattern taxonomy, classifying patterns into on/off-chain interaction patterns and on-chain patterns, including smart contracts, data management, and domain-based patterns. Five patterns addressing decentralised identity were identified among the latter. Liu et al. [2,14] proposed twelve design patterns for blockchain-based SSI systems, intending to help architects understand and apply the SSI concept to system design easily. They identified the main SSI objects and defined their lifecycle. Building on these foundations, they presented each pattern thoroughly [14], connecting it to a lifecycle state or transition and classifying it into one of three categories, i.e., key management (Master and Sub-Key Generation, Hot and Cold Wallet Storage, Key Sharding), DID management (Identifier Registry, Multiple Registration, Blockchain and Social Media Account Pair, Dual Resolution, Delegate List),

and credential design (Selective Content Generation, Time-Constrained Access, One-Off Access, Blockchain Anchor), which also coincided with the identified objects. However, besides the aforementioned papers, we did not identify any other relevant research which focuses specifically on SSI patterns. Even though the results of the papers introduce the starting point for discussing a comprehensive catalogue, they still lack common SSI design patterns already in use today, and some of the patterns outlined are not presented well enough, and have to be corrected. Furthermore, the studies lack a methodological approach, as these do not provide a methodology for obtaining and defining the patterns, and focus solely on patterns for blockchain-based SSI solutions. Although blockchain initiated and accelerated the development of decentralised identity, it is not necessary for the implementation of SSI solutions. As previously mentioned, instead of blockchain, distributed ledgers, decentralised file systems, different database systems, peer-to-peer networks, and other forms of decentralised trusted data storage [24] can be utilised as a verifiable data registry. Moreover, some DID methods [16], such as DID:peer and DID:key, allow for establishing DID connections and exchanging data independently of any decentralised system [25], which should be taken into account and reflected in the collection of SSI patterns.

3. Self-Sovereign Identity Design Patterns and Their Classification

The Systematic Mapping Study (SMS) [1] identified a research gap in the field of SSI, indicating a lack of studies addressing usability and user experience, patterns, and design practices that are crucial for the faster, streamlined development of quality SSI solutions. Only a few works focus on SSI design patterns. They focus solely on blockchain-based SSI solutions, not considering the possibility of utilising other VRDs. Moreover, the studies lack a methodological approach, as they do not provide a methodology for obtaining and defining the patterns. Therefore, we conducted research with the aim of enhancing the existing pattern collection by providing a comprehensive, improved *catalogue of SSI design patterns* that could serve as a starting point for the architects and engineers responsible for designing high-quality solutions.

3.1. Objectives

The goal of this study was to investigate existing implementations of SSI solutions and review the literature (i.e., white) and technical documentation (i.e., grey literature) to identify design patterns and good practices of SSI systems. We wanted to answer the following research questions: (RQ1) *Which SSI design patterns are defined in the existing literature?* (RQ2) *What is the classification of existing patterns?* (RQ3) *Are already defined SSI design patterns presented correctly?* (RQ4) *Can we identify additional design patterns (and good practices) of SSI systems by analysing technical documentation?* (RQ5) *In which classification categories can design patterns be placed?*

3.2. Methodology

To verify that no further progress has been made in this area in the years since the SMS was conducted in 2021, and that no additional research has been carried out, we searched the digital databases IEEE Xplore, Science Direct, and ACM DL, using the search string *("Self-Sovereign Identity" OR "Self Sovereign Identity" OR "Decentralised Identity" OR "Blockchain Identity") AND ("Design Pattern" OR "Design Patterns" OR "Design Principle" OR "Design Principles" OR "Good Practice" OR "Good Practices")*. Thus, in the initial phase, we conducted a literature review which resulted in the identification of two sources, from which we obtained a set of twelve SSI design patterns [2,14]. Based on preliminary research and existing knowledge, we identified shortcomings in the defined patterns and recognised the possibility of expanding or supplementing the set of patterns. Consequently, a review and analysis of the technical documentation followed. We focused on the extensive European Self-Sovereign Identity Framework (ESSIF) documentation by the European Blockchain Services Infrastructure (EBSI) [26], Decentralised Identity Foundation (DIF) [27] and the World Wide Web Consortium (W3C) [7,8,16] documentation and standards, and, later, the

Veramo [28] and Sovrin documentation [29]. The review and analysis were carried out in March and April 2022, and resulted in the identification of an additional twenty-three patterns. We synthesised the findings carefully and defined all the patterns based on the Alexander template [17]. For each identified pattern, we defined (i) name, (ii) summary, (iii) context, (iv) problem, (v) forces, (vi) solution, (vii) example of application, (viii) resulting context, (ix) related patterns, and (x) known use cases. Additionally, each pattern was also classified into a representative category. The process described is visualised in Figure 3.

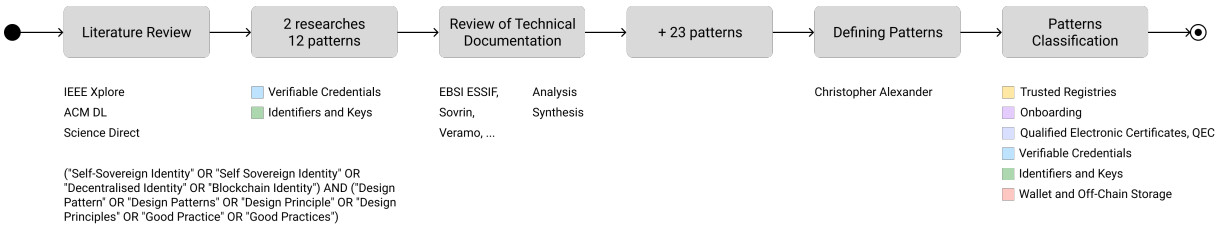


Figure 3. The steps we followed in the SSI design pattern identification, definition, and classification process.

3.3. Patterns and Their Classification

In this section we present an extended SSI pattern collection consisting of 35 design patterns, whereby each is summarised in Tables 1–6. Additionally, we also analysed and classified the synthesised patterns into six representative categories, namely, (1) *Trusted Registries*, (2) *Onboarding*, (3) *Verifiable Credentials and Presentations*, (4) *Decentralised Identifiers and Cryptographic Keys*, (5) *Qualified Electronic Certificates*, and (6) *Wallet and Off-Chain Storage*. The objective of the categorisation was to generalise the patterns and group them in order to enable a better understanding of patterns, i.e., how these complement each other, or how these are connected to each other.

In the context of research, each pattern was defined according to the Alexander template [17]. Due to the length of our output, only summaries of the patterns are presented below, with the complete documentation available at [30].

Moreover, to present and visualise better where the summarised patterns come into play and can be applied, we also connected these with the life-cycle states and transitions of main SSI objects depicted in Figure 4.

The category *Trusted Registries* (Table 1) combines patterns that enable the establishment of trust between different entities, i.e., holder, issuer and verifier, in a decentralised identity ecosystem, which is crucial in digital interactions. It represents one of the key elements for removing centralised intermediaries (i.e., identity providers, certification authorities, etc.) in the verification process, as it allows verifiers to verify the received proofs, i.e., verifiable presentations, without the involvement of credential issuers. It includes patterns that address various aspects of building trust, namely, (1.1) *Public Institution Registry*, (1.2) *Trusted Schemas Registry*, (1.3) *Status Registry*, (1.4) *DID Registry or Identifier Registry*, (1.5) *Identifier-Attribute Registry*, (1.6) *Blockchain Anchor*.

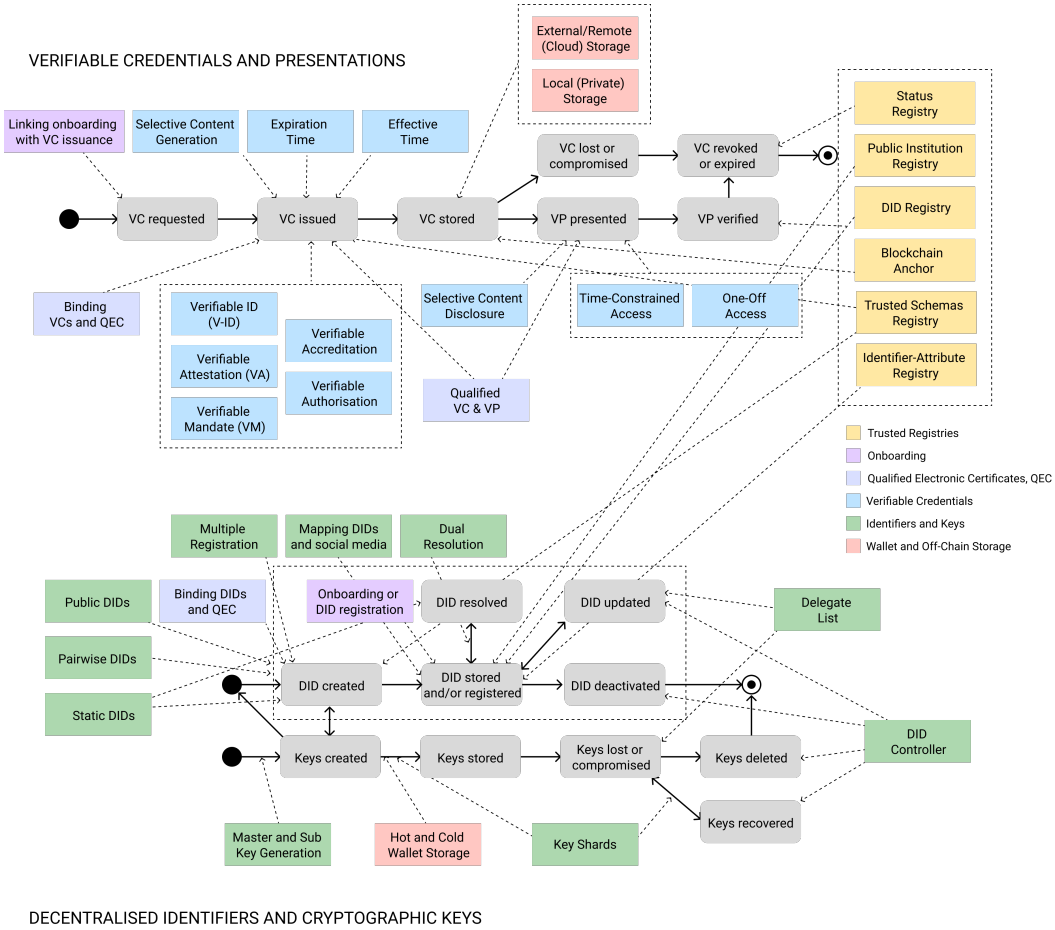


Figure 4. Life-cycle of three main SSI objects (VCs, DIDs, and keys) and patterns connected to their states or transitions, built on the work of Liu et al. [2,14].

(1.1) The *Public Institution Registry* maintains a publicly accessible list of (public) institutions/organisations/businesses and their information. It allows verification of their identity, accreditations, and authorisations. It enables verifiers to verify the credential issuer and its publicly available data, to ensure that the issuer has the required accreditations for the issuance of a particular type of credential at the time of issuance. This is crucial, especially when dealing with unknown issuers, as verifiers have to trust issuers to be able to trust the credibility of the data. (1.2) The *Trusted Schemas Registry* pattern is paramount for ensuring semantic interoperability between various systems, alignment with domain-specific requirements, and providing quality data objects. It provides a publicly available registry containing data schemas of data objects, and facilitates digital interactions and data exchange between different systems. We refer to data objects as any type of VCs or DIDs, e.g., a verifiable attestation as a subtype of VC, which attest that an individual has a formal higher education (HE) diploma, whereby this diploma is defined with a VC through the official schema for HE diplomas. The schema is, thus, based on the official template for defining formal diplomas, whereby there could be several different HE diploma schemas, depending on the regulation, legislation, country, membership, etc. To establish trust, entities must be able to verify that data objects are structured correctly and approved by relevant authorities. Moreover, without predefined schemas, an unregulated explosion of data objects may occur. (1.3) The *Status Registry* contains a status list with information on credential's validity. Most credentials require the possibility of revocation, suspension, or other means for terminating the validity before the credential's expiration date for various reasons. Identity and other identity attributes are a dynamic concept, so some of the information in the issued credentials may change; there may be errors in the issuance of the credential, devices, or private key(s) may be lost, or the credential may be misused. Thus,

issuers must be able to revoke or suspend credentials, while verifiers must have the means to check the validity of presented credentials without contacting the issuers. (1.4) The *DID Registry or Identifier Registry* pattern serves as a decentralised public-key infrastructure (DPKI), and provides a binding between the entity's identifier(s) and associated public key(s), service endpoint, and other metadata. It is essential for providing a means for discovery, identification, and authentication, and provides trust and integrity. To ensure compliance with the GDPR and prevent disclosure of identity, the Registry should not contain any personally identifiable information of a natural person, since, in the majority of cases, decentralised storage or ledgers will be used for the registries. (1.5) The *Identifier-Attribute Registry* maintains mappings between an identifier and the correlated data or its location (off-chain storage URI). The identity data of a natural person should never be stored on-chain due to privacy and security reasons. On the other hand, there may be limitations to the storage capacity of blockchain. Thus, data can be stored off-chain on decentralised file systems such as IPFS, Ceramic Network, etc., and referenced on-chain [14]. (1.6) The *Blockchain Anchor* pattern can be used to preserve the integrity of off-chain stored data and maintain the entity's privacy. Instead of storing everything on-chain, only hashed values of off-chain data can be anchored on the blockchain. The pattern solves problems related to blockchain storage capacity limitations and GDPR implications, as personally identifiable information should never be stored on-chain and takes into account transaction fees [14].

Table 1. SSI design patterns, classified into the Trusted Registries category.

Design Pattern	Summary
Public Institution Registry	A register containing information about public entities such as companies, organisations, and institutions, their accreditations, authorisations, etc. [31].
Trusted Schemas Registry	A registry that contains data schemas (templates) of data objects (DIDs, VCs, semantic context) and allows their reuse [32].
Status Registry	The registry contains information required for the verification of (i) verifiable credentials, or (ii) verifiable presentation validity [33].
DID Registry	A registry for the storage of public DIDs that provides a binding between an entity's identifier(s), public key(s), and other metadata such as service endpoints, etc. Thus, it serves as a decentralised public-key infrastructure (DPKI) [33].
Identifier-Attribute Registry	A registry that maintains a mapping between an identifier and the location, i.e., the address of identity attributes for off-chain storage (e.g., a name and profile picture of an entity) [2,14].
Blockchain Anchor	A registry for storing evidence in the form of hashed data values. Instead of recording all the data on-chain, only a hashed value of the data is recorded on-chain, which can be used for verifying the authenticity of data stored outside the blockchain [2,14].

The *Onboarding* category (Table 2) includes two patterns, (2.1) *Onboarding or DID Registration*, and (2.2) *Linking onboarding with credential issuance*, to interact with other entities within decentralised identity ecosystem and identity management frameworks such as those of EBSI, which is also utilising its own permissioned ledger, where a public entity must first (2.1) *onboard/register* through an onboarding service to obtain writing and updating permissions, and store at least one DID and correlated public key on the DID Registry. However, *onboarding* should be linked with VC issuance, as it requires identification, and regulations demand sufficient justification when requesting and processing personal data. The mere reason for identification is not sufficient. On the other hand, *linking VC issuance with the DID registration* process provides an adequate justification. Moreover, entities usually do not have a profound interest in merely registering DIDs, but have an interest in collecting their data (VCs), building a digital identity and acquiring access to the services. Therefore, isolated DID registration is unreasonable.

Table 2. SSI design patterns, classified into the Onboarding category.

Design Pattern	Summary
Onboarding or DID Registration	Provision of an entity onboarding service to register identifiers in the DID registry (required when using private and consortium blockchains) [34].
Linking onboarding with VC issuance	The onboarding or registration of DIDs should be linked to verifiable credential issuance. Onboarding requires identity verification, which must be substantiated [35].

Category *Verifiable Credentials and Presentations* (Table 3) includes patterns that support data exchange through the VC data format. Patterns (3.1) *Verifiable ID (V-ID)*, (3.2) *Verifiable Attestation (VA)*, (3.3) *Verifiable Mandate (VM)*, (3.4) *Verifiable Accreditation (VAcc)*, and (3.5) *Verifiable Authorisation (VAuth)* support different scenarios and use cases. Although they are all technically the same VC generic data format they provide an important distinction, as different scenarios have various purposes and (functional/legal) requirements regarding the level of assurance (LoA), which has implications for the credential issuance process and the attributes contained, etc. Each of the data formats thus has its complex issuance and verification requirements in technical, as well as organisational/management, terms.

(3.1) A *V-ID* should be used for identification and authentication purposes, e.g., a natural person receives a signed VC of type V-ID from his/her administrative unit, which acts as a digital ID card, holding the minimally required information for an ID card, defined through a specific schema ((1.2) *Trusted Schemas Registry*). It should be noted that, due to various regulations and legislations, different administrative units can have different templates, and, thus, schemas for the same generalised ID card. Additionally, V-IDs are not only bound to formal documents, but can also be applied to less formal or informal ones, e.g., student cards, library cards, etc. (3.2) A *VA* should be used for proving certain attributes/properties and permits/attestation/authorisation to access services. Examples of such attestations are, again, more formal documents such as diplomas or driving licences, proving the holder has finished some form of education or has gained some knowledge and thus has gained some rights, such as applying to a specific job posting or driving a car. However, the attestations can also be less formal or informal, such as proof of attending an event, proof of achieving first place at a sports competition, etc. As with the V-ID, each of the aforementioned scenarios or use cases for VA could have its own schema defined (i.e., not mandatory). (3.3) A *VM* should be used for transferring authority and/or access from one entity to another. An example of such a mandate is indirect identity controls, such as those of delegation, as well as guardianship and/or controllership. Each of these use cases has its own complexity and rules. The former are classical power of attorney rights, while the latter are use cases where parents or custodians are acting in the name of their children, and entities such as IoT devices are acting in the name of their owners/controllers. (3.4) *VAcc* should be used for proving needed accreditations that ensure compliance with the standards and requirements, and (3.5). An example of such is formal accreditation for higher education institutions received from Ministries, which gives them the right to issue formal diplomas. Lastly, (3.5) *VAuth* should be used as authorisation proof for accessing permissioned ledgers. These VC data formats play the role of an access token for accessing services, or, in this case, a permissioned ledger. An example of such is seen in the EBSI onboarding processes.

The remaining patterns within this category are solving problems related to flexibility, privacy, and data minimisation. (3.6) *Selective Content Generation* and (3.7) *Selective Content Disclosure* support the creation of customised credentials and presentations, allowing data minimisation by generating (3.6) VCs according to the holders' specific requirements, or (3.7) VPs according to the verifiers' requirements. Both enable the holder to disclose the minimum amount of data necessary for a particular interaction by utilising mechanisms such as atomic credentials, selective disclosure signatures, hashed values and zero-knowledge proof (ZKP) [14]. By utilising selective content disclosure, identity holders can create VPs consisting of only a subset of attributes from one or more credentials that are required for a

particular interaction. Moreover, they can provide attributes without actually revealing their value. For example, if an identity holder wants to prove his/her age, he/she can use selective disclosure to share only the year of birth from the driver's licence, while not revealing the rest of the attributes, such as day and month of birth, address, etc. Additionally, by using ZKP, he/she can prove that, for example, he/she is over 18 years old without revealing the actual date of birth. This is especially useful in situations where identity holders do not trust the verifiers. (3.8) *Time-Constrained Access* and (3.9) *One-Off Access* patterns provide privacy and flexibility, and allow identity holders to restrict access to shared credentials to (3.8) specific time frames, or (3.9) limit it to one-time access only. This is important, as the holder's identity data should not be accessed or verified without a legitimate reason after completing the initial identification process [14]. (3.10) *Expiration Time* enhances trust, increases data accuracy, and reduces misuse and/or abuse of credentials. It enables issuers to restrict the effectivity of credentials, as identity is a dynamic concept, meaning the holder's identity attributes and conditions may change. Therefore, credentials should not be long-term or effective permanently. On the other hand, (3.11) *Effective Time* allows differentiation between the time of the issuance and the time when the credential becomes valid.

Table 3. SSI design patterns, classified into the Verifiable Credentials and Presentations category.

Design Pattern	Summary
Verifiable ID	Use verifiable ID (VI-D) as evidence/proof of an entity's identity for identification and authentication purposes [36–38].
Verifiable Attestation	Use verifiable attestation (VA) as evidence/proof of specific identity attributes or compliance with certain conditions for nonidentification and non-verification-related scenarios [8,39–42].
Verifiable Mandate	Use a verifiable mandate (VM) as evidence/proof that an entity (delegate) can act on behalf of another entity (delegator) [43–45].
Verifiable Accreditation	Use verifiable accreditation (VAcc) as evidence/proof to demonstrate the appropriate/required accreditations of legal entities to perform particular tasks, such as issuance of certain types of credentials [46–51].
Verifiable Authorisation	Use verifiable authorisation (VAuth) to authorise entities to interact with permissioned ledgers and access resources/services [52].
Selective Content Generation	Creating a customised credential according to the identity holder's specific requirements for credential attributes/contents (interaction between the issuer and the identity holder) [2,14].
Selective Content Disclosure	Creating a customised presentation according to the verifier-specific requirements for credential attributes/contents (interaction between the identity holder and the verifier) [2,14,53].
Time-Constrained Access	Generate and forward a time-limited credential link to allow the verifier access only within the specified time window [2,14].
One-Off Access	Generate and forward a time-limited credential link to allow the verifier to access it only once [2,14].
Expiration Time	Include information that restricts the time of the credential validity/usage [8,54].
Effective Time	Differentiate between the issuance date and the date when the credential becomes valid [39,54].

The category *Decentralised Identifiers and Cryptographic Key* (Table 4) includes patterns (4.1) *Multiple Registration*, (4.2) *Blockchain and Social Media Account Pair*, (4.3) *Public DIDs or Anywise DIDs*, (4.4) *Pairwise DIDs (and N-wise DIDs)*, (4.5) *Static DIDs*, (4.6) *Dual Resolution or DID Resolution*, (4.7) *Delegate List*, (4.8) *DID Controller*, (4.9) *Master and Sub-Key Generation*,

and (4.10) *Key Shards* patterns, dealing mainly with solving problems connected to identification, authentication, and management of DIDs, DID documents, and cryptographic keys.

Table 4. SSI design patterns, classified into the Decentralised Identifiers and Cryptographic Keys category.

Design Pattern	Summary
Multiple Registration	Each entity can create (and register) an unlimited number of unique identifiers (DIDs) and corresponding cryptographic keys for each relationship (i.e., every identity) it has [2,14].
Blockchain and Social Media Account Pair	Establishing a bidirectional link between a social media profile and a blockchain-based identity can increase trust in social networks and the identifier stored on the blockchain [2,14].
Public DIDs or Anywise DIDs	Utilise anywise, public DIDs that require the use of a verifiable data registry (for the execution of CRUD operations) when identifiers need to be available and resolvable by anyone/strangers publicly [7,16,55,56].
Pairwise DIDs (and N-wise DIDs)	Utilise pairwise (and N-wise) private DIDs in personal, confidential interactions between a small number of entities. Such DIDs do not require anchoring to a verifiable data registry, as they are exchanged peer-to-peer [7,57].
Static DIDs	Utilise static DIDs for single, ephemeral interactions that do not require updating and deactivating DID documents [55–58].
Dual Resolution or DID Resolution	Entities engaged in a mutual interaction have to acquire each other's DID and resolve it to DID documents to access the information necessary for authentication (e.g., public key) and communication (e.g., service endpoints for provided services) [2,14].
Delegate List	Each entity can maintain a list of delegates or proxies in the DID document who can update a DID document and help recover the identifier in case of lost or compromised keys [2,14].
DID Controller	Entities can designate one or multiple DID controllers (in the DID document) who can control the DID and modify the DID document [7].
Master and Sub Key Generation	Each entity has a master key for managing a set of sub-keys used to sign transactions for different purposes or identity accounts (verification relationships) [2,14].
Key Shards	Entities can split cryptographic keys into several different pieces and recover them with a sufficient number of parts [2,14].

(4.1) The *Multiple Registrations* pattern enables entities to create and register multiple unique identifiers and correlated cryptographic keys, separate for every relationship they have. Each identifier can be used in different interactions (relationships) to preserve privacy, minimise the possibility of correlation and increase security [14]. (4.2) The *Blockchain and Social Media Account Pair* provides a solution for establishing a bidirectional binding between social media accounts and a decentralised identity, which can enhance the trustworthiness of social media accounts and allows for building a digital reputation [14]. (4.3) *Public DIDs or Anywise DIDs*, (4.4) *Pairwise DIDs (and N-wise DIDs)*, and (4.5) *Static DIDs* allow entities to identify and authenticate. Each pattern supports different types of relationships and their requirements, as some identifiers and interactions should remain private, while others require public availability. Therefore, (4.3) *Public DIDs* (e.g., did:ebsi for legal entities, did:ethr, did:sov) should be utilised when DIDs need to be available publicly and resolvable by anyone. (4.4) *Pairwise DIDs* (e.g., did:peer) should be used for peer-to-peer, private interactions between a small number of entities, and (4.5) *Static DIDs* (e.g., did:key) should be used for single, ephemeral interactions that do not require updating and deactivating DID documents, (4.3) require anchoring DIDs on the VDR, a shared registry that provides a public source of truth and allows arbitrary entities to resolve DIDs to an endpoint and keys. On the other hand, (4.4) and (4.5) are entirely off-chain. Regardless of the type of identifier,

entities have to identify and verify each other before engaging in mutual interaction and exchanging information. (4.6) *Dual Resolution or DID Resolution* allows them to obtain DIDs and resolve them to DID documents to access the information necessary for authentication (e.g., a public key) and communication purposes (e.g., service endpoints). Each DID is managed by a private key, which is used to prove ownership of the identifier. If the key is lost, stolen, or compromised, patterns (4.7) *Delegate List* and (4.10) *Key Shards* can be used to retrieve ownership. Both patterns provide approaches for solving problems related to identity recovery, which is crucial for maintaining ownership and control over identifiers, related identity data, and assets in decentralised settings. The *Delegate List* allows the identity holder to determine a list of delegates who can help to recover his/her identity if private keys are lost or compromised. The identifier and their keys can be replaced and updated in a DID document after a sufficient number of trustees/delegates vouch for the DID owner. On the other hand, keys can be secured and split into multiple pieces/shards, stored on multiple devices or by multiple trusted entities (i.e., social recovery), and restored by using enough pieces, utilising the *Key Shards* pattern [14]. (4.8) The *DID Controller* is related to DID document management, and allows differentiation between a DID subject and the DID controller(s), as, sometimes, the DID subject is not the same as the DID controller (e.g., parents control the identity of their underage children; pet owners control the identifier of their pets, etc.), and some use cases require multiple entities to control a DID (e.g., multiple owners control the organisation identifier) which can be determined in the DID document. Thus, the controller(s) defined in the documents are allowed to make changes to DID document(s), supporting more complex use cases where DID subjects are not capable of controlling their identity (e.g., guardianship as a type of indirect identity control). (4.9) The *Master and Sub-Key Generation* pattern can be used to minimise correlation and increase the privacy and security of entities' identity and their keys. Thus, each entity has multiple keys connected with a particular identity, used for different purposes, meaning that each entity has a master key for managing a set of sub-keys used for signing different types of transactions [14].

Patterns within the *Qualified Electronic Certificates (QEC)* category (Table 5) can provide legal value, as they bind SSI objects (VCs and DIDs) with the QEC issued by a duly accredited trust service provider. The binding can, therefore, provide the highest legal guarantee, instil trust in the entity's identity, and assure the integrity of signed documents and their enforceability. This is crucial, because some transactions require a higher LoA to establish sufficient trust. Therefore, some credentials and presentations require a regulatory-supported legal value.

Table 5. SSI design patterns, classified into the Qualified Electronic Certificates (QEC) category.

Design Pattern	Summary
Qualified Verifiable Credentials	Signing verifiable credentials (and presentations) with qualified digital signatures or seals (eSignature, eSeal) and adding legal value to credentials [59,60].
Binding VCs and QEC	Linking verifiable credentials with qualified electronic certificates [59,60].
Binding DIDs and QEC	Linking decentralised identifiers with qualified electronic certificates, either by enriching the DID document or using trusted registries [61].

VCs can be signed by DIDs' corresponding private keys. However, currently, there is no binding between the DID and real-world natural or legal persons, and no law confers binding between legal value to credentials. Thus, (5.1) *Qualified Verifiable Credentials* enables entities to sign/seal VCs and VPs with QEC private keys that have a legally binding value/effect. Decentralised digital identity can be linked with a real-world identity by establishing binding with a QEC, either through VCs or DIDs. (5.2) *Binding VCs and QECs* require issuing verifiable credentials that demand identification and authentication before

VC issuance, and they contain information about an entity and can provide substantial proof and evidence. (5.3) *Binding DIDs and QECs* allows entities, i.e., verifiers, to retrieve qualified certificates and verify them against a trusted list. Thus, verifiers can check reliably which entity is behind a particular DID. DIDs can be linked with QECs using different approaches, either by enriching DID documents or by utilising trusted registries. However, it should be noted that trusted registries should not link natural person identifiers with public certificates, to avoid revealing their identity and comply with GDPR.

Last but not least, the *Wallet and Off-Chain Storage* category encloses (6.1) *Hot and Cold Wallet Storage*, (6.2) *Local (Private) Storage*, and (6.3) *External (Remote) Cloud Storage* patterns.

Identity wallets provide a means for digital identification, authentication, and authorisation. They serve as a tool for creating and managing identifiers and associated cryptographic keys, as well as for obtaining, storing, and distributing credentials and other personal data, which have to be stored securely and under the complete control of the identity holder. Patterns (6.2) and (6.3) provide storage for identifiers, keys, credentials and other personal data completely off-chain, either locally on the identity holder's device (smartphone, tablet, computer, etc.), or in remote locations in the cloud where they can be accessible to their holders or wallet controllers. Thereby, trusted cloud-based data storage can be used to avoid the availability and accessibility challenges of local storage.

Table 6. SSI design patterns, classified into the Wallet and Off-Chain Storage category.

Design Pattern	Summary
Hot and Cold Wallet Storage	Each entity can store keys in two wallets. It stores frequently used keys in a hot wallet with an internet connection and rarely used keys in a cold wallet without an internet connection [2,14].
Local (Private) Storage	Local (off-chain) storage of identifiers, keys, verifiable credentials, and other personal data that can be accessed only by an identity holder or wallet controller [62,63].
External/Remote (Cloud) Storage	External (off-chain) cloud storage for storing identifiers, keys, verifiable credentials, and other personal data [62].

(6.1) *Hot and cold wallet storage* provides an additional security layer, and can be used for recovery purposes. It allows each entity to maintain keys in two wallets, i.e., hot and cold wallets, providing online storage for frequently used keys and offline storage for infrequently used keys, respectively [14].

3.4. Example of a Verifiable Mandate Pattern

As mentioned in the previous section, the result of the research is a documentation of 35 design patterns defined using the Alexander template, whereby an example of (3.3) the *Verifiable Mandate* pattern is given below. The pattern addresses indirect identity control. Precisely, it provides a solution for delegation use cases where there are not just one, but multiple identity holders that delegate tasks/permissions from one to another, as presented in Figure 5. The remaining patterns have been defined in the same way, and are available at [30].

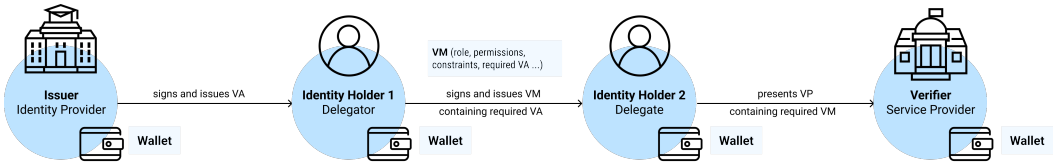


Figure 5. The Verifiable Mandate pattern enables delegating tasks and transferring permissions from one identity holder to another.

Name: Verifiable Mandate (VM)

Summary: Use a verifiable mandate (VM) as evidence/proof that an entity (delegate) can act on behalf of another entity (delegator).

Problem
1. Verifiable credentials represent credentials with or without a physical equivalent, and can contain various information supporting various use cases. Hence, different scenarios/use cases have distinct purposes and (functional/legal) requirements regarding the level of assurance, attributes contained, etc.
2. There are cases where an entity (delegate) has to act on behalf of another entity (delegator) in order to achieve a particular goal. Therefore, the delegate must be able to prove that he/she has the required permissions, and can act in a particular situation on behalf of the delegator. Such situations require consideration of permissions and constraints that must be integrated into credentials.
3. Credentials must address constraints to prevent overuse and abuse of delegation by a delegate.

Context
In some cases, authority and/or access need to be transferred from one entity to another, which is especially evident in the work environment with superior–subordinate relationships. Superiors usually delegate tasks to subordinates, who are responsible for carrying out the tasks on their behalf.

Forces
1. Flexibility: Identity holders (delegators) can transfer part of their tasks to other entities (delegates). After obtaining the VM, delegates can act on behalf of the delegator, which allows flexibility and supports various scenarios.
2. Control: Holders can obtain greater control in digital interactions.
3. Security: There are some risks associated with determining constraints to prevent the overuse and abuse of delegated rights.
4. Privacy: The VM should contain the minimum amount of data necessary for delegates to carry out tasks on behalf of delegators.

Solutions
A special type of VC, a verifiable mandate (VM), can be used for the delegation of authority and access. The VM should contain claims that may serve as evidence of permission/authorisation transfer from one entity to another. Therefore, such a credential allows an entity (VM holder) to act on behalf of another entity (the VM issuer). The VM should be built on existing standards, namely, the W3C Verifiable Credentials Data Model and date time format. Therefore, the VM data model is a VC, with additional properties that depend on the use case and are contained within the *credentialSubject* object. However, they must include a *holder* property that describes a subject called holder (delegate) and includes (i) a *role* property that defines the role(s) the holder plays in the permission scheme of the delegator;
(ii) *permissions*, i.e., actions or grants that can be performed by the VM holder; (iii) *constraints* that limit the scope of the delegated authority. Constraints may include properties such as *startTime, endTime, location, boundaries, circumstances, pointOfOrigin, and radiusKm* [64]; (iv) *PolicySchemaURI* with reference to the policy validation script that can be registered on the verifiable data registry (in trusted policy registry) and contains available roles, permissions, and constraints. Moreover, it should be accompanied by the credentials required for carrying out the delegated task(s), either in the VM itself or separately.

Example
Listing 1 presents an example of an issued credential (JSON) utilising the verifiable mandate template that was issued by delegator (VM issuer) *did:ebsi:ztZgP4NGDZgo GRqHpaxQkfX* to delegate (VM holder), *did:key:z6MkvUtihkikPAdwzs18AExS4GFr6t4owy HFRrLFdpRScZDd* [43]. By obtaining the aforementioned VM, the delegate is entitled to enrol a delegate to a Master's degree in Slovenia, whereby the application has the same degree of validity as if the delegate had performed the process by himself/herself.

Resulting Context
1. Indirect identity control use cases (i.e., delegation processes) can be supported.
2. Entities (delegators) can delegate (part of) their work/responsibilities to other entities (delegates), who can act on the delegator's behalf with the same level of validity while remaining in full control of their identity (identifiers and keys).
3. An entity (delegator) who delegates (part of) work/responsibilities to another entity (delegate) is in charge of defining the roles, permissions, and constraints that limit the delegate from exceeding his/her authority. Therefore, the VM provides flexibility and even more control to identity holders, as it supports complex use-cases that cannot be supported by utilising a general VC.
4. The VM can be used as proof that an entity (VM holder) can act on behalf of another entity (VM issuer). Thus, during the identification/authentication phase, an entity can also present a VM showing that he/she can act on behalf of another entity within the corresponding limitations.

Related Patterns
1. The Trusted Schema Registry: VM schema can be defined, stored and maintained in the TSR by entities (verifiers) who support delegation scenarios.
3. Blockchain Anchor: VM issuance can be anchored on the BC (as proof) to preserve the integrity of the data.
4. Status Registry: The VM can contain a reference to a status registry, allowing delegators to suspend or revoke the validity of VMs.
5. Verifiable ID and Verifiable Attestation: The VM is complementary to V-ID and VA, and is usually presented together with a V-ID and/or VA.
6. Patterns connected to the Verifiable Credentials category: A VM pattern can be utilised together with Time-Constrained Access, One-Off Access, Expiration Time and Validity Time patterns that enhance the flexibility of the pattern.

Known use
The Hyperledger Aries RFC 0103 proposed a technical solution, including proxy credentials for supporting various indirect control use cases, such as delegation, guardianship, and controllership [64–66]. The proposed solution is currently under investigation for implementation into various pilots and proofs of concept, e.g., [67]. The EBSI ESSIF specified three distinct types of VCs aligned with the W3C VC Standard [45,68], including the VM, which was also addressed in the project collaboration by Netis, Walt.id, and the Faculty of Electrical Engineering and Computer Science of the University of Maribor [43].

Listing 1. Verifiable Mandate example [43].

```
{
  "@context" : [ "https://www.w3.org/2018/credentials/v1"],
  "credentialSchema" : {
    "id" : "https://api.preprod.ebsi.eu/trusted-schemas-registry/v1/schemas/
    0xb77f8516a965631b4f197ad54c65a9e2f9936ebfb76bae4906d33744dbcc60ba",
    "type" : "FullJsonSchemaValidator2021"
  },
  "credentialSubject" : {
    "holder" : {
      "constraints" : {
        "location" : "Slovenia"
      },
      "grant" : "apply_to_masters",
      "id" : "did:key:z6MkvUtihkikPAdwzs18AExS4GFr6t4owyHFRrLFdpRScZDd",
      "role" : "family"
    },
    "id" : "did:ebsi:ztZgP4NGDZgoGRqHpaxQkfX",
    "policySchemaURI" : "https://raw.githubusercontent.com/walt-id/waltid-ssikit/
    master/src/test/resources/verifiable-mandates/test-policy.rego"
  },
  "evidence" : [ {
    "evidenceValue" : "",
    "id" : "https://essif.europa.eu/tsr-va/evidence/
    f2aeec97-fc0d-42bf-8ca7-0548192d5678",
    "type" : [ "VerifiableMandatePresentation" ]
  } ],
  "id" : "urn:uuid:40733519-78c7-4248-a80a-fede7c5ce978",
  "issued" : "2022-05-19T21:39:21.880729594Z",
  "issuer" : "did:ebsi:ztZgP4NGDZgoGRqHpaxQkfX",
  "validFrom" : "2022-05-19T21:39:21.880731854Z",
  "issuanceDate" : "2022-05-19T21:39:21.880731854Z",
  "type" : [ "VerifiableCredential", "VerifiableMandate" ],
  "proof" : {
    "type" : "EcdsaSecp256k1Signature2019",
    "creator" : "did:ebsi:ztZgP4NGDZgoGRqHpaxQkfX",
    "created" : "2022-05-19T21:39:53Z",
    "domain" : "https://api.preprod.ebsi.eu",
    "nonce" : "0f083aa8-4e8e-4bb3-acef-5467fbc077a3",
    "proofPurpose" : "assertion",
    "jws" : "eyJiNjQiOmZhbHNlLCJjcml0IjpbImI2NCJdLCJhbGciOiJFUzI1NksifQ..OPHTDcz6AWS
    F8SODePIRM9xCCvmOqbOuUk88E6piCALTOQticpiHnfnOZiYZRAvXmXEJ1iDjI6tVrNAq2kKTug"
  }
}
```

4. Discussion

This study resulted in an SSI design patterns' collection, grouping 35 patterns into six representative categories that enable a better understanding of patterns and their relation to each other. The patterns were grouped into (1) *Trusted Registries*, (2) *Onboarding*, (3) *Verifiable Credentials and Presentations*, (4) *Decentralised Identifiers and Cryptographic Keys*, (5) *Qualified Electronic Certificates*, and (6) *Wallet and Off-Chain Storage* categories, as presented below.

1. *Trusted Registries*
    - 1.1 Public Institution Registry
    - 1.2 Trusted Schemas Registry
    - 1.3 Status Registry
    - 1.4 DID Registry or Identifier Registry
    - 1.5 Identifier-Attribute Registry
    - 1.6 Blockchain Anchor
2. *Onboarding*
    - 2.1 Onboarding or DID Registration
    - 2.2 Linking onboarding with VC issuance

3. *Verifiable Credentials and Presentations*
    - 3.1 Verifiable ID
    - 3.2 Verifiable Attestation
    - 3.3 Verifiable Mandate
    - 3.4 Verifiable Accreditation
    - 3.5 Verifiable Authorisation
    - 3.6 Selective Content Generation
    - 3.7 Selective Content Disclosure
    - 3.8 Time-Constrained Access
    - 3.9 One-Off Access
    - 3.10 Expiration Time
    - 3.11 Effective Time
4. *Decentralised Identifiers and Cryptographic Keys*
    - 4.1 Multiple Registration
    - 4.2 Blockchain and Social Media Account Pair
    - 4.3 Public DIDs or Anywise DIDs
    - 4.4 Pairwise DIDs (and N-wise DIDs)
    - 4.5 Static DIDs
    - 4.6 Dual Resolution
    - 4.7 Delegate List
    - 4.8 DID Controller
    - 4.9 Master and Sub-Key Generation
    - 4.10 Key Shards
5. *Qualified Electronic Certificates*
    - 5.1 Qualified Verifiable Credentials
    - 5.2 Binding VCs and QEC
    - 5.3 Binding DIDs and QEC
6. *Wallet and Off-Chain Storage*
    - 6.1 Hot and Cold Wallet Storage
    - 6.2 Local (Private) Storage
    - 6.3 External/Remote (Cloud) Storage

Patterns are related to the state transitions in lifecycles of the main SSI objects, namely, decentralised identifiers (DIDs), cryptographic keys, and verifiable credentials (VCs). In addition, they are also interconnected with patterns from the same or distinct categories. Figures 6 and 7 show the relationships between patterns, where patterns from the same category are represented by the same colour, and the grey lines illustrate their interconnections. While Figure 6 depicts the broader angle of pattern connection, Figure 7 focuses on patterns from the Verifiable Credential category.

Each pattern is related in some way to patterns from the same category, whereby a pattern may either be (i) equivalent, or used (ii) in combination with another pattern within the category. For example, (i) patterns (3.1) *Verifiable ID (V-ID)*, (3.2) *Verifiable Attestation (VA)*, (3.3) *Verifiable Mandate (VM)*, (3.4) *Verifiable Accreditation (VAcc)*, and (3.5) *Verifiable Authorisation (VAuth)* are equivalents that can be used independently of one another, each supporting a different use case with distinct issuance and verification requirements in technical and organisational terms. (ii) On the other hand, the remaining patterns in the category ((3.6) *Selective Content Generation*, (3.7) *Selective Content Disclosure*, (3.8) *Time-Constrained Access*, (3.9) *One-Off Access*, (3.10) *Expiration Time* and (3.11) *Effective Time*) can provide flexibility and control over identity data, and can be seen as an extension of patterns 3.1–3.5. Thus, they can be used in combination with different types of VCs. Alternatively, patterns are also related to patterns in other categories. For example, VC schemas are stored in the (1.3) *Trusted Schemas Registry*, whereby the VC contains a reference to the registry. Similarly, the VC can reference the (1.3) *Status Registry*, containing information

about the VC's validity. It can serve as a trust anchor that allows verifiers to check whether the respective VC is still valid, or has been revoked or suspended. Moreover, a hashed VC, or evidence of a VC's issuance/reception/sharing, can be anchored on the VDR as proof/evidence (1.6 *Blockchain Anchor*). Furthermore, to enlist the legal value of a VC, (i) the VC can be signed with a qualified electronic signature (QEC) instead of cryptographic keys correlated to the entities' DID (5.1 *Qualified Verifiable Credentials*), or (ii) the VC can be linked with QEC (5.2 *Binding VCs and QEC*).

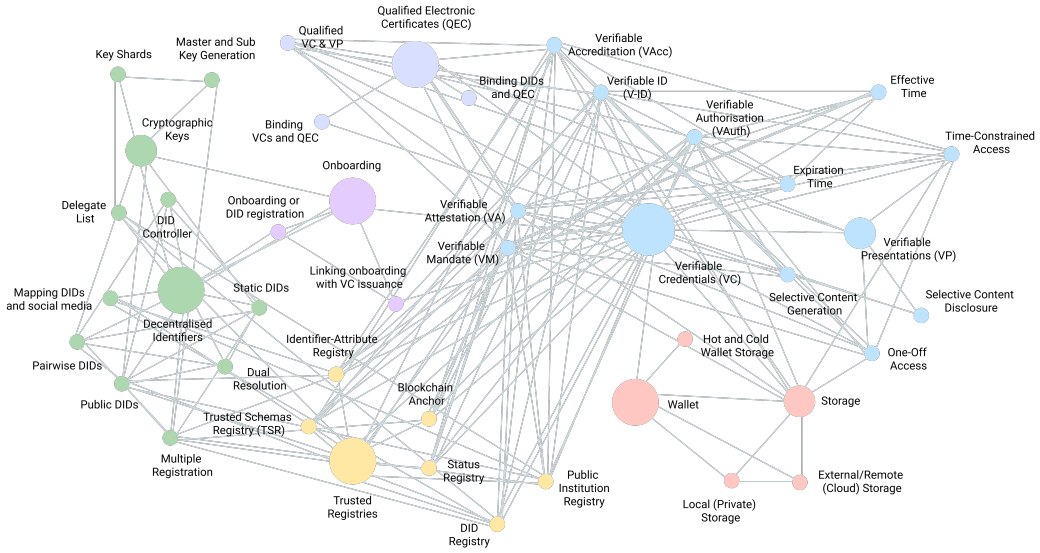


Figure 6. Interconnection of SSI design patterns from different categories.

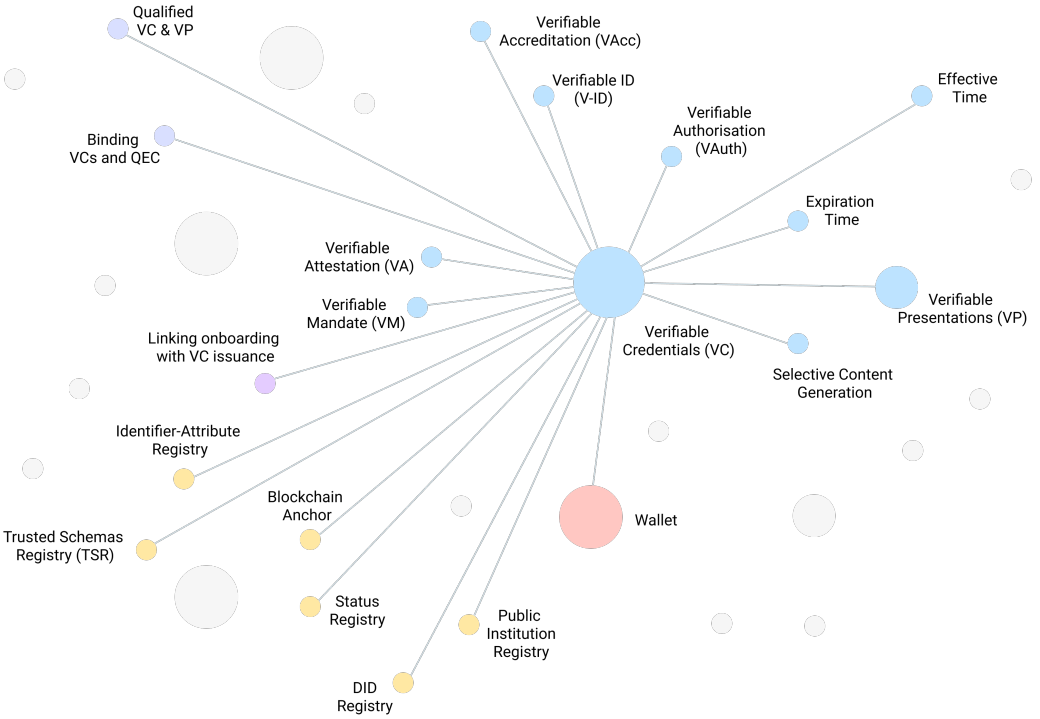


Figure 7. Highlighting relationships between VCs and other patterns.

5. Conclusions

There are many benefits of design patterns, as they present a means for solving common design problems. They provide tested and proven paradigms, and can be reused in new implementations as the best possible solution for specific problems. They capture good practices and lessons learned, and can speed up the development process drastically.

They can increase the quality, readability, and system documentation while helping to understand the software and the rationale behind solutions.

With the evolvement of the SSI field, patterns can be recognised from proof-of-concept, implemented solutions, and technical documentation. Thus, identifying and gathering the patterns can be beneficial for guiding engineers in designing and developing SSI solutions. Therefore, this study investigated the literature and technical documentation of existing implementations, with the aim to provide a classified catalogue of self-sovereign identity design patterns that can serve as a starting point for architects and engineers responsible for designing robust and well-designed SSI systems and applications.

Building on existing knowledge and based on studying the EBSI ESSIF, Veramo, and Sovrin documentation, we collected 35 design patterns, whereby we defined twenty-three of them based on the Alexander template, while the remaining twelve were already defined by Lie et al. Patterns are summarised throughout this paper (Section 2.2), and are classified into representative categories.

The survey could be improved by: (i) Analysis of additional technical documentation to complement/extend the known use cases. This would allow us potentially to identify additional patterns (which were overlooked in this iteration). (ii) Connecting patterns and validated SSI properties or identifying patterns that enable the SSI properties to be met. SSI properties can be used to validate and determine whether implemented solutions are truly SSI, or merely decentralised identities. (iii) Validation of newly defined design patterns by experts (researchers and developers of SSI solutions). To improve the quality of patterns we could imply, i.e., a shepherding method, which was introduced by Harrison [17]. The method is used in the pattern community to improve pattern quality by having an experienced pattern writer review the patterns. (iv) It would also be useful to implement a tool to provide decision support and the selection of appropriate patterns to address specific problems in the design and development of SSI solutions.

Author Contributions: Conceptualisation, Š.Č. and M.T.; data curation, Š.Č.; formal analysis, Š.Č.; investigation, Š.Č. and V.K.; methodology, Š.Č. and M.T.; project administration, M.T.; supervision, M.T.; validation, Š.Č., V.K. and M.T.; visualisation, Š.Č.; writing—original draft, Š.Č. and M.T.; writing—review and editing, Š.Č., V.K. and M.T. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Slovenian Research Agency (Research Core Funding) under Grant P2-00577, and by the European Union's Horizon 2020 Research and Innovation Program under Grant Agreement No. 870635 (Digital Europe for All—DE4A).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Pattern definitions and connection visualisations are available at [30].

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

DID	Decentralised Identifier
DIF	Decentralised Identity Foundation
DLT	Distributed Ledger Technology
DPKI	Decentralised Public-Key Infrastructure
EBSI	European Blockchain Services Infrastructure
ESSIF	European Self Sovereign Identity Framework
HE	Higher Education
IdP	Identity Provider
IoT	Internet of Things

JSON	JavaScript Object Notation
LoA	Level of Assurance
OOP	Object-Oriented Programming
PoC	Proofs-of-Concept
QEC	Qualified Electronic Signature
RQ	Research Question
SMS	Systematic Mapping Study
SP	Service Provider
SSI	Self-Sovereign Identity
ToIP	Trust Over IP
URI	Uniform Resource Identifier
URN	Uniform Resource Name
V-ID	Verifiable ID
VA	Verifiable Attestation
VAcc	Verifiable Accreditation
VAuth	Verifiable Authorisation
VDR	Verifiable Data Registry
VC	Verifiable Credential
VM	Verifiable Mandate
VP	Verifiable Presentation
W3C	World Wide Web Consortium
ZKP	Zero-Knowledge Proof

References

1. Čučko, Š.; Turkanović, M. Decentralized and Self-Sovereign Identity: Systematic Mapping Study. *IEEE Access* **2021**, *9*, 139009–139027. [CrossRef]
2. Liu, Y.; Lu, Q.; Paik, H.Y.; Xu, X.; Chen, S.; Zhu, L. Design Pattern as a Service for Blockchain-Based Self-Sovereign Identity. *IEEE Softw.* **2020**, *37*, 30–36. [CrossRef]
3. Xu, J.; Xue, K.; Tian, H.; Hong, J.; Wei, D.S.L.; Hong, P. An Identity Management and Authentication Scheme Based on Redactable Blockchain for Mobile Networks. *IEEE Trans. Veh. Technol.* **2020**, *69*, 6688–6698. [CrossRef]
4. Terzi, S.; Savvaidis, C.; Votis, K.; Tzovaras, D.; Stamelos, I. Securing Emission Data of Smart Vehicles with Blockchain and Self-Sovereign Identities. In Proceedings of the 2020 IEEE International Conference on Blockchain (Blockchain), Rhodes, Greece, 2–6 November 2020; pp. 462–469. [CrossRef]
5. Toth, K.C.; Anderson-Priddy, A. Self-Sovereign Digital Identity: A Paradigm Shift for Identity. *IEEE Secur. Priv.* **2019**, *17*, 17–27. [CrossRef]
6. Preukschat, A.; Reed, D. *Self-Sovereign Identity: Decentralized Digital Identity and Verifiable Credentials*; Manning: Shelter Island, NY, USA, 2021.
7. Sporny, M.; Longley, D.; Sabadello, M.; Reed, D.; Steele, O.; Allen, C. Decentralized Identifiers (DIDs) v1.0: Core Architecture, Data Model, and Representations. 2022. Available online: https://www.w3.org/TR/did-core/ (accessed on 5 November 2022).
8. Sporny, M.; Longley, D.; Chadwick, D. Verifiable Credentials Data Model 1.0. Expressing Verifiable Information on the Web. 2021. Available online: https://www.w3.org/TR/vc-data-model/ (accessed on 10 November 2022).
9. Podgorelec, B.; Alber, L.; Zefferer, T. What is a (Digital) Identity Wallet? A Systematic Literature Review. In Proceedings of the 2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC), Los Alamitos, CA, USA, 27 June–1 July 2022; pp. 809–818. [CrossRef]
10. Sartor, S.; Sedlmeir, J.; Rieger, A.; Heidi Roth, T. Love at First Sight? A User Experience Study of Self-Sovereign Identity Wallets. In Proceedings of the 30th European Conference on Information Systems (ECIS 2022), Timisoara, Romania, 18–24 June 2022.
11. Lab:UM, B. Unlock Decentralized Identity with MetaMask: SSI Snap. 2022. Available online: https://blockchain-lab-um.github.io/ssi-snap-docs/ (accessed on 3 March 2023).
12. Fainusa, A.F.; Nurcahyo, R.; Dachyar, M. Conceptual Framework for Digital Wallet User Satisfaction. In Proceedings of the 2019 IEEE 6th International Conference on Engineering Technologies and Applied Sciences (ICETAS), Kuala Lumpur, Malaysia, 20–21 December 2019; pp. 1–4. [CrossRef]
13. di Angelo, M.; Salzer, G. Characteristics of Wallet Contracts on Ethereum. In Proceedings of the 2020 2nd Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS), Paris, France, 28–30 September 2020; pp. 232–239. [CrossRef]
14. Liu, Y.; Lu, Q.; young Paik, H.; Xu, X. Design Patterns for Blockchain-based Self-Sovereign Identity. In Proceedings of the European Conference on Pattern Languages of Programs 2020, Virtual, 1–4 July 2020; pp. 1–14.
15. Mühle, A.; Grüner, A.; Gayvoronskaya, T.; Meinel, C. A survey on essential components of a self-sovereign identity. *Comput. Sci. Rev.* **2018**, *30*, 80–86. [CrossRef]

16. Steele, O.; Sporny, M. DID Specification Registries. 2022. Available online: https://www.w3.org/TR/did-spec-registries/#did-methods (accessed on 10 October 2022).
17. Six, N.; Herbaut, N.; Salinesi, C. Blockchain software patterns for the design of decentralized applications: A systematic literature review. *Blockchain Res. Appl.* **2022**, *3*, 100061. [CrossRef]
18. Alexander, C.; Ishikawa, S.; Silverstein, M.; Jacobson, M.; Fiksdahl-King, I.; Angel, S. *A Pattern Language: Towns, Buildings, Construction*; Oxford University Press: New York, NY, USA, 1977.
19. Dawes, M.J.; Ostwald, M.J. Christopher Alexander's A Pattern Language: Analysing, mapping and classifying the critical response. *City Territ. Archit.* **2017**, *4*, 2195–2701. [CrossRef]
20. Beck, K.; Cunningham, W. Using Pattern Languages for Object Oriented Programs. In Proceedings of the Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA), Orlando, FL, USA, 4–8 October 1987.
21. Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J. *Design Patterns: Elements of Reusable Object-Oriented Software*; Addison-Wesley Professional: Boston, MA, USA, 1994.
22. Krupitzer, C.; Temizer, T.; Prantl, T.; Raibulet, C. An Overview of Design Patterns for Self-Adaptive Systems in the Context of the Internet of Things. *IEEE Access* **2020**, *8*, 187384–187399. [CrossRef]
23. Gašparič, M.; Turkanović, M.; Heričko, M. Towards a comprehensive catalog of architectural and design patterns for blockchain-based applications—A literature review. In Proceedings of the CECIIS: Central European Conference on Information and Intelligent Systems: Proceedings: 31th International Scientific Conference, Varaždin, Croatia, 7–9 October 2020; pp. 259–265.
24. Enge, A.; Satybaldy, A.; Nowostawski, M. An offline mobile access control system based on self-sovereign identity standards. *Comput. Netw.* **2022**, *219*, 109434. [CrossRef]
25. Davie, M.; Gisolfi, D.; Hardman, D.; Jordan, J.; O'Donnell, D.; Reed, D. The Trust over IP Stack. *IEEE Commun. Stand. Mag.* **2019**, *3*, 46–51. [CrossRef]
26. EBSI. EBSI Specifications. 2022. Available online: https://ec.europa.eu/digital-building-blocks/wikis/display/EBSIDOC/ (accessed on 3 March 2022).
27. Decentralized Identity Foundation. Together We're Building a New Identity Ecosystem. 2022. Available online: https://identity.foundation/ (accessed on 5 April 2022).
28. Veramo. Veramo (v4.1.1) Specifications. 2022. Available online: https://veramo.io/docs/basics/introduction (accessed on 17 April 2022).
29. Sovrin. Sovrin Specifications. 2022. Available online: https://sovrin.org/library/sovrin-governance-framework/ (accessed on 29 April 2022).
30. Čučko, Š.; Keršič, V.; Turkanović, M. Self-Sovereign Identity Design Patterns Available online: https://gitfront.io/r/user-1520005/3GEr6tq4S43t/ssi-patterns/ (accessed on 2 March 2023).
31. EBSI. Trusted Registries ESSIF. 2022. Available online: https://ec.europa.eu/cefdigital/wiki/display/EBSIDOC/1.3.2.4.+Trusted+Registries+ESSIF+v2 (accessed on 6 March 2022).
32. EBSI. Trusted Schemas Registry. 2022. Available online: https://ec.europa.eu/digital-building-blocks/wikis/display/EBSIDOC/1.3.2.5.+Trusted+Schemas+Registry+ESSIF+v2 (accessed on 6 March 2022).
33. EBSI. Nodes & Ledgers (DID & Revocation/Endorsement Registries). 2022. Available online: https://ec.europa.eu/digital-building-blocks/wikis/pages/viewpage.action?pageId=380862983 (accessed on 10 March 2022).
34. EBSI. DID Registry API. 2022. Available online: https://ec.europa.eu/cefdigital/wiki/display/EBSIDOC/DID+Registry+API (accessed on 10 March 2022).
35. EBSI. Natural Person Onboards On ESSIF. 2022. Available online: https://ec.europa.eu/cefdigital/wiki/pages/viewpage.action?pageId=379913702 (accessed on 15 March 2022).
36. EBSI. EBSI Verifiable ID. 2022. Available online: https://ec.europa.eu/digital-building-blocks/code/projects/EBSI/repos/json-schema/browse/schemas/ebsi-vid (accessed on 17 March 2022).
37. EBSI. Verifiable ID—Natural Person. 2022. Available online: https://ec.europa.eu/digital-building-blocks/wikis/display/EBSIDOC/Verifiable+ID+-+Natural+Person (accessed on 17 March 2022).
38. EBSI. Verifiable ID—Legal Entity. 2022. Available online: https://ec.europa.eu/digital-building-blocks/wikis/display/EBSIDOC/Verifiable+ID+-+Legal+Entity (accessed on 17 March 2022).
39. EBSI. Verifiable Attestation. 2022. Available online: https://ec.europa.eu/digital-building-blocks/wikis/display/EBSIDOC/Verifiable+Attestation (accessed on 20 March 2022).
40. EBSI. Verifiable Attestation for ID. 2022. Available online: https://ec.europa.eu/digital-building-blocks/wikis/display/EBSIDOC/Verifiable+Attestation+for+ID (accessed on 20 March 2022).
41. EBSI. Issuance of Verifiable Attestations Based on Verifiable IDs. 2022. Available online: https://ec.europa.eu/digital-building-blocks/wikis/pages/viewpage.action?pageId=385876752 (accessed on 21 March 2022).
42. EBSI. What Is EBSI? 2022. Available online: https://ec.europa.eu/digital-building-blocks/wikis/display/EBSI/What+is+ebsi (accessed on 2 March 2022).
43. Walt.id. Delegation and Mandates. Available online: https://docs.walt.id/v/ssikit/usage-examples/verifiable-credentials/delegation-and-mandates (accessed on 8 August 2022).

44. EBSI. ESSIF Conceptual Reference for Architecture Definition. 2022. Available online: https://ec.europa.eu/digital-building-blocks/wikis/display/EBSIDOC/%5Barchived%5DESSIF+Conceptual+Reference+for+Architecture+Definition (accessed on 4 March 2022).
45. EBSI. Terminology. 2022. Available online: https://ec.europa.eu/digital-building-blocks/wikis/display/EBSIDOC/%5Barchived%5DTerminology (accessed on 1 March 2022).
46. EBSI. Diploma Accreditation. 2022. Available online: https://ec.europa.eu/digital-building-blocks/wikis/display/EBSIDOC/Diploma+Accreditation+Records (accessed on 21 March 2022).
47. EBSI. Verifiable Accreditation. 2022. Available online: https://ec.europa.eu/digital-building-blocks/wikis/display/EBSIDOC/Verifiable+Accreditation (accessed on 22 March 2022).
48. EBSI. Trusted Accreditation Body Accredits an Educational Organisation. 2022. Available online: https://ec.europa.eu/digital-building-blocks/wikis/pages/viewpage.action?pageId=380862832 (accessed on 12 March 2022).
49. EBSI. On-Boarding Legal Entities Flows Clarifications. 2022. Available online: https://ec.europa.eu/digital-building-blocks/wikis/pages/viewpage.action?pageId=489652740 (accessed on 16 March 2022).
50. EBSI. On-Boarding Legal Entities Example. 2022. Available online: https://ec.europa.eu/digital-building-blocks/wikis/display/EBSIDOC/On-boarding+legal+entities+-+example (accessed on 16 March 2022).
51. EBSI. TIR Supporting Flows. 2022. Available online: https://ec.europa.eu/digital-building-blocks/wikis/display/EBSIDOC/TIR+supporting+flows (accessed on 6 March 2022).
52. EBSI. Users Onboarding API. 2022. Available online: https://ec.europa.eu/digital-building-blocks/wikis/display/EBSIDOC/Users+Onboarding+API#UsersOnboardingAPI-Datamodels, (accessed on 15 March).
53. Veramo. 2022. Selective Disclosure Request. Available online: https://veramo.io/docs/veramo_agent/sdr_request/ (accessed on 28 March 2022).
54. EBSI. Verifiable ID & Verifiable Attestation Modelling. 2022. Available online: https://ec.europa.eu/digital-building-blocks/wikis/pages/viewpage.action?pageId=385876700 (accessed on 22 March 2022).
55. Veramo. DID Methods. 2022. Available online: https://veramo.io/docs/veramo_agent/did_methods/ (accessed on 26 April 2022).
56. Walt.id. Decentralized Identifiers (DIDs). 2022. Available online: https://docs.walt.id/v/ssikit/ssi-kit/ssi-kit/tech-stack/decentralized-identifiers-dids (accessed on 28 April 2022).
57. Deventer, O.; Lundkvist, C.; Csernai, M.; Hartog, K.D.; Sabadello, M.; Curren, S.; Gisolfi, D.; Varley, M.; Hammann, S.; Jordan, J.; et al. Peer DID Method Specification. 2022. Available online: https://identity.foundation/peer-did-method-spec/ (accessed on 11 November 2022).
58. Longley, D.; Zagidulin, D.; Sporny, M. The did:key Method. 2022. Available online: https://w3c-ccg.github.io/did-method-key/ (accessed on 7 October 2022).
59. Vila, X. SSI eIDAS Bridge (SEB) Project Summary. 2022. Available online: https://gitlab.grnet.gr/essif-lab/infrastructure/validated-id/seb_project_summary (accessed on 28 March 2022).
60. EBSI. Technical Specification (15)—eIDAS Bridge for VC-eSealing. 2022. Available online: https://ec.europa.eu/digital-building-blocks/wikis/display/EBSIDOC/%5Barchived%5DTechnical+Specification+%2815%29+-+eIDAS+bridge+for+VC-eSealing (accessed on 28 March 2022).
61. EBSI. Trusted Registries ESSIF v2. 2022. Available online: https://ec.europa.eu/digital-building-blocks/wikis/display/EBSIDOC/1.3.2.4.+Trusted+Registries+ESSIF+v2 (accessed on 11 March 2022).
62. Hughes, R. SSI and the Cloud. 2022. Available online: https://trinsic.id/ssi-cloud/ (accessed on 15 May 2022).
63. Evernym. Connect.Me: Your Wallet Just Went Digital. 2018. Available online: https://www.connect.me/ (accessed on 15 May 2022).
64. Hardman, D. Aries RFC 0103: Indirect Identity Control. 2019. Available online: https://github.com/hyperledger/aries-rfcs/tree/main/concepts/0103-indirect-identity-control (accessed on 19 April 2022).
65. Hardman, D. Guardianship-Sample. 2019. Available online: https://github.com/hyperledger/aries-rfcs/tree/main/concepts/0103-indirect-identity-control/guardianship-sample (accessed on 20 April 2022).
66. Hardman, D.; Harchandani, L. Aries RFC 0104: Chained Credentials. 2019. Available online: https://github.com/hyperledger/aries-rfcs/blob/main/concepts/0104-chained-credentials/README.md (accessed on 19 April 2022).
67. Dündar, Y.; Sertkaya, I. Self Sovereign Identity based mutual guardianship. *J. Mod. Technol. Eng.* **2020**, *5*, 189–211.
68. EBSI. Data Models and Schemas. Available online: https://ec.europa.eu/digital-building-blocks/wikis/display/EBSIDOC/Data+Models+and+Schemas (accessed on 21 March 2022).

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.