mathematics

Article

From Model to Mechanism: Enforcing Delegated Authority in SSI with Language-Based Security

Muhamed Turkanović [1,*], Vid Keršič [1], Alen Horvat [2], Dominik Beron [3] and Špela Čučko [1]

[1] Faculty of Electrical Engineering and Computer Science, University of Maribor, Koroška Cesta 46, 2000 Maribor, Slovenia; vid.kersic@um.si (V.K.); spela.cucko@um.si (Š.Č.)
[2] Netis, Blockchain Technologies d.o.o.—OU Tržaška, 1000 Ljubljana, Slovenia; alen.horvat@netis.si
[3] walt.id GmbH, 1190 Vienna, Austria
* Correspondence: muhamed.turkanovic@um.si

Abstract

Delegation of authority remains a critical yet insufficiently addressed capability in Self-Sovereign Identity (SSI) systems. Building on an existing delegation model that introduced the concept of a Verifiable Mandate (VM) for expressing authority and access rights, this paper extends the approach with a rigorous formalization of delegation semantics, enabling unambiguous reasoning over roles, grants, and constraints. The formal model is aligned with standards from the World Wide Web Consortium (W3C), and its constructs are embedded into an extended credential schema that preserves compatibility with the Verifiable Credentials (VC) data model while introducing delegation-specific attributes. A generalized VM schema is defined, supporting both generic and business-specific instantiations, and ensuring structural and semantic interoperability. Policy compliance is operationalized through a policy-based enforcement architecture, where rules are authored in the Rego language and evaluated at runtime by the Open Policy Agent (OPA). The architecture incorporates trusted registries for schema and policy distribution, allowing verifiers to define and enforce context-specific delegation rules in a modular and interoperable manner. Validation through realistic scenarios, such as postal service and academic use cases, demonstrates how formal semantics, schema validation, and language-based policy enforcement can be combined to enable secure, verifiable, and context-aware delegation in SSI ecosystems.

Keywords: self-sovereign identity; delegation; verifiable mandate; formalization; policy-based enforcement; verifiable credentials; blockchain; language-based security

MSC: 68M10; 94A60; 68P30; 68Q85

check for updates

Academic Editor: Hui Cui

Received: 8 August 2025
Revised: 6 September 2025
Accepted: 9 September 2025
Published: 14 September 2025

Citation: Turkanović, M.; Keršič, V.; Horvat, A.; Beron, D.; Čučko, Š. From Model to Mechanism: Enforcing Delegated Authority in SSI with Language-Based Security. *Mathematics* **2025**, *13*, 2971. https://doi.org/10.3390/math13182971

Copyright: © 2025 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

The Self-Sovereign Identity (SSI) paradigm is a novel concept in Computer Science and Information and Communications Technology (ICT), which follows the principles of decentralization and user-centrism. In operational terms, it primarily focuses on digital Identity Management (IdM), whereby it also covers the control and access management of personal resources. The idea of SSI sprang up in 2015 within the digital identity community, based on the growing interest in blockchain technology, as the main usher of the age of decentralization [1]. At that time, it was realized that the core challenges of IdM (e.g., federated approach, lack of full data control, anonymity, and user-friendliness) could be

resolved by introducing technology, which was already proving to ensure decentralization and trust at a large scale.

Nowadays, SSI is gathering traction and interest in academia, as well as in the industry [2]. As such, a lot of attention is being devoted by the World Wide Web Consortium (W3C) and the Decentralized Identity Foundation (DIF) to define the scope and standardize the protocols and artifacts involved in the SSI ecosystem. The latter comprises various actors(i.e., holder, issuer, verifier), artifacts (i.e., Decentralized Identifiers, Verifiable Credentials, etc.), protocols (i.e., DID-comm, P2P, OIDC4VC, etc.), as well as core components (i.e., wallets, agents, VDR, etc.) [1]. At the center of the SSI or decentralized ecosystem of digital identity is thus the user, who, using a digital (mobile) wallet, personally and locally, generates its own Decentralized Identifiers (DID) with a so-called SSI agent component, which they further use to identify and connect to another party (i.e., issuer), which also has a DID and digital wallet. This party can safely and trustfully digitally sign and issue to the user an attestation in the form of a machine-readable verifiable credential (VC), also called verifiable attestation (VA), claiming an attribute to the user, whereby the VC is then stored securely and locally on the digital wallet of the user. The flow is finalized when the user connects to yet another party (i.e., verifier) using their digital wallet and presents the VC in the form of a Verifiable Presentation (VP), proving to the verifier a collection of claims received by various issuers. The presented simplified flow demonstrates the user-centric approach, which is further supported by the mobile-first design. However, the latter, as well as the introduction of the blockchain ledger, are not required to support the SSI paradigm [3].

Since SSI aims to bring control back to the users, it has also gathered interest from agencies, governments, and governmental organizations [4]. As such, it is worth mentioning the European Blockchain Partnership and its European Blockchain Service Infrastructure (EBSI), which is backed by the European Commission (EC) [5]. These partnerships and organizations are already exploring and piloting the usage of blockchain technology and SSI in the EU's Single Digital Market (SDM) spectrum. Specifically, a lot of attention was put towards the European Self-Sovereign Identity Framework (ESSIF), which actively explores and shapes the EU's SSI ecosystem and its integration and involvement in the current digital and organizational structures of the EU [6]. This, however, is easier said than done due to the legacy digital ecosystem and related technologies, their integration with the legal and governance framework (e.g., eIDAS), the EU's complexity, etc.

However, considering the current stage of SSI development, we can see through the piloting stages of various use cases of SSI (e.g., diploma use case) that it is headed towards public adoption. Nevertheless, for full adoption, it still requires solutions for various aforementioned governance challenges, as well as technical solutions for concepts, which are still not fully covered with SSI, among which **delegation** is one of the primary ones needed for the private as well as the business sphere. If the SSI paradigm should become the de facto technological foundation for modern digitalization processes, it has to provide a comprehensive technical and governance framework that mimics the real-world complexity. Thus, it has to enable the support of situations where a natural or legal person delegates their rights or processes towards another party (e.g., power of attorney). The professional and scientific community was not ambivalent toward the concept of delegation, also called Access or Authority Delegation (AD). There have already been some suggestions on how to incorporate AD into SSI. ESSIF also defined the concept of a delegation under the term Verifiable Mandate (VM), which is a subtype of a VC, alongside VA and Verifiable ID (V.ID) [7]. Ref. [8] later on introduced a comprehensive model for Authority and Access Delegation in SSI. That work proposed the concept of the Verifiable Mandate (VM) as a structured, transferable, and cryptographically verifiable mechanism for expressing delegation within

the SSI framework. The model captured key delegation elements, including the delegator (issuer), delegatee (holder), the associated roles, grants (permissions), and optional constraints (e.g., time, location). It supported one-time or multi-hop delegation, the inclusion of legal entities, anchoring mechanisms for auditability, and defined supporting system components like the VM Credential Validator, VM Schema Generator, Policy Engine, etc. While that model laid the foundational framework for enabling structured delegation in SSI, it left open important questions regarding policy definition, machine-enforceable validation, and formal definition of delegation applicability and acceptance. In this paper, we extend and complement that model by introducing a formal semantics layer, schema-policy alignment, and a policy-based enforcement architecture, enabling both issuers and verifiers to define, evaluate, and enforce delegation logic in a transparent, modular, and verifiable manner.

1.1. Methodology, and Contribution

This paper follows a layered and structured methodology, starting from the problem identification and conceptual modeling of delegation in Self-Sovereign Identity (SSI), to the definition of machine-readable schemas and runtime-enforceable policies. Building on the previously published delegation model, we introduce a formal semantics for policy-compliant delegation, integrate it with schema definitions, and implement a rule-based enforcement mechanism using language-based security controls. Our approach combines theoretical modeling, standards-compliant schema design (JSON Schema and JSON-LD), and applied policy enforcement through the Open Policy Agent (OPA) and its Rego language. We validate the architecture through reference implementations and delegation scenarios, structured around real-world role-grant-constraint mappings.

The main contributions of this paper are as follows: (1) a formal extension of the existing Verifiable Mandate (VM) model, introducing delegation semantics based on a logical consequence relation and a policy compliance verification function; (2) the development of a synthesized taxonomy of delegation scenarios, abstracting recurring patterns in roles, grants, and contextual constraints from both personal and professional domains; (3) the definition of a modular and extensible set of schemas for delegation credentials using JSON Schema and JSON-LD, including a general-purpose VM schema and a business-specific version with strict constraint enforcement; (4) the design and integration of a policy-based enforcement architecture, enabling runtime validation of VMs via externalized Rego policies executed within the OPA framework; (5) the implementation and evaluation of a reference system, demonstrating the practical feasibility of the proposed approach through schema-driven validation, dynamic policy resolution, and automated decision-making. Together, these contributions aim to operationalize delegation in SSI systems by bridging formal models, credential structure, and dynamic, verifiable policy enforcement.

1.2. Methodology and Contribution

This paper follows a layered and structured methodology, starting from the problem identification and conceptual modeling of delegation in Self-Sovereign Identity (SSI), to the definition of machine-readable schemas and runtime-enforceable policies. Building on the previously published delegation model, we introduce a formal semantics for policy-compliant delegation, integrate it with schema definitions, and implement a rule-based enforcement mechanism using language-based security controls. Our approach combines theoretical modeling, standards-compliant schema design (JSON Schema and JSON-LD), and applied policy enforcement through the Open Policy Agent (OPA) and its Rego language. We validate the architecture through reference implementations and delegation scenarios, structured around real-world role–grant–constraint mappings.

The main contributions of this paper are as follows: (1) A formal extension of the existing Verifiable Mandate (VM) model, introducing delegation semantics based on a logical consequence relation and a policy compliance verification function. (2) The development of a synthesized taxonomy of delegation scenarios, abstracting recurring patterns in roles, grants, and contextual constraints from both personal and professional domains. (3) We define a modular and extensible set of schemas for delegation credentials using JSON Schema and JSON-LD, including a general-purpose VM schema and a business-specific version with strict constraint enforcement. (4) The design and integration of a policy-based enforcement architecture, enabling runtime validation of VMs via externalized Rego policies executed within the OPA framework. (5) The implementation and evaluation of a reference system, demonstrating the practical feasibility of the proposed approach through schema-driven validation, dynamic policy resolution, and automated decision-making. (6) A detailed operationalization of mandate transferability, specifying conditions under which a VM can be re-delegated. (7) The explicit use of Trusted Schema and Trusted Policy Registries to ensure discoverability and interoperability of schemas and policies, aligning the model with verifiable data registry infrastructures.

Together, these contributions aim to operationalize delegation in SSI systems by bridging formal models, credential structure, and dynamic, verifiable policy enforcement.

Table 1 explicitly contrasts the previously published delegation model with the extensions presented in this manuscript.

Table 1. Comparison between the previously published delegation model [8] and this work.

Aspect	Core Model	This Work
Focus	Conceptual introduction of Verifiable Mandate (VM) and core delegation components.	Operationalization of VM through formalization, schemas, and runtime enforcement.
Formalization	None.	Formal semantics introduced, including logical consequence and verification function.
Schemas	Conceptually.	JSON Schema/JSON-LD definitions for VM and MandatePresentation, aligned with W3C.
Policies	Conceptually.	Rego/OPA-based enforcement; runtime evaluation against verifier-defined rules.
Registries	Conceptually.	Trusted Schema Registry and Trusted Policy Registry explicitly included.
Validation	Conceptually.	Reference implementation with realistic scenarios (e.g., postal service).
Transferability	Conceptually.	Detailed rules for mandate transfer, including scope subsetting and identity role switching.
Schema layering	Not considered.	Introduction of a general Verifiable Mandate schema as a baseline, extended by business-oriented schemas with stricter, context-specific requirements (e.g., postal delegation).

2. Background

2.1. Technical Preliminaries

2.1.1. Self-Sovereign Identity

Self-Sovereign Identity (SSI) is a decentralized, user-centric identity approach that enables entities to fully control and manage their digital information [9] without the involvement of any external authority [10]. It utilizes two standards defined by the World Wide Web Consortium (W3C), namely **Decentralized Identifiers (DID)** [11] and **Verifiable Credentials (VC)** [12], and involves three main entities, acting as holders, issuers, and verifiers that participate in the SSI ecosystem by using digital wallets that serve as a means for identification, authentication, communication, and personal storage. All the entities can

generate and manage multiple unique DIDs and associated cryptographic keys within their digital wallets independently of any third party [13] as DIDs are decoupled from centralized service/identity providers, certificate authorities, or domain name registries that serve as a root of trust in centralized systems. Therefore, entities are identified by DIDs and authenticated via cryptographic proofs such as digital signatures without the involvement of any entity other than the DID controller. DIDs are URIs, a new type of globally unique verifiable identifier that does not require a registration authority. They can be registered at a VDR [11] that serves as Decentralized Public Key Infrastructure (DPKI). However, there are also DID methods [14], such as DID:peer and DID:key, which allow interactions, completely outside of decentralized registers [3]. Entities can gather credentials from various sources, i.e., issuers, that can attest to identity claims/attributes in a safe, privacy-respecting, and trustworthy manner by digitally signing and issuing machine-readable Verifiable Credentials (VC). Afterward, VCs are securely and autonomously stored in a digital wallet under the user's control and can be managed and distributed to other entities, i.e., verifiers, when proof of identity is required. VCs are digital equivalents of physical credentials that contain information about the (i) subject of the credential, (ii) the issuing authority, (iii) the type of credential, (iv) information related to asserted attributes/claims, (v) evidence related to credential derivation, and (vi) information related to constraints of credential. Additionally, they include technologies, such as digital signatures, that make VC tamper-proof and trustworthy [12].

2.1.2. Role- and Attribute-Based Access Control (RBAC/ABAC)

Role-Based Access Control (RBAC) and Attribute-Based Access Control (ABAC) are two well-established models for Access Control (AC) in distributed systems. RBAC focuses on assigning permissions to roles rather than directly to users, thereby abstracting access rights and simplifying policy management. A user inherits permissions by being assigned to one or more roles, which in turn are authorized to perform specific actions on system resources [15]. RBAC is widely adopted in enterprise systems, offering a coarse-grained but easily manageable AC model.

ABAC extends this idea by introducing dynamic attributes that describe the user, resource, action, or environment. Access decisions are made based on policy rules that evaluate these attributes at runtime [16]. For instance, an ABAC policy may grant access only if the user's department is "Legal" and the time is within office hours. ABAC allows for more expressive and context-aware policies than RBAC, but often introduces increased complexity in policy authoring and enforcement.

In the context of this paper, RBAC and ABAC concepts are relevant in defining and enforcing delegation rules, where the assignment of roles (RBAC) and the evaluation of constraints (ABAC) form the foundation of policy enforcement using tools like OPA and Rego.

2.1.3. Indirect Identity Control

Indirect Identity Control (IIC) was introduced to SSI to deal with the real-world complexity of relationships between natural and legal persons and their intention and willingness to exert control on behalf of another [17]. However, the complexity of the real-world scenarios reveals three types of IIC, which were also classified as (i) **delegation**, (ii) **guardianship**, and (iii) **controllership** [17]. All three types are described as follows.

(i) The first type of IIC is delegation. There are two roles in delegation: delegatee and delegator. Delegators give the right to the delegatees to act on their behalf. A classical example is the power of attorney. It is the most self-sovereign type of IIC out of all three because both parties are always completely controlling their identity, i.e., private keys [17].

Delegation rights are represented in VC (VM), which are digitally signed by the delegator. Each VM can contain several constraints defined by the delegator (e.g., time, location, etc.), which are specific to a use case. The constraints prevent the overuse and abuse of delegation by a delegatee.

(ii) Guardianship is the least self-sovereign type of IIC [17]. Here, the two roles are called guardian and dependent. In guardianship, the dependent does not control their private keys and gives complete identity control to the guardian, who also, in most cases, takes legal responsibility for the dependent. An example of such a relationship is the child-parent relationship. This type of identity control is in danger of more external attacks, i.e., abuse cannot be prevented by cryptography and cybersecurity, and has many complications related to privacy. Thus the formal document describing the relationship is needed and must outline all constraints and conditions for guardianship [17].

(iii) Controllership is the most transparent type of IIC (both delegation and guardianship can be hidden from a third party) since the controlled things, i.e., device or pet, cannot control their own identity by themselves in the first place [17]. Its characteristics are similar to delegation, e.g., transparent, and guardianship, e.g., a formal document outlining the relationship (can be ownership of the thing) [17].

2.1.4. Delegation Based on Actors and Interactions

The main focus of this paper is IIC by delegation. However, this type of control can be further split into three categories, looking at it from the SSI actors or VC lifecycle perspective, i.e., issuer, holder, and verifier [18]. **Delegation by issuer** deals with the fact that an issuer wants to enable another entity to issue VC (VA) on its behalf, e.g., Visa Inc. delegates the issuance of credit cards to their partnering banks. The former is the so-called downward issuance delegation. Delegation by an issuer makes VC issuance much more scalable since multiple independent parties can issue the same type of VC (VA). In the EBSI and ESSIF ecosystem, Verifiable Accreditations are a particular type of delegation mechanism, adding constraints and limits to a VC issuance process. The second type is the **delegation by the holder**, which is the most common use case for delegation and is the most mentioned use case in the literature since it directly affects the end-user [18]. Holders can delegate the authority to another holder to act on their behalf, e.g., applying to a master's degree program with a diploma when holders cannot perform this action themselves. **Delegation by the verifier**, similar to delegation by the issuer, makes the verification process more scalable since many external or third parties can authorize/verify the VCs of the holders. The result of the verification process can then be sent to the delegator. Let us consider the delegation process based on interaction forms. These can then be split into Human-to-Human, Human-to-Machine, and Machine-to-Machine, whereby the former is the most used form and the focus of this research [19]. An example of such is the aforementioned scenario of applying for a master's degree, where a student would give their parents the right to fulfill such action in their name. As presented by [19], an example of the delegation based on Human-to-Machine is a user who delegates a web application some sub-set of their rights, as is the case with the OAuth protocol [20]. The Machine-to-Machine delegation is built upon the notion that one machine delegates some rights to another in a pre-defined and semi-automatized way.

2.1.5. Language-Based Security

Language-based security is one of the standard techniques in computer systems to define security and access policies in high-level language instead of in executable programs [21]. This approach enables decoupling AC policies and the guarded programs or data, e.g., application programming interfaces (API), databases, etc. [22,23]. A similar

approach was also applied to several delegation schemes, where delegation between users was represented in custom-made domain-specific languages (DSL) [24,25]. Representing delegation with DSL was also proposed for VCs by Hyperledger Aries using Simple Grant Language (SGL) [26]. With the complexity of information technology applications and computer systems, multiple policy languages became widely used, such as Open Policy Agent (OPA) and Common Expression Language (CEL) [27,28]. Because of the different characteristics of the systems, several ways to handle AC were also proposed, e.g., RBAC and ABAC [29].

2.1.6. Delegation Model Foundations

This paper builds on our previously published delegation model, which introduced the concept of a Verifiable Mandate (VM) as a credential-based mechanism for expressing authority and access delegation within SSI systems [8]. The model formalized a delegation structure involving two parties—a delegator (issuer) and a delegatee (holder)—connected through a digitally signed Verifiable Credential that encodes a specific **role**, one or more **grants** (permissions), and an optional set of contextual **constraints** (e.g., time, location). Each VM is cryptographically bound to its issuer and subject and may be evaluated against external verification rules.

The published model laid the groundwork for introducing delegation in SSI systems by defining a structured flow and system-level architecture for VM credentials. A key contribution of the paper was the proposal of a set of delegation-specific components, including the VM Schema Generator, VM Credential Generator, VM Credential Manager, VM Credential Validator, Verification Engine, Policy Generator, and Policy Engine. These components were designed to cover all phases of delegation—from schema definition, credential issuance, and validation to revocation and policy-based decision-making—and were aligned with the modular integration into SSI wallets and agents. Their specification, while technology-independent, clarified the architectural requirements and responsibilities of each actor in the delegation process. These components are reused and extended in this work to support policy-aware delegation enforcement, as we shift focus toward formalizing semantics and embedding runtime enforceability into the delegation lifecycle.

2.2. Governance Preliminaries

European Blockchain Service Infrastructure

European Blockchain Service Infrastructure (EBSI) is a joint undertaking of the so-called European Blockchain Partnership (EBP; includes all EU member states, Norway, and Lichtenstein), with the vision to leverage blockchain technology in such a way that it can be legally and technically used by the public administrations and businesses for services, which would benefit from such a technology, in a seamless and trustworthy way. The core of EBSI is a public permissioned (consortium) blockchain network with nodes distributed across the EBP partnering states, which are running both Hyperledger Besu and Fabric clients [30].

EBSI not only provides the technical backbone but also builds on use cases and projects, which are being designed and piloted recently. Such projects are Identity (based on SSI), Diploma (based on VC), Document traceability (based on the network's ledger), etc. The former is known by its acronym ESSIF (**European Self-Sovereign Identity Framework**). ESSIF's goal is to bring SSI to Europe. To do so, it fulfills different functions (i) governance and trust framework, where it drives the creation of an EU Governance and Trust Framework supported by Member States; (ii) standards and specifications, where it creates business, functional, and technical specifications, which make up the European "flavor of SSI"; and (iii) facilitate adoption, where it coordinates the adoption of SSI across Europe.

Part of the governance and trust framework of EBSI/ESSIF is to define and leverage a trusted, public ledger, i.e., Verifiable Data Registry (VDR), that serves as a single source of truth for vital information. As such, EBSI/ESSIF instantiates different Trust Registries, such as (i) **Trusted Issuers Registry** (TIR), which contains information about organizations (e.g., public keys, accreditations, legal name, etc.); (ii) **Trusted Accreditation Organisation Registry** (TAOR), which is similar to the TIR, but contains information about organizations that can authorize Issuers to issue credentials in regulated fields; and (iii) **Trusted Schemas Registry** (TSR), which contains information about semantic contexts and vocabularies, policies, and templates of VCs and their data models to ensure semantic interoperability.

A complete list of abbreviations and formal symbols used throughout the paper is provided in Appendix C for reference.

3. Related Works

In terms of the classical definition space of delegation mechanisms in Computer Science, research in this field is quite exhaustive [15,31,32] since Delegation is an important security policy for AC, which is a part of many application systems. It covers Discretionary Access Control (DAC), Mandatory Access Control (MAC), Role-Based Access Control (RBAC), Attribute-Based Access Control (ABAC), etc. The latter spans again into several models, like the Ability-Based Delegation Model (ABDM) or Attribute-Based Delegation Model (ABDM), etc. [32,33]. All of these define the basic components: roles, groups, permissions, constraints, etc. They also define the delegation processes and limitations, etc. Nevertheless, the models are based on the challenges of classical computer and application systems, focusing on the AC mechanisms built into them. However, as such, they are not directly applicable to the SSI paradigm and its connection to complex real-world use cases.

Considering the relation between SSI and blockchain technology, we further identified several papers on the topic of Blockchain-Based AC. However, as the keywords suggest, this research focuses mainly on using blockchain technology to ensure AC in various forms, like publishing policies, which express permission to access resources on the ledger [34]. Such an approach was also partially used in our proposed model. Furthermore, Ref. [35] addressed the challenge of distributing AC rights and attributes towards various centers but does not provide any proposal for general authorization and AC mechanisms focusing on delegation principles. The same applies to similar research focusing on the Internet of Things domain [36–38]. We thus conclude that, to the best of our knowledge, no published scientific work has been done with the direct focus on SSI and its delegation models. An example is Schardong and Custódio [39], who conducted a Systematic Mapping Study where they identified a study addressing VC Delegation [40], which, however, only focuses on the holder-centric structure of VCs and thus addresses the challenge of having holders that can transfer credentials without the subject's permission. Fedrecheski et al. [20] compared data models for attributes including Public Key Certificates (PKCERTs), Attribute Certificates (ACERTs), and VCs, and noted that only ACERTs support delegation in some limited form. Mahalle and Shinde [41] proposed OAuth-Based delegation using Decentralized Identifiers (DIDs) and Verifiable Credentials (VCs).

The Sovrin Foundation acknowledged the inadequacies/limitations of SSI to serve individuals without internet access or the legal capacity to control and manage their digital identity by themselves [42]. In a whitepaper [43] published in 2019, they defined three types of IIC, emphasizing their commonalities and differences as well as interconnectivity/interworking that can lead to a transition from one type of relationship to another. They explored core theoretical principles, including (i) various types and (ii) dynamic nature of the guardianship relationships, such as differences in time, context, permissions, etc., (iii) guardianship life cycle, from inception to creation, usage, and termination, (iv) risks

and vulnerabilities of such relationships, and (v) integration within Trust Over IP (ToIP) stack [43] defined by Hyperledger Aries RFC 0289 [44].

The Hyperledger Aries RFC 0103 [17] proposed a technical solution for implementing IIC based on Sovrin's findings. The solution includes (i) a proxy trust framework that specifies the rules and conventions in human and machine-readable form, (ii) a proxy credential that specifies a controlled entity and its proxy and precisely defines the nature and limits of their relationship, and (iii) a proxy challenge that evaluates the proxy credential according to the context in which they are presented. The solution is explored through the refugee use case and includes a sample guardianship trust framework [45] and a sample guardianship schema in JSON format [46]. The proposed solution is currently under study for implementation in various pilots and Proof-of-Concepts (PoC) and should be modified to meet all the needs of individual use cases.

Although both The Sovrin Foundation [43] and the Hyperledger Aries [17] focused on guardianship, the insights can serve as the foundation for further research into other types of indirect control relationships that should be further explored regarding various personal and user stories.

Dündar and Sertkaya [47] addressed IIC, focusing on custodianship as a subset of guardianship. The authors presented a joint custody use case demanding the consideration of two guardians who jointly manage the dependent's identity. They implemented the PoC, including a custody trust framework and credential schema based on prior solutions presented by the Sovrin Foundation and the Hyperledger Aries. Concluding that the proposed framework can be integrated without overloading the existing systems.

The Hyperledger Aries RFC 0104 [26] proposed Chained Credentials (CC) leveraging the VC data model [12]. Credential chaining alters trust dynamics and allows data origin from VCs to be tracked while preserving verifiable quality. Tracking is possible due to provenance data within VC that links it back to its source. Such credentials can be used to delegate privileges that unlock many new use cases and offline verification, resulting in an object capabilities (OCAP) solution similar to Authorization Capabilities for Linked Data (ZCAP-LD, originally OCAP-LD). ZCAP-LD was utilized for chaining capability documents in Veres One, a decentralized identity solution that allows, among other functions, a delegation of authority to other entities [48].

Mukta et al. [49] proposed a high-level credential-based framework for trust management that includes the concept of supportive credentials to create the 'Web of Trust'. Similar to CC, the proposed solution enables personal/private issuers who do not need a globally recognizable reputation [26], as the supporting credential specifies the issuer's trust in the holder of the supporting credential. Hence, "a credential chain can be created where the holder of the support credential becomes the issuer of the next credential in the chain." However, both solutions require a well-known, trusted issuer at the root of the credential chain [49].

W3C Verifiable Credentials Data Model [12] defined a possible subject-holder relationship that can be expressed within the VC Data Model, including relationships where the holder is not the subject of the VC. The specification contains multiple suggested ways to handle such relationships: (i) The issuer issues the VC_1 to the subject, who passes it to another holder together with a new VC_2 in which the holder to whom the VC_1 is being passed is the subject, and the original subject is the issuer of the new VC_2. (ii) The issuer issues the VC_1 to the subject, who passes it to another holder together with the relationship VC_2 issued by the subject of the original VC_1, indicating the relationship between them. (iii) The issuer issues the VC_1 and a relationship VC_2 directly to the holder. The holder is not the subject of VC_1. Relationship VC_2 indicates the relationship between the subject and the holder of VC_1. (iv) The issuer issues the VC_1 to the subject. VC_1 includes a relationship

property/claim that asserts the relationship between the subject and potential, e.g., third-party. Delegation is outside the scope of the VC Data Model specification, but VCs can be adopted and used for delegation purposes.

Lim et al. [40] proposed subject-centric VCs that, when used by a holder, i.e., proxy, require the involvement of a VC subject, as the VC Data Model does not take into account loss of control over VP and the secondary unauthorized credential transfer, making the VC holder-centric. The authors suggest that credentials should be encrypted with the subject's public key upon transfer, preventing the content from being visible without the subject's permission. The holder then creates a presentation (VP) without decryption, presents it to the verifier, who requests credential decryption from its subject, who can accept or deny it. Thus, the use of the original VC is limited and requires authorization and complementation of the VP. However, there are some limitations, as the presented approach is not fully compatible with selective disclosure, it does not allow transfer tracking, and it is not applicable to use cases where multiple subjects are specified in a VC.

4. Formalization of Delegation Semantics

To rigorously define the delegation model underpinning the proposed Verifiable Mandate (VM) structure [8], we formalize the core components of delegated authority using relational sets and logical mappings. Users (U), roles (R), grants (G), and constraints (C) are defined as finite sets, and their relationships are captured via typed relations such as User-Role (UR), Role-Grant (RG), and Grant-Constraint (GC).

We introduce the concept of a *business process* ($BProcess$) and *delegation policy* ($DPolicy$) to describe the context in which mandates are evaluated. A Verifiable Mandate is modeled as a tuple that binds a unique delegator and delegatee with a non-empty set of roles and grants, and optionally a set of constraints. We extend this model to support delegation transferability and define conditions under which such mandates remain valid across hops.

The evaluation of whether a VM conforms to an acceptable policy is expressed via a logical consequence relation ($\models$), and the final decision mechanism is captured by the δ function, which determines whether a VM is valid within the scope of a business process. This formal structure forms the basis for both policy expression in Rego and runtime validation within OPA.

Figure 1 visually illustrates the delegation flow and the relationships between roles, grants, and constraints that form the foundation of our formalization. The verifier defines a policy space consisting of valid combinations of roles, grants (actions), and applicable constraints. Based on this predefined policy space, the delegator *(VM issuer)* selects valid combinations and constructs a VM, which is subsequently issued to the delegatee *(VM holder)*. These elements—roles, grants, and constraints—correspond to the formal sets R, G, and C in our delegation model, and their interdependencies are formally captured through the relations RG, GC, and the derived triplets in $DPolicy$.

While the proposed model enables the formalization of many constraints in a machine-readable format (e.g., older than 18 years, VC issued by the Ministry of Education, delegation expires in 30 days, etc.) within the VM schema and corresponding policies, certain categories of rules remain difficult to formalize. These include soft, discretionary, contextual, or dynamic rules that inherently require human interpretation, oversight, or hybrid governance mechanisms.

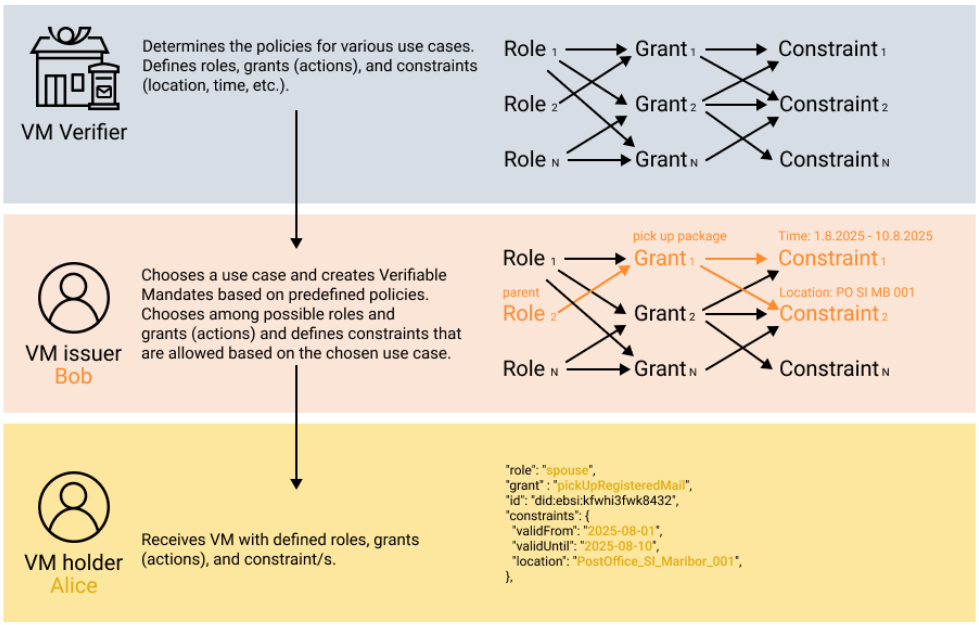


Figure 1. Visualization of the delegation chain across verifier, VM issuer, and VM holder. The verifier defines roles, grants, and constraints, which are selected by the delegator (VM issuer) to construct a VM issued to the delegatee (VM holder).

Formal Definitions

- Let a finite set of **users**, **roles**, **grants**, and **constraints** be denoted as $U = \{u_1, ..., u_n\}, R = \{r_1, ..., r_n\}, G = \{g_1, ..., g_n\}$ and $C = \{c_1, ..., c_n\}$
- **User-Role** relation is denoted as UR and defines a subset of a cartesian product of users and roles:
  - $UR \subseteq (U \times R) = \{(u_i, r_i) | u_i \in U \wedge r_i \in R\}$
- **Role-Grant** relation is denoted as RG and defines a subset of a cartesian product of roles and grants:
  - $RG \subseteq (R \times G) = \{(r_i, g_i) \mid r_i \in R \wedge g_i \in G\}$
- **Grant-Constraint** relation is denoted as GC and defines a subset of a cartesian product of grants and constraints:
  - $GC \subseteq (G \times C) = \{(g_i, c_i) \mid g_i \in G \wedge c_i \in C\}$
- A **business process**, denoted by $BProcess$, is defined over the following range:
  $BProcess := \{(r_i, g_i, c_i) \mid (r_i, g_i) \in RG_bp \wedge (g_i, c_i) \in GC_bp\}$,
  and defines a concrete business process, which is bound to a specific role or set of roles having grants and has its own set of constraint parameters by which it operates.
- A **delegation policy**, denoted by $DPolicy$, is defined as a set of valid triplets over roles, grants, and constraints:
  $DPolicy := \{(r_i, g_i, c_i) \mid (r_i, g_i) \in RG_p \wedge (g_i, c_i) \in GC_p\}$,
  and conforms to the following properties:
  - Delegation policy $DPolicy_i$ can be defined to suit a specific business process $BProcess_i$ or a set thereof: $\{BProcess_i, ..., BProcess_j\} \subseteq DPolicy$
  - Delegation policies $DPolicy_1 \neq DPolicy_2$ are distinguishable if $RG_1 \neq RG_2 \vee GC_1 \neq GC_2$, where $RG_1, GC_1 \subseteq DPolicy_1$ and $RG_2, GC_2 \subseteq DPolicy_2$.
- A **Verifiable Mandate** denoted by $VM \subseteq (U_i \cup (U_h R \times RG \times GC) \cup VM_i)$ specifies that a VM is based on a sub set of two specific users, also called holders/delegatees (U_h) and issuers/delegators (U_i) in VC, a non-empty set of roles the holder/delegatee is assigned to by the issuer/delegator, a non-empty set of grants/permissions he/she is

assigned to and an optional set of constraints the grants are set in, whereby absolutely one holder/delegatee and one issuer/delegator are defined, which are two different entitites and where itself can be based on another VM if transfered:

- $|U_h| \wedge |U_i| = 1$
- $u_i \in U_i \wedge u_h \in U_h \wedge u_i \neq u_h$
- $u_h \in U_h R \wedge u_i \notin U_h R$
- $U_h R \wedge RG \neq \emptyset$
- $GC = \emptyset \vee GC \neq \emptyset$
- $VM_i = \emptyset \vee VM_i \neq \emptyset$

- A Verifiable Mandate VM **is transferable** if and only if it holds a grant **TRANSFER**, whereby the previous holder is the new issuer and it can hold the same set of permissions and constraints or only a subset of those originally given by the issuer:
  - $r_i \in R \in VM_{original} \mid r_i = \text{TRANSFER}$
  - $VM_{delegated} \subset VM_{original}$
  - $u_i \in U_i, u_h \in U_h \wedge \{U_i, U_h\} \in VM_{original}$
  - $u_{ii} \in U_{ii}, u_{hh} \in U_{hh} \wedge \{U_{ii}, U_{hh}\} \in VM_{delegated}$
  - $u_i \neq u_{ii} \wedge u_h \neq u_{hh} \wedge u_h = u_{ii}$
  - $UR_{delegated} \subseteq UR_{original} \wedge RG_{delegated} \subseteq RG_{original} \wedge GC_{delegated} \subseteq GC_{original}$
- A **logical consequence** denoted $\vDash$ and defined as $DPolicy \vDash VM$, specifies that a specific verifiable mandate VM is a logical consequence of a policy $DPolicy$, if and only if it is specified (mapped) with the relations RG and GC, which are a subset of those relations defined in the policy, whereby the function mapping is general (i.e., non-bijective, surjective or injective):
  - $VM \subseteq UR_{VM} \times RG_{VM} \times GC_{VM}$
  - $DPolicy \subseteq RG_{policy} \times GC_{policy}$
  - $RG_{VM} \subseteq RG_{policy} \wedge GC_{VM} \subseteq GC_{policy}$
  - $R_{VM} \subseteq R_{policy} \wedge G_{VM} \subseteq G_{policy} \wedge C_{VM} \subseteq C_{policy}$
  - $\exists rg_j \in RG_{VM}$ such that $rg_j = rg_k \mid rg_k \in RG_{policy}$
  - $\exists gc_j \in GC_{VM}$ such that $gc_j = gc_k \mid gc_k \in GC_{policy}$
- The function **Verify Delegation**, which is being used by a specific service provider (i.e., verifier) as a boolean gate for its business process $BProcess$, is denoted as δ and defines $\delta : (VM, BProcess) \rightarrow \{0, 1\}$, satisfying a boolean result, either true or false:
  - $$\delta(VM, BProcess) = \begin{cases} 1, & \text{if } \exists DPolicy_i \text{ such that } DPolicy_i \supseteq BProcess \\ & \wedge DPolicy_i \vDash VM \\ 0, & \text{otherwise} \end{cases}$$

Defining the $DPolicy$ represents a critical step in the delegation model. It is authored by the verifier—typically the entity responsible for a specific business process ($BProcess$)—with support from the Policy and Schema Generator components. Once defined, the policy is published to a Trusted Policy Registry hosted on a Verifiable Data Registry (VDR). This allows future issuers (delegators)/holders (delegatees) to retrieve the verifier's policy using the VM Credential Generator and Validator components to understand the expected roles, grants, and constraints, and construct Verifiable Mandates (VMs) that comply with those predefined requirements.

The δ function serves as the core verification mechanism invoked by the model's dedicated **Policy Engine** component, enabling the evaluation of VM data against the corresponding delegation $DPolicy$.

Transferability of Mandates

Figure 2 illustrates a multi–step delegation chain: the initial delegator A issues a Verifiable Mandate VM_1 to B; B, empowered by a *transfer* grant, issues VM_2 to C while embedding a verifiable reference to VM_1 ("delegation proof 1"); C may analogously issue VM_3 to D with embedded proofs to VM_1 and VM_2. The final holder D presents a *MandatePresentation* containing VM_3 and the embedded chain to the verifier.

Security is enforced by (i) **cryptographic traceability**—each VM_k includes a verifiable reference (hash/URI + signature) to the prior mandate(s), yielding a tamper-evident chain; (ii) **cumulative constraints**—constraints are monotonic (downstream mandates may only *restrict* scope), ensuring that no hop can broaden roles, grants, or constraint sets beyond those allowed upstream; and (iii) **policy conformance**—the Policy Engine evaluates the normalized input (holder, roles, grants, constraints, chain) against the verifier's Rego policy, which checks, inter alia: holder–delegatee equality, allowed role–grant pairs, scope subsetting across the chain, validity windows, location limits, and revocation status.

The W3C VC Implementation Guidelines [50] defines a non-normative `nonTransferable` property indicating that a VC *must only* be encapsulated into a VP whose proof is created by the `credentialSubject`; a VP containing a VC with `nonTransferable: true` whose proof creator is *not* the `credentialSubject` is *invalid*. (See W3C VC Data Model section "§ 6.1.1 nonTransferable Property" (non-normative) [50].) In our context this provides a standards-aligned switch to *disable* onward delegation:

- If the delegator wants **no transfer**, set `nonTransferable: true` in the VM. Any attempt to re-delegate would fail because the next VP's proof would not be created by the original `credentialSubject`.
- If **controlled transfer** is allowed, omit the property (or set it to `false`) and require an explicit `transfer` grant; each hop must then issue a *new* VM that references the prior mandate(s), preserving chain traceability and cumulative constraints.

At presentation time, the verifier: (1) validates the VP and each VM signature; (2) verifies the delegation chain links ($VM_k \rightarrow VM_{k-1}$); (3) enforces `nonTransferable` (if true, reject any onward use); (4) checks scope monotonicity (roles, grants, constraints do not expand downstream); (5) evaluates time/location and status (revocation) constraints; and (6) runs the Rego policy to decide $\delta \in \{0, 1\}$.

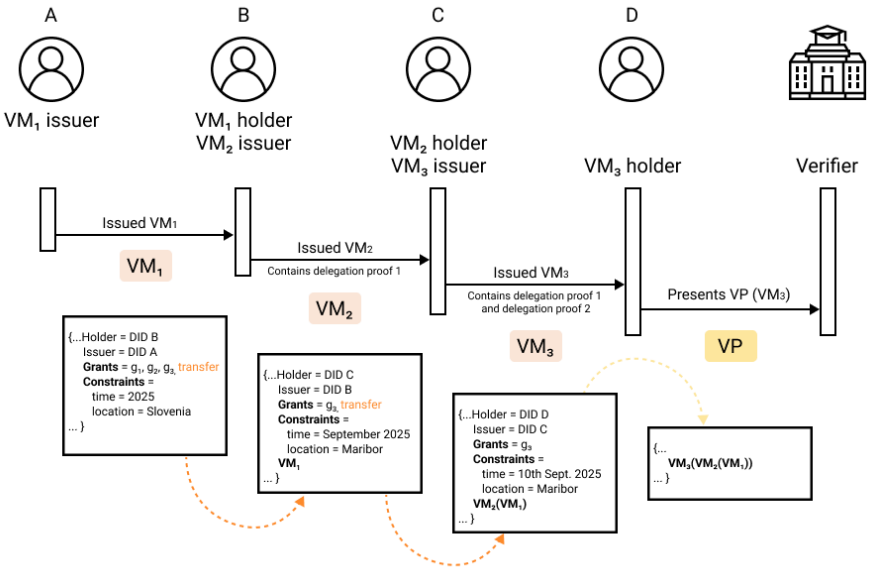


Figure 2. Transferable mandates across multiple holders.

5. Delegation Use Case Landscape

To operationalize delegation in Self-Sovereign Identity (SSI), it is essential to understand the breadth and diversity of real-world use cases. This section synthesizes use cases involving authority and access delegation by analyzing recurring actor **roles**, actions (**grants**), and contextual **constraints** across various personal and professional scenarios. Such an approach was needed to identify possible scenarios for using the model and to validate the model according to these.

The synthesis is based on a review of existing delegation taxonomies, user stories, and institutional requirements, combined with formal modeling of delegation logic. Patterns were extracted from both Human-to-Human and organizational delegation scenarios.

At the core of delegation semantics lies the triad of roles, grants, and constraints. Figure 3 illustrates a synthesized set of delegable actions (grants) mapped to various roles, highlighting possible permissions that may be bound by temporal or spatial constraints.

Several use cases of IIC were identified from the literature, including use cases for delegation, where delegates have different privileges and responsibilities and act on behalf of the delegators, performing interactions according to predefined permissions and constraints. Interactions may vary depending on the use case and delegator's needs/requirements and can include **permissions** or actions (grants), such as those defined in Figure 3. An example is the action **file** action, which could define actions, e.g., file an appeal or a lawsuit.



Figure 3. Identified set of possible permissions (actions or grants) that delegators can delegate to delegates.

A delegation can include different entities, including people, organizations, things, buildings, premises, and animals, as different types of IIC are closely related and can quickly intertwine and overlap. In general, delegation use cases can be divided into full or partial delegation of (i) tasks/work, (ii) guardianship role, including supervision and care for individuals and other related tasks, (iii) controllership role, including delegation of access, usage, and management of buildings, premises, things, and equipment.

In addition, delegation can be of a personal or professional nature and can be applied in various domains. Personal delegation addresses situations in which an individual delegates (part of) their responsibilities/tasks/work to another entity, usually their friends or relatives. For example, Bob authorizes Alice (i) to pick up his package at the post office,

(ii) to drive and pick up his child from school, or (iii) to enroll him in the University, (iv), etc. On the other hand, well-known examples of professional delegation might be found within the workplace environments where departments and employees have different delegated responsibilities and privileges in interactions that they do on behalf of the company. For example, (i) the company delegates the Human Resources department to recruit new employees, issue employment credentials, and further delegate credential issuing, (ii) the hospital delegates doctors to access medical records, examine and treat patients, prescribe prescriptions, access particular laboratory in the hospital, delegate responsibilities to other employees, etc. All these actions are a generalized list of possible permissions, which could be defined within the delegation process and used for building a taxonomy and/or generalized data model vocabulary. The list of the defined actions in Figure 3 is not a finite set, while the ordering and coloring do not represent any pattern or order besides the visualization purpose.

To further emphasize the landscape and complexity of the delegation process through use cases, we additionally analyzed the possible **roles** of the user, which can be singular or plural. Delegates and delegators are usually individuals who can take several different roles regarding a specific use case. These roles are bound to the use cases and are then predefined with the aforementioned permissions. Figure 4 depicts a landscape of user roles and their corresponding permissions across common delegation scenarios. It visualizes how contextual delegation patterns emerge across sectors such as healthcare, education, and e-government.

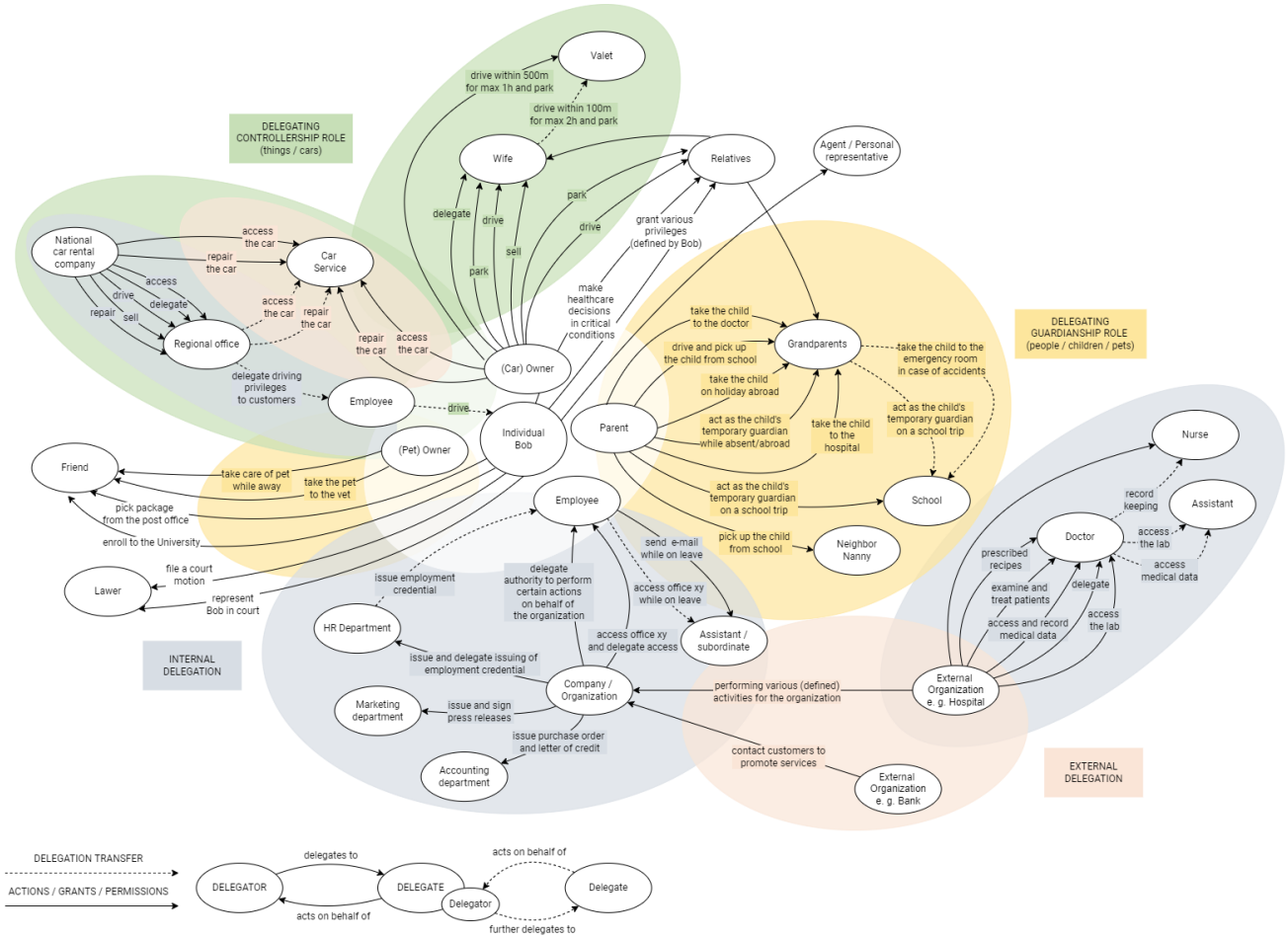


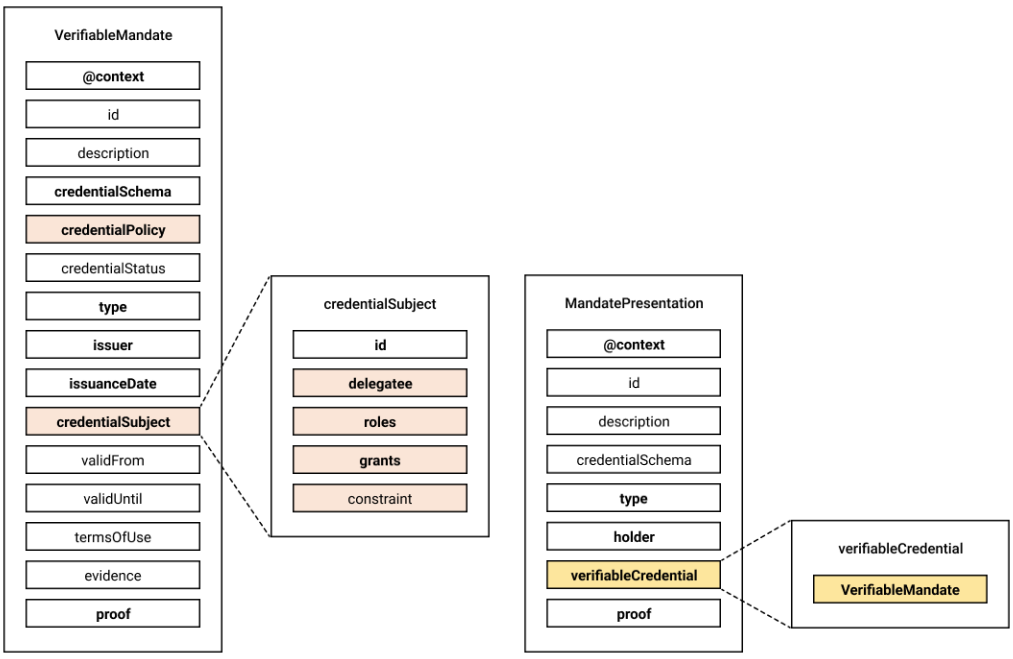
Figure 4. Identified landscape of delegation use cases from the perspective of user/individual's roles and possible permissions.

This synthesized taxonomy provides an empirical foundation for the delegation model proposed in this work. By identifying recurring patterns of roles, grants (actions), and constraints, we were able to abstract these elements into formal sets (R, G, C) and define their relations (RG, GC), which support the construction of delegation policies ($DPolicy$) and the logical consequence relation used in validation. These insights directly informed the vocabulary design for Verifiable Mandates, guided schema structuring, and enabled domain-relevant policy enforcement using Rego. Grounding the model in real-world scenarios was essential to ensure the flexibility, expressiveness, and practical applicability of the overall framework.

6. Delegation Data Models and Schemas

To enable the operationalization of our delegation model within the Self-Sovereign Identity (SSI) paradigm, we extend the core W3C Verifiable Credential (VC) structure by introducing new attributes aligned with the formal model defined in Section 4. These extensions are designed to remain compliant with the W3C VC Data Model specification, leveraging its flexibility while enriching it to support delegation-specific semantics. In doing so, we bridge the gap between abstract delegation logic and machine-verifiable credential structures.

Our starting point is the standard W3C VC object, where the `credentialSubject` field represents the subject about whom the issuer asserts certain claims. In the context of delegation, we interpret the `credentialSubject.id` as referring to the **delegator**—the person asserting that someone else is allowed to act on their behalf. This aligns with the semantic interpretation of a delegator stating, *"I, the delegator, assert that I authorize someone else to act on my behalf."* Within the `credentialSubject` object, we introduce the attributes `delegatee`, `roles`, `grants`, and `constraint`, which correspond directly to the formalized sets R, G, and C. The `delegatee` refers to the entity receiving the delegated authority, while `roles` and `grants` define the scope of that authority. The optional `constraint` field captures contextual limitations such as time and location. These extensions are visually depicted in Figure 5a, where fields introduced by our model are marked in orange.



(**a**) Data model for the VerifiableMandate. (**b**) Data model for the MandatePresentation.

Figure 5. Delegation-related data models: VerifiableMandate (**a**) and MandatePresentation (**b**).

Beyond the `credentialSubject`, we also introduce the `credentialPolicy` attribute at the top level of the VC. This field is not part of the core W3C specification but is necessary to support runtime policy enforcement. It provides a URI reference to an externally hosted policy (e.g., a Rego file), defined by the verifier as the owner of a specific business process (*BProcess* as defined in Section 4). The addition of this field allows for a dynamic link between static credential structure and executable delegation rules. Together, these attributes ensure that the Verifiable Mandate (VM) carries all information needed for both static validation (i.e., using a schema) and contextual evaluation (i.e., using a policy language-based enforcement).

To clarify the structure and purpose of each attribute, Table 2 presents a complete listing of all data model fields introduced or extended, along with their JSON type and requirement status. The values expected within `roles`, `grants`, and `constraints` are derived from the synthesized use case landscape in Section 5, ensuring semantic alignment between abstract patterns and structured schema vocabularies.

To formalize these structures, we define a **layered schema architecture** using JSON Schema. A schema defines the required structure (i.e., required attributes), data types of each attribute, and syntactic constraints of a VC, ensuring it adheres to a predictable format. At the base of the layered schema architecture is the **generalized VC schema** defined by W3C, which defines the core VC elements and their requirements. On top of that, we define a **general VM schema**, which generalizes from the base VC schema, while adding specialized objects for the alignment with the delegation model, i.e., requiring `delegatee`, `roles`, `grants`, and `credentialPolicy`, while treating `constraint` as optional. This schema allows flexibility across broad use cases and defers fine-grained enforcement to the policy level. To ease the understanding and validation, we drafted a schema for such a generalized VM in Appendix A, where we can deduce what attributes are required and optional, and how it uses the core VC schema as the basis.

All this is needed because of the proposed delegation model, which foresees a VDR, where business-specific policies and schemas would reside, all aligned with the formalized *BProcess* and *Dpolicy* in Section 4. These policies would be defined by the verifiers (owners of business processes or domains), whereby they define the fine-grained contextualized requirements, meaning that they allow only for specific actions (grant) only for specific roles, and maybe only during some specialized constraints (e.g., time, location). Thus, they will define their own policy and schema, which, however, will have to be built upon the generalized VM schema presented before, in order to enforce the delegation model and all the SSI flow. For this, we define a **business-oriented VM schema**, which builds on the general VM schema but allows specific constraints (e.g., `time` and `location`) to be mandatory within the `constraint` object. This schema is shown in Appendix B. They are structured using the `allOf` key in JSON Schema, enabling inheritance from general definitions and promoting reusability. This design ensures that domain-specific schemas remain aligned with the delegation model while preserving interoperability.

Figure 6 illustrates the relationship between the core VC schema, the generalized VM schema, and the business-oriented schema, as well as how they interface with policies. These schemas are expected to reside in a Trusted Schema Registry (e.g., EBSI TSR), while policies are registered in a Trusted Policy Registry. Together, these registries provide the necessary infrastructure for scalable and verifiable delegation across domains.

Table 2. Verifiable mandate data model: attributes, types, and requirements.

Attribute	Description	JSON Type	Required
	Top-level Verifiable Mandate Fields		
`@context`	JSON-LD context that defines semantic meaning of terms	array or string	Yes
`id`	Unique identifier of the credential (e.g., UUID or URL)	string	No
`description`	Human-readable summary of the mandate	string	No
`credentialSchema`	Link to the JSON Schema used to validate the structure of the VM	object	No (but recommended)
`credentialPolicy`	Link to the Policy file used to validate the meaning of the VM	object	No (but recommended)
`credentialStatus`	Status information (e.g., revocation, suspension)	object	No
`type`	Credential type array (e.g., `['VerifiableCredential'', "VerifiableMandate"]`)	array	Yes
`issuer`	DID or URI of the delegator issuing the mandate	string	Yes
`issuanceDate`	Date and time when the credential was issued	string (date-time)	Yes
`credentialSubject`	Object containing delegated claims and roles	object	Yes
`validFrom`	Optional lower-bound validity start time	string (date-time)	No
`validUntil`	Expiration date and time for the mandate	string (date-time)	No
`termsOfUse`	Usage terms or licensing conditions for the VC or VP	object or array	No
`evidence`	Supporting documents or data proving the claim	object or array	No
`proof`	Digital signature proving authenticity and integrity	object	Yes
	Nested `credentialSubject` Fields		
`id`	DID of the credential subject (typically the delegator)	string	Yes
`delegatee`	DID (or object) of the authorized delegate	string or object	Yes
`roles`	Role(s) assigned to the delegatee (e.g., parent, guardian)	array of strings	Yes
`grants`	Delegated permissions/actions (e.g., pickup, sign)	array of strings	Yes
`constraint`	Optional conditions on delegation (time, location, etc.)	object	No

In addition to schemas for VCs, we also define a schema for the **MandatePresentation** object—a specialized Verifiable Presentation (VP) that encapsulates at least one VM (Figure 5b). Although the W3C specifications primarily focus on VCs, schema definitions for VPs are allowed when specific validation logic is needed. In our case, this schema enforces that the VP performs the following: (i) Includes one or more valid VerifiableMandates; (ii) References a holder that matches the `delegatee` inside the included VM; (iii) Optionally includes additional supporting VCs.

This alignment ensures that when the delegatee presents a VP to the verifier, the delegation-specific component Policy Engine [8] can validate not only the structure but also the consistency between the presented identity and the authority granted in the VM.

Schemas play a foundational role in SSI by ensuring that credentials follow a predictable and verifiable structure. While policies are responsible for dynamic, context-dependent decisions, schemas guarantee that credentials are syntactically valid and semantically consistent with expected data models. In our architecture, schema validation serves as the first line of enforcement, followed by policy evaluation using tools like OPA and Rego. This dual-layer approach provides both structural and contextual assurance, ensuring that delegation credentials are not only valid in form but also meaningful in application.

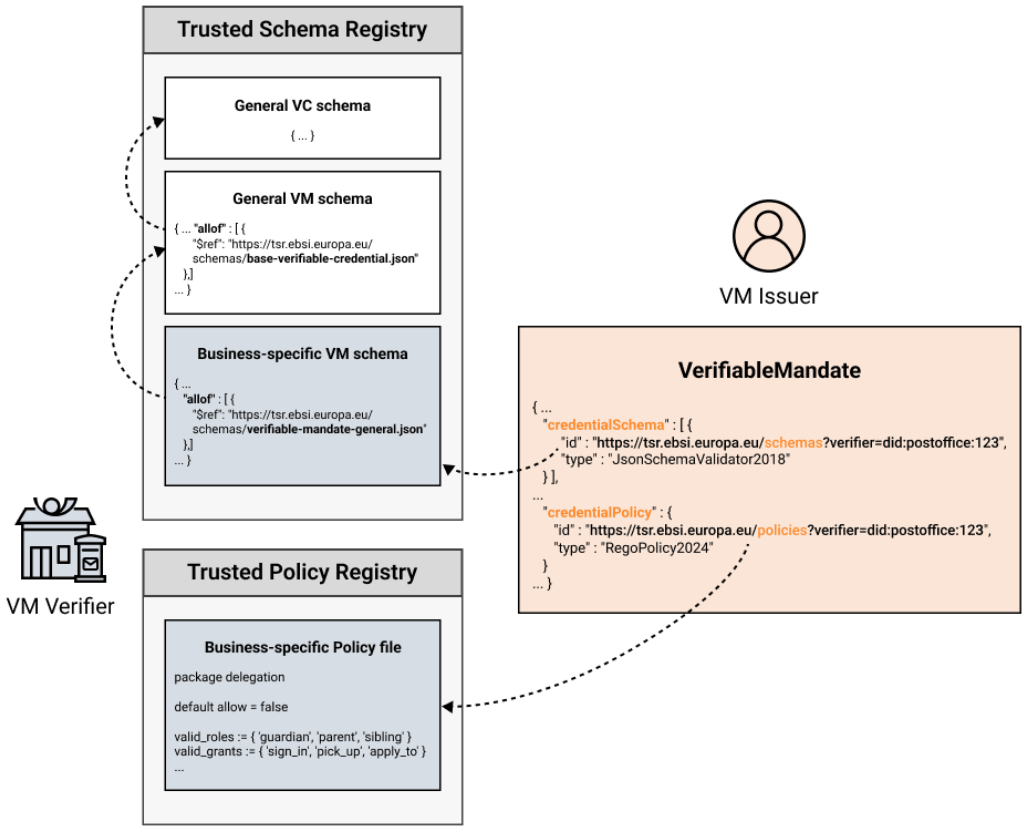


Figure 6. Relation between schemas, policies and Verifiable Mandates.

Intra-Domain Interoperability and Cross-Domain Limitations

The introduction of a *general-purpose Verifiable Mandate (VM) schema* is motivated precisely by the challenge of cross-domain interoperability. The universal schema defines a stable set of core constructs—*delegatee, roles, grants, and constraints*—which are aligned with the W3C Verifiable Credentials Data Model and serve as the minimal common denominator across all delegation scenarios. This baseline ensures that regardless of the business process or sector, the same structural and semantic principles apply to the core elements of a delegation credential.

On top of this universal schema, *business-oriented schemas* specialize the credential for concrete domains (e.g., postal services, education, healthcare). These schemas inherit from the general VM schema and add domain-specific attributes or constraint requirements, such as a postal tracking identifier in the postal scenario. This layered approach provides strong intra-domain consistency—verifiers within a given sector can rely on shared semantics and mandatory fields tailored to their workflows—while preserving interoperability with the universal schema.

However, the model does not attempt to guarantee *full cross-domain interoperability*. Semantic conflicts are inevitable when comparing, for example, grants in the postal sector with those in the healthcare or academic domains. As such, cross-domain alignment can only be ensured at the level of the universal VM schema, which enforces the presence and structure of the core delegation attributes. Beyond this, interoperability depends on domain-specific agreements and policies. This reflects an explicit scope decision: the proposed framework ensures structural and semantic consistency *within* domains and at the universal core, but does not aim to harmonize every possible cross-domain linkage. Attempting such global unification would require sector-wide governance processes and standardization efforts, which fall outside the scope of this work.

7. Policy-Based Enforcement Architecture

This section describes the full lifecycle of how delegated authority encoded in Verifiable Credentials (VCs) can be formally validated and enforced using a policy-based access control approach. The purpose of policy verification in this flow is to ensure that the delegation defined in a Verifiable Mandate (VM) is acceptable to the verifier, according to a policy that the verifier has defined (e.g., who can delegate what, to whom, under which conditions).

The previously published delegation model introduced a set of essential system components for supporting policy-compliant delegation in Self-Sovereign Identity (SSI) systems. These include the **Policy Generator**, the **Policy Engine**, and the **Verification Engine**, which jointly facilitate the construction, storage, retrieval, and enforcement of delegation-specific rules [8]. Building on this architectural foundation, we now define a policy-based enforcement architecture that integrates machine-readable policies, schema validation, and runtime evaluation logic.

At the center of this enforcement pipeline is the **Policy Engine**, which is responsible for dynamically verifying whether a received VM complies with the defined delegation policy. This component was already introduced in the original model, and it is here extended through integration with the *Open Policy Agent (OPA)* and its domain-specific language *Rego*.

OPA is a general-purpose policy engine designed to enforce declarative policies across systems and services. It decouples policy decision-making from application logic, making access control and governance more flexible, auditable, and portable. OPA is written in the Go programming language, maintained and open-sourced by the Cloud Native Computing Foundation. OPA can be used in three different modes depending on the system design: (1) Standalone microservice (e.g., RESTful API), (2) Embedded (Go) library, (3) WASM execution in the form of Rego policies compiled into WebAssembly and run inside wallets/apps/clients.

Rego is the high-level declarative language used to express policies in OPA. Inspired by Datalog, Rego allows users to define rules that evaluate structured data (typically JSON) and return decisions such as allow/deny, compliance reports, or structured responses. Rego operates over an input object and can query it against rules, sets, and logical conditions. Its power lies in separating business rules from application logic while remaining human-readable and machine-enforceable.

Rego policies are written as rule sets and stored in the **Trusted Policy Registry**, a registry introduced in the delegation model and distinct from the Trusted Schemas Registry (TSR) (Figure 6). The **Policy Generator** component is responsible for authoring and publishing these rules, tailored to the specific needs of a business process (*BProcess*) and the formalized delegation policy *DPolicy* (see Section 4).

The runtime enforcement process begins when a delegatee presents a *MandatePresentation* that includes one or more VMs. The **Verification Engine** component first validates the VM against the structural schema (using JSON Schema stored in the TSR) and then passes the validated content to the Policy Engine. Algorithm 1 governs this flow and is executed by the Verification Engine. Once the VM is structurally valid, Algorithm 2 is invoked by the Policy Engine, transforming the VM into a JSON input compatible with the referenced Rego policy. The Rego engine evaluates this input against the loaded rules and returns a Boolean decision $\delta \in \{0, 1\}$.

Algorithm 1 VM Credential Validation Workflow

Require: Verifiable Presentation VP containing Verifiable Mandate VM
Ensure: Validation result: `valid` or `invalid`
Extract *holder*, $proof_{VP}$ from VP
Resolve *holder* DID → fetch DID Document
VP Signature Validator: validate VP signature using *holder*'s public key
if VP signature is invalid **then**
    return `invalid`
end if
Extract VM from $VP.verifiableCredential$
Extract *id*, *issuer*, *proof*, *credentialSchema*, *validFrom*, *validUntil*, *credentialStatus* from VM
Resolve *issuer* DID → fetch DID Document
VM Signature Validator: validate digital signature using issuer's public key
if VM signature is invalid **then**
    return `invalid`
end if
VM Validity Period Validator: check current time ∈ [*validFrom*, *validUntil*]
if current time outside validity range **then**
    return `invalid`
end if
VM Schema Validator: fetch schema from *credentialSchema.id*
Validate structure and fields of VM against the schema
if schema validation fails **then**
    return `invalid`
end if
VM Status Validator: fetch credential status from *credentialStatus.id*
if status is `revoked` or `suspended` **then**
    return `invalid`
end if
VM Trust Builder:
if *issuer* is in trusted Legal Entity registry **then**
    trust ← `true`
else if VM contains additional credentials (VID, VA) **then**
    validate credentials and extract issuer identity
    trust ← result of validation
else
    trust ← `false`
end if
if trust is `false` **then**
    return `invalid`
end if
return `valid`

To illustrate the enforcement logic, consider the following simplified Rego policy example:

This policy (Listing 1), associated with a given *BProcess*, allows the grant `pickUpRegisteredMail` to be used *only* if the delegatee holds the role `spouse`, the location is `PostOffice SI Maribor 001`, and the date constraint matches `2025-08-01`. When a VM is submitted, the Policy Engine feeds the relevant fields into the Rego evaluator and receives either `allow = true` (accepted) or `false` (denied).

The `input` object in Rego corresponds to the deserialized VM structure, which is extracted from the `credentialSubject` of the presented VM. The alignment of these fields with the formal constructs R, G, and C, as well as the *DPolicy* specification, ensures coherence across the model, the credential schema, and the runtime policy evaluation.

It is important to distinguish the role of policies from schemas and contexts. **Schemas** define the structural expectations of a VM, i.e., what fields must exist and how they must be typed. **JSON-LD contexts** provide semantic meaning and support interoperability through shared vocabularies. **Rego policies**, in contrast, enforce *contextual and logical*

constraints—answering the question whether a particular delegation credential is valid under current conditions.

Algorithm 2 Policy Engine Delegation Evaluation Workflow

Require: Normalized Verifiable Mandate JSON VM_{norm}, including `credentialSubject` and `holder`, policy reference URL *policy_url*
Ensure: Delegation decision: `allow` or `deny`
Extract from VM_{norm}:
- $delegatee \leftarrow VM_{norm}.credentialSubject.delegatee$
- $roles \leftarrow VM_{norm}.credentialSubject.roles$
- $grants \leftarrow VM_{norm}.credentialSubject.grants$
- $constraints \leftarrow VM_{norm}.credentialSubject.constraint$
- $holder \leftarrow VM_{norm}.holder$

Step 1: Sanity Check
if $delegatee \neq holder$ **then**
    return `deny`
end if
Step 2: Fetch and Load Policy
Download policy file P from *policy_url*
Load Rego policy P into the OPA policy engine
Step 3: Prepare Input for Evaluation
Construct policy input object I with:
$I.delegatee \leftarrow delegatee$
$I.roles \leftarrow roles$
$I.grants \leftarrow grants$
$I.constraints \leftarrow constraints$
Step 4: Policy Evaluation
$decision \leftarrow$ evaluate(P, I)
if $decision =$ `allow` **then**
    return `allow`
else
    return `deny`
end if

Listing 1. Simplified Rego policy for contextual delegation.

```
package delegation

default allow = false

allow {
  input.role == "spouse"
  input.grant == "pickUpRegisteredMail"
  input.constraint.location == "PostOffice SI Maribor 001"
  input.constraint.time == "2025-08-01"
}
```

Finally, Figure 7 outlines the entire pipeline, from MandatePresentation submission, through schema validation and policy evaluation, to final access decision. This architecture enables verifiers to automate decision-making while retaining fine-grained control over the delegation logic, using modular, standards-compliant components.

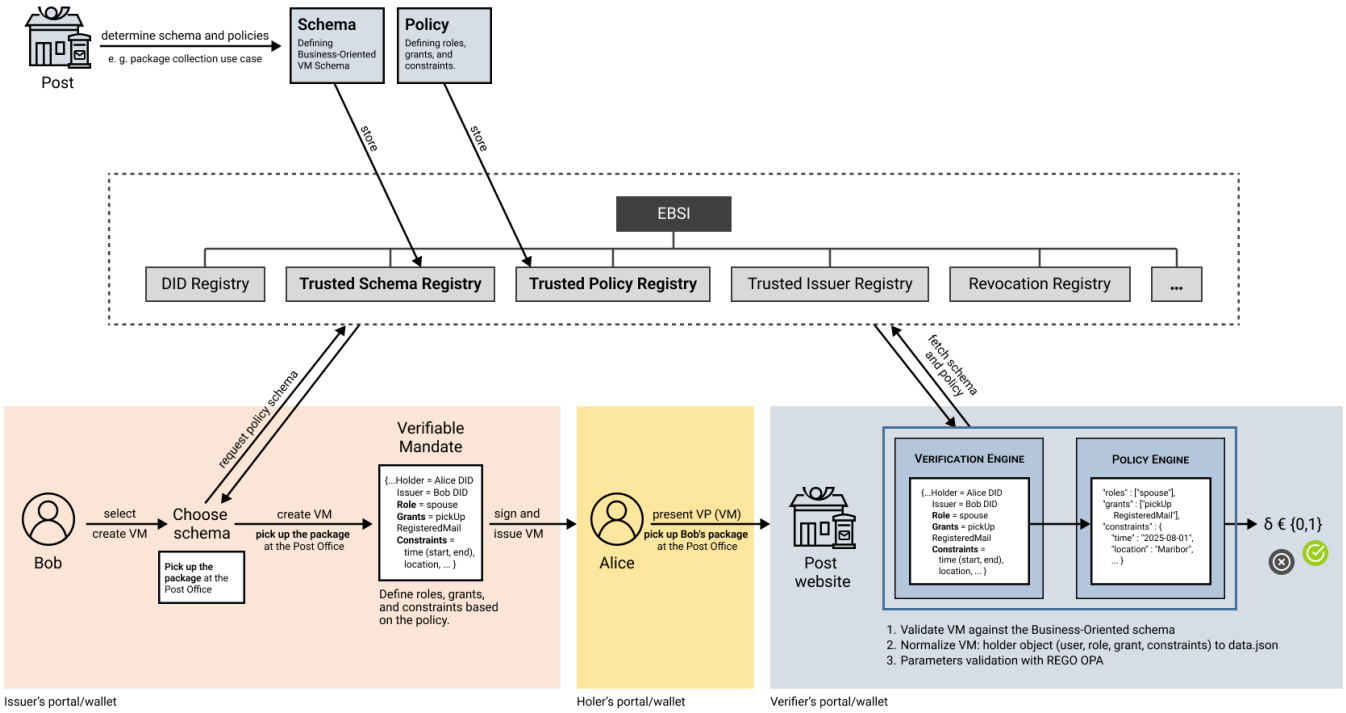


Figure 7. High-level delegation enforcement architecture and flow.

8. Validation

To validate the proposed delegation model's policy enforcement pipeline, we designed a reference implementation that supports the issuance, verification, and evaluation of Verifiable Mandates (VMs). Such a system implements both schema validation and policy compliance logic, enabling the verification of delegation scenarios end-to-end.

Figure 8 illustrates a realistic and complex delegation use case. A national postal operator (acting as the verifier) defines a policy (*DPolicy*) for its business process of ***mail retrieval delegation*** (*BProcess*). Two delegation scenarios are defined: the `pick_up_package` action can be granted to roles such as `friend`, `family`, or `neighbor`; whereas `pick_up_registered_mail` is restricted to the `spouse` role and must include both a valid tracking identifier and a supporting `MarriageCredential`. This high-assurance flow ensures that the delegatee is not only structurally and cryptographically validated, but also semantically bound to the delegator via an official credential. Both scenarios enforce the constraint of the specific post office's location and optionally allow date validations.

The system is composed of the delegation-specific components defined in the delegation model ([8]): the **Verification Engine**, responsible for structural schema checks (Algorithm 1), and the **Policy Engine**, which evaluates Rego policies (Algorithm 2). The Rego files, alongside the Business-oriented VM schema (see Appendix B), are authored by the verifier through the **Policy Generator**, and published in the Trusted Schema and Policy Registry, respectively, enabling issuers (i.e., delegators) to align with defined delegation requirements when issuing VMs.

Delegators access this registry when constructing a VM and validate its structure against the corresponding schema stored in the TSR and TPR. When the delegatee submits a Verifiable Presentation (i.e., MandatePresentation), the verifier-side Verification Engine validates the VM schema, and the Policy Engine applies the Rego policy to determine compliance. All policy decisions are local, deterministic, and isolated from the rest of the SSI flow.

Rego Policy file

```
package delegation

default allow := false

# High-level allow rule combining all checks
allow if {
  input.credentialSubject.delegatee == input.holder
  valid_grant_and_role
  valid_constraints
}

# Grant and Role Validation
valid_grant_and_role if {
  "pick_up_package" in input.credentialSubject.grants
  some r in input.credentialSubject.roles
  r in allowed_roles_for_package
}

# Registered mail: role must be spouse AND a MarriageCredential must be present
valid_grant_and_role if {
  "pick_up_registered_mail" in input.credentialSubject.grants
  "spouse" in input.credentialSubject.roles
  input.credentialSubject.constraint.registeredMailTrackingId
  has_marriage_credential
}

allowed_roles_for_package := ["friend", "neighbor", "family"]

# ---- Supporting credential checks ----------------------------------

has_marriage_credential if {
  input.supportingCredentials  # guard existence
  some sc in input.supportingCredentials
  "VerifiableCredential" in sc.type
  "MarriageCredential" in sc.type
}

# Constraint Validation
valid_constraints if {
  startswith(input.credentialSubject.constraint.location, "PostOffice_SI_")
  valid_date_window
}

valid_date_window if {
  not missing_date_fields
  input.credentialSubject.constraint.validFrom <= input.currentDate
  input.currentDate <= input.credentialSubject.constraint.validUntil
}

missing_date_fields if {
  not input.credentialSubject.constraint.validFrom
} else if {
  not input.credentialSubject.constraint.validUntil
}
```

JSON Input

```
{
  "holder": "did:example:alice",
  "currentDate": "2025-08-06",
  "credentialSubject": {
    "delegatee": "did:example:alice",
    "roles": ["spouse"],
    "grants": ["pick_up_registered_mail"],
    "constraint": {
      "location": "PostOffice_SI_Maribor_001",
      "registeredMailTrackingId": "SI998877665PO",
      "validFrom": "2025-08-01",
      "validUntil": "2025-08-10"
    }
  },
  "supportingCredentials": [
    {
      "id": "cred-marriage-001",
      "issuer": "did:gov:si-mju",
      "type": ["VerifiableCredential", "MarriageCredential"],
      "credentialSubject": {
        "spouseA": "did:example:delegatorA",
        "spouseB": "did:example:alice"
      },
      "expirationDate": "2028-12-31"
    }
  ]
}
```

OPA Result

```
{
  "allow": true,
  "allowed_roles_for_package": ["friend", "neighbor", "family"],
  "has_marriage_credential": true,
  "valid_constraints": true,
  "valid_date_window": true,
  "valid_grant_and_role": true
}
```

Figure 8. Rego policy evaluation for a post-office delegation with high-assurance flow. The spouse is authorized to pick up registered mail only if location and date constraints hold and a `MarriageCredential` is included in the presentation.

A snapshot of representative policy validation results is provided in Table 3, while a complete set of test cases, Rego files, and sample Verifiable Presentations are publicly available in our GitHub repository https://github.com/blockchain-lab-um/ssi_delegation_rego_policy (accessed on 6 September 2025).

This integrated setup confirms that the delegation model, schemas, and policy logic can be operationalized in a modular and standards-aligned fashion, supporting a variety of real-world use cases, while also accommodating more complex delegation scenarios that require multi-credential validation.

Table 3. Sample Rego policy evaluation results. A subset of test cases is shown; full logs are available in the GitHub repository.

VM ID	Role	Grant	Supporting Credential	Allow
vm-001	spouse	pick_up_registered_mail	MarriageCredential	`true`
vm-002	spouse	pick_up_registered_mail	none	`false` *
vm-003	friend	pick_up_package	none	`true`
vm-004	neighbor	pick_up_registered_mail	MarriageCredential	`false` **
vm-005	family	pick_up_package	none	`true`

* Denied due to missing required supporting `MarriageCredential`. ** Denied due to the wrong role `neighbor`.

Technical Setup

From a technical perspective, the reference implementation was designed as a microservice-based architecture consisting of several specialized services:

- The verifier backend encapsulated the business logic governing delegation processes;
- The OPA microservice was responsible for performing policy validations;
- The EBSI blockchain node (test environment);
- The EBSI API service, handling read/write access to the EBSI blockchain network.

The EBSI blockchain node's sole responsibility was to remain synchronized with the EBSI network, whereas the API services handled read and write operations, thereby allowing services to interface with the blockchain in a scalable manner. This separation ensured that the more demanding application workloads could be managed by horizontally scaling the API services. To reduce redundant communication overhead between services—particularly given the frequent read access from the blockchain node—caching mechanisms were introduced, lowering bandwidth consumption and improving response latency.

The chosen microservice architecture offered elasticity, as individual services could be scaled independently depending on workload requirements. Deployment was carried out primarily on Amazon Web Services (AWS), with the exception of the EBSI blockchain node, which was required to run on premises to comply with consortium security requirements. The orchestration of services was managed with Kubernetes, which provided reliable scaling and fault tolerance. The OPA ecosystem, widely adopted in industry by organizations with large-scale workloads, further facilitated efficient deployment of validation services by leveraging its mature tooling and integration support. This combination of microservices, container orchestration, and industry-hardened validation frameworks enabled the implementation to remain both resilient and adaptable under varying future operational demands.

9. Discussion

The presented delegation model, along with the schema and policy enforcement mechanisms, adheres to existing W3C specifications wherever applicable. Nevertheless, we emphasize that the concept of a *Verifiable Mandate (VM)* and its corresponding generalized schema—while aligned with the core Verifiable Credential (VC) data model—requires a broader formalization and standardization across the SSI ecosystem. Without such efforts, implementers are likely to define inconsistent structures and semantics for delegation, which may delay the unification of approaches and hinder interoperability.

One of the foundational components of our model is the inclusion of *Trusted Registries*, specifically the *Trusted Schema Registry (TSR)* and the *Trusted Policy Registry (TPR)*, forming a part of the Verifiable Data Registry (VDR). In this work, we propose leveraging the

existing EBSI infrastructure to host and manage these registries. EBSI aligns well with the objectives of promoting trust, decentralization, and cross-border interoperability. However, we also recognize that other VDR platforms may be used. This flexibility, while offering architectural freedom, presents the risk of fragmentation—verifiers may be forced to query and interpret schemas and policies across incompatible infrastructures. This again emphasizes the need for consensus and shared standards.

From a technical perspective, we have demonstrated that delegation can be operationalized through the combination of schema validation and runtime enforcement via language-based policy rules. However, the process of authoring and maintaining such policies and schemas remains burdensome for non-technical stakeholders. Therefore, we stress the need for a well-designed, modular *Policy and Schema Generator* component. This was already envisaged in the original model and will be essential for practical implementation, allowing business users to define contextual role–grant–constraint structures in a usable and verifiable manner.

The urgency of supporting standardized delegation is further underlined by the recently adopted *European Regulation on the European Digital Identity Wallets (EUDIW)*. The regulation builds on the principles of SSI and mandates high trust assurance and interoperability across digital identity systems. Given the wide applicability of delegation in legal, administrative, and commercial contexts, we argue that a standardized delegation mechanism, such as the one we propose, is a necessary component for enabling full adoption and integration of these wallets.

Lastly, although the presented model and implementation focus on holder-side delegation, future research must also address guardianship, controllership, and more complex cross-domain delegation scenarios. The formal foundations introduced in this work allow for such extensions, but they require further alignment with both legal frameworks and governance models.

10. Conclusions

This paper builds upon our previously published delegation model for Self-Sovereign Identity (SSI), extending it with formal delegation semantics, W3C-aligned schema definitions, and runtime policy enforcement through language-based security using the Open Policy Agent (OPA) and its Rego language. By aligning formal models, credential structure, and contextual policy enforcement, we demonstrate a standards-compliant approach to operationalizing delegation in SSI. The proposed architecture integrates schema validation, policy retrieval, and automated decision-making, providing a modular and verifiable framework suitable for both public and private sector use cases.

10.1. Limitations

This work inherits several scoped decisions from the original delegation model, which introduced the Verifiable Mandate (VM) as a mechanism for expressing authority and access delegation in SSI. The original model deliberately excluded broader Indirect Identity Control (IIC) constructs such as guardianship and controllership, and focused primarily on Human-to-Human delegation from the perspective of the holder (delegatee), without fully modeling the governance processes of issuers or verifiers.

While our contributions extend this foundation with formal semantics and enforceable policies, the scope remains at the Verifiable Credential layer and does not address DID Document-based delegation mechanisms (e.g., via the `capabilityDelegation` attribute). The evaluation is limited to natural person delegates and does not yet address organizational or autonomous agent delegation scenarios.

Furthermore, the approach assumes that verifiers host and control the policy logic and related infrastructure, and that trusted schema and policy registries are available—assumptions that align with ecosystems such as EBSI but are not yet universally adopted. Although the schema–policy framework is extensible, operational considerations such as automated revocation, prevention of misuse, and resilience against adversarial behavior remain beyond the scope of this work.

10.2. Future Work

Future research will focus on extending the formal semantics to cover *delegation chains*, *transitive delegation*, and other IIC types, including guardianship and controllership. A priority is contributing to *W3C standardization* efforts by formalizing the VM type and defining a generalized VM schema, to prevent fragmentation and ensure interoperability across SSI implementations.

The Trusted Schema Registry and Trusted Policy Registry components could be enhanced to support multi-VDR environments, standardized schema–policy bindings, and reusable policy templates. Developing a fully functional and modular *Policy/Schema Generator* to guide verifiers in creating compliant delegation rules is essential for real-world adoption.

Given the recent adoption of the EU Regulation on European Digital Identity Wallets, which builds upon SSI concepts, advancing delegation support is critical to ensure smooth integration into regulated identity frameworks. Additional research directions include richer constraint models (e.g., geofencing, recurring time windows), cross-jurisdictional legal encoding of delegation, offline policy verification for mobile wallets, and usability studies to ensure that delegation mechanisms remain secure, intuitive, and accessible for end users.

Author Contributions: Conceptualization, M.T., Š.Č., A.H.; Methodology, M.T. and Š.Č.; Formal analysis, M.T.; Investigation, V.K.; Data curation, V.K., A.H. and D.B.; Software, D.B. and V.K.; Validation, A.H.; Writing—original draft, M.T., Š.Č. and V.K.; Writing—review & editing, M.T., Š.Č., V.K., A.H. and D.B.; Visualization, Š.Č.; Resources, D.B.; Project administration, M.T.; Funding acquisition, M.T. and A.H.; Supervision, Š.Č. 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 P2-0057, and by the European Union's Horizon 2020 research and innovation program under grant agreement No 871932 (eSSIF-Lab) and its Cascading funding scheme (project AD and VM).

Data Availability Statement: The original data presented in the study are openly available in a publicly accessible repository GitHub at https://github.com/blockchain-lab-um/ssi_delegation_rego_policy (accessed on 6 September 2025).

Conflicts of Interest: Author Alen Horvat was employed by the Netis, blockchain technologies d.o.o.—OU Tržaška, author Dominik Beron was employed by the walt.id GmbH. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Appendix A. Verifiable Mandate (General) Schema

The following Listing A1 shows the full JSON Schema definition for the *Verifiable Mandate (General)* credential. This schema extends a standard W3C Verifiable Credential to support delegation-specific fields such as `delegatee`, `roles`, `grants`, and a flexible `constraint` object. Additionally, the schema enforces the presence of a `credentialPolicy` reference, which informs the verifier where to retrieve the matching Rego policy for validation.

Listing A1. Verifiable Mandate (General) JSON Schema.

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "$id": "https://your-university.edu/schemas/verifiable-mandate-general.json",
  "title": "Verifiable Mandate (General)",
  "description": "General Verifiable Mandate schema extending VC with delegatee, roles, grants, and constraint.",
  "allOf": [
    {
      "$ref": "https://tsr.ebsi.europa.eu/schemas/verifiable-credential.json"
    },
    {
      "type": "object",
      "required": ["credentialSubject", "type", "credentialPolicy"],
      "properties": {
        "type": {
          "description": "Credential types including VerifiableMandate.",
          "type": "array",
          "items": { "type": "string" },
          "contains": { "const": "VerifiableMandate" }
        },
        "credentialSubject": {
          "type": "object",
          "required": ["id", "delegatee", "roles", "grants"],
          "properties": {
            "id": {
              "type": "string",
              "description": "DID of the delegator (credential subject)."
            },
            "delegatee": {
              "description": "DID or object representing the delegated entity.",
              "oneOf": [
                { "type": "string" },
                { "type": "object" }
              ]
            },
            "roles": {
              "type": "array",
              "items": { "type": "string" },
              "minItems": 1,
              "description": "Roles assigned to the delegatee."
            },
            "grants": {
              "type": "array",
              "items": { "type": "string" },
              "minItems": 1,
              "description": "Delegated actions/permissions."
            },
            "constraint": {
              "type": "object",
              "description": "Optional constraints with flexible structure.",
              "additionalProperties": true
            }
          }
        },
        "credentialPolicy": {
          "description": "Reference to the policy that defines how this credential should be validated (e.g., Rego).",
          "type": "object",
          "required": ["id", "type"],
          "properties": {
            "id": {
              "type": "string",
              "format": "uri",
              "description": "URI of the policy file."
            },
            "type": {
              "type": "string",
              "description": "Type of policy language or format. Recommended: 'RegoPolicy2024'."
            },
            "description": {
              "type": "string",
              "description": "Optional description of the policy."
            }
          }
        }
      }
    }
  ]
}
```

Appendix B. Verifiable Mandate (Business-Oriented) Schema

The following JSON Schema (Listing A2) describes a specialized version of the *Verifiable Mandate*, tailored to a postal use case. It builds upon the general VM schema, requiring location as a mandatory constraint and supporting optional fields like `'validFrom'`, `'validUntil'`, and `'registeredMailTrackingId'`.

Listing A2. Verifiable Mandate (Business-Oriented) JSON Schema.

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "$id": "https://your-postoffice.org/schemas/verifiable-mandate-postal.json",
  "title": "Verifiable Mandate (Postal Delegation)",
  "description": "Verifiable Mandate for postal delegation with required location and optional mail-specific constraints.",
  "allOf": [
    { "$ref": "https://your-university.edu/schemas/verifiable-mandate-general.json" },
    {
      "type": "object",
      "properties": {
        "credentialSubject": {
          "type": "object",
          "properties": {
            "constraint": {
              "type": "object",
              "required": ["location"],
              "properties": {
                "location": {
                  "type": "string",
                  "description": "Required location constraint, e.g., PostOffice_SI_MB12"
                },
                "validFrom": {
                  "type": "string",
                  "format": "date",
                  "description": "Optional start of the allowed validity window."
                },
                "validUntil": {
                  "type": "string",
                  "format": "date",
                  "description": "Optional end of the allowed validity window."
                },
                "registeredMailTrackingId": {
                  "type": "string",
                  "description": "Optional tracking ID if registered mail is delegated."
                }
              },
              "additionalProperties": true,
              "description": "Constraint object with required and optional fields specific to the postal domain."
            }
          }
        }
      }
    }
  ]
}
```

Appendix C. Abbreviations and Symbols

Alphabetically ordered abbreviations (Table A1) and symbols (Table A2) used throughout the paper.

Table A1. Abbreviations used throughout the paper.

Abbreviation	Description
ABAC	Attribute-Based Access Control
ABDM	Ability-Based OR Attribute-Based Delegation Model
AC	Access Control
ACERT	Attribute Certificate
AD	Access or Authority Delegation
API	Application Programming Interface
CC	Chained Credentials

Table A1. *Cont.*

Abbreviation	Description
CEL	Common Expression Language
DAC	Discretionary Access Control
DID	Decentralized Identifier
DID-comm	DID Communication
DIF	Decentralized Identity Foundation
DPKI	Decentralized Public Key Infrastructure
DSL	Domain-Specific Language
EBP	European Blockchain Partnership
EBSI	European Blockchain Services Infrastructure
EC	European Commission
eIDAS	Electronic Identification and Trust Services
ESSIF	European Self Sovereign Identity Framework
EU	European Union
EUDIW	European Digital Identity Wallet
ICT	Information and Communications Technology
IdM	Identity Management
IIC	Indirect Identity Control
JSON	JavaScript Object Notation
JSON-LD	JSON for Linked Data
MAC	Mandatory Access Control
OAuth	Open Authorization
OCAP	Object Capability
OCAP-LD	Object-Capability for Linked Data
OIDC4VC	OpenID for Verifiable Credentials
OPA	Open Policy Agent
P2P	Peer-to-peer
PKCERT	Public Key Certificate
PoC	Proof-of-Concepts
RBAC	Role Based Access Control
Rego	Declarative policy language used by OPA
RFC	Request for Comments
SDM	Single Digital Market
SGL	Simple Grant Language
SSI	Self-Sovereign Identity
TAOR	Trusted Accreditation Organisation Registry
TIR	Trusted Issuers Registry
ToIP	Trust Over IP
TPR	Trusted Policy Registry (e.g., EBSI TSR)
TSR	Trusted Schema Registry (e.g., EBSI TSR)
URI	Uniform Resource Identifier
V.ID	Verifiable ID
VA	Verifiable Attestation
VAuthz	Verifiable Authorization Credential
VC	Verifiable Credential
VDR	Verifiable Data Registry
VM	Verifiable Mandate
VP	Verifiable Presentation
W3C	World Wide Web Consortium
ZCAP-LD	Authorization Capabilities for Linked Data

Table A2. Symbols used throughout the paper.

Symbol	Description
U	Set of users
R	Set of roles
G	Set of grants (permissions/actions)
C	Set of constraints (time, location, etc.)
UR	User-Role relation: $UR \subseteq U \times R$

Table A2. *Cont.*

Symbol	Description
RG	Role-Grant relation: $RG \subseteq R \times G$
GC	Grant-Constraint relation: $GC \subseteq G \times C$
$DPolicy$	Delegation Policy defined over (R, G, C)
$BProcess$	Business Process defined by grants and constraints
VM_i	Verifiable Mandate

References

1. Preukschat, A.; Reed, D. *Self-Sovereign Identity: Decentralized Digital Identity and Verifiable Credentials*; Manning: Shelter Island, NY, USA , 2021.
2. Čučko, Š.; Turkanović, M. Decentralized and Self-Sovereign Identity: Systematic Mapping Study. *IEEE Access* **2021**, *9*, 139009–139027. [CrossRef]
3. 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]
4. European Commission. Commission Proposes a Trusted and Secure Digital Identity for All Europeans. Available online: https://ec.europa.eu/commission/presscorner/detail/en/ip_21_2663 (accessed on 5 April 2022).
5. European Blockchain Services Infrastructure (EBSI). EBSI Documentation. Available online: https://ec.europa.eu/digital-building-blocks/sites/spaces/EBSI/overview (accessed on 5 April 2022).
6. eSSIF-Lab. European Self-Sovereign Identity Framework. Available online: https://essif-lab.eu (accessed on 5 April 2022).
7. Čučko, S.; Keršic, V.; Turkanović, M. Towards a Catalogue of Self-Sovereign Identity Design Patterns. *Appl. Sci.* **2023**, *13*, 5395. [CrossRef]
8. Čučko, Š.; Turkanović, M. A Novel Model for Authority and Access Delegation Utilizing Self-Sovereign Identity and Verifiable Credentials. *IEEE Access* **2025**, *13*, 115102–115134. [CrossRef]
9. 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]
10. 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]
11. Sporny, M.; Longley, D.; Sabadello, M.; Reed, D.; Steele, O.; Allen, C. Decentralized Identifiers (DIDs) v1.0. 2021. Available online: https://www.w3.org/TR/did-core/ (accessed on 4 April 2022).
12. Sporny, M.; Longley, D.; Chadwick, D. Verifiable Credentials Data Model 1.0. November 2019. Available online: https://www.w3.org/TR/vc-data-model/ (accessed on 21 October 2021).
13. Wagner, K.; Némethi, B.; Renieris, E.; Lang, P.; Brunet, E.; Holst, E. *Self-Sovereign Identity: A Position Paper on Blockchain Enabled Identity and the Road Ahead*; Blockchain Bundesverband: Berlin, Germany , 2018.
14. Steele, O.; Sporny, M. DID Specification Registries. 2021. Available online: https://www.w3.org/TR/2021/NOTE-did-spec-registries-20210409/ (accessed on 4 April 2022).
15. Sandhu, R.; Bhamidipati, V.; Munawer, Q. The ARBAC97 Model for Role-Based Administration of Roles. *ACM Trans. Inf. Syst. Secur.* **1999**, *2*, 105–135. [CrossRef]
16. Hu, V.C.; Ferraiolo, D.; Kuhn, R.; Schnitzer, A.; Sandlin, K.; Miller, R.; Scarfone, K. *Guide to Attribute Based Access Control (ABAC) Definition and Considerations*; National Institute of Standards and Technology Special Publication: Gaithersburg, MD, USA, 2014. [CrossRef]
17. Hardman, D. Aries RFC 0103: Indirect Identity Control. October 2019. Available online: https://github.com/hyperledger/aries-rfcs/tree/main/concepts/0103-indirect-identity-control (accessed on 21 October 2021).
18. Chadwick, D. Verifiable Credentials Lifecycle 1.0. December 2020. Available online: https://w3c-ccg.github.io/vc-lifecycle/#delegation-of-authority (accessed on 30 March 2022).
19. Ali, A.; Habiba, U.; Shibli, M.A. Taxonomy of Delegation Model. In Proceedings of the 2015 12th International Conference on Information Technology—New Generations, Las Vegas, NV, USA, 13–15 April 2015; pp. 218–223. [CrossRef]
20. Fedrecheski, G.; Rabaey, J.M.; Costa, L.C.P.; Calcina Ccori, P.C.; Pereira, W.T.; Zuffo, M.K. Self-Sovereign Identity for IoT environments: A Perspective. In Proceedings of the 2020 Global Internet of Things Summit (GIoTS), Dublin, Ireland, 3 June 2020; pp. 1–6. [CrossRef]
21. Kozen, D. Language-Based Security. In *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*; Springer: Berlin/Heidelberg, Germany, 1999; Volume 1672, pp. 284–298. [CrossRef]

22. Bhargavan, K.; Fournet, C.; Gordon, A.D. Verifying policy-based security for web services. In Proceedings of the ACM Conference on Computer and Communications Security, Washington, DC, USA, 25–29 October 2004; pp. 268–277. [CrossRef]
23. Varadharajan, V.; Karmakar, K.; Tupakula, U.; Hitchens, M. A policy-based security architecture for software-defined networks. *IEEE Trans. Inf. Forensics Secur.* **2019**, *14*, 897–912. [CrossRef]
24. She, W.; Thuraisingham, B.; Yen, I.L. Delegation-Based Security Model for Web Services. In Proceedings of the 10th IEEE High Assurance Systems Engineering Symposium (HASE'07), Plano, TX, USA, 14–16 November 2007; pp. 82–91. [CrossRef]
25. Hassan, D.; Mousavi, M.R.; Reniers, M.A. Restricted delegation and revocation in language-based security: (Position paper). In Proceedings of the 5th ACM SIGPLAN Workshop on Programming Languages and Analysis for Security, Toronto, ON, Canada, 10 June 2010; pp. 1–7. [CrossRef]
26. Hardman, D.; Harchandani, L. Aries RFC 0104: Chained Credentials. November 2019. Available online: https://github.com/hyperledger/aries-rfcs/blob/main/concepts/0104-chained-credentials/README.md (accessed on 21 October 2021).
27. Cloud Native Computing Foundation. Open Policy Agent 2025. Available online: https://www.openpolicyagent.org (accessed on 8 April 2022).
28. Google. Google/cel-spec: Common Expression Language—Specification and Binary Representation. Available online: https://github.com/google/cel-spec (accessed on 8 April 2022).
29. Hu, V.C.; Ferraiolo, D., F.; Kuhn, D.R. *Assessment of Access Control Systems*; National Institute of Standards and Technology Special Publication: Gaithersburg, MD, USA , 2006. [CrossRef]
30. Turkanović, M.; Podgorelec, B. Signing Blockchain Transactions Using Qualified Certificates. *IEEE Internet Comput.* **2020**, *24*, 37–43. [CrossRef]
31. Xiang, H.; Wan, Z.; Liu, L. A Three-Dimensional Role-Based Delegation Model. In Proceedings of the 2009 International Conference on Information Engineering and Computer Science, Wuhan, China, 19–20 December 2009; pp. 1–4. [CrossRef]
32. Ye, C.; Fu, Y.; Wu, Z. An Attribute-Based-Delegation-Model. In Proceedings of the 3rd International Conference on Information Security (InfoSecu '04), Shanghai, China, 14–16 November 2004; Association for Computing Machinery: New York, NY, USA, 2004; pp. 220–221. [CrossRef]
33. Zhang, X.; Oh, S.; Sandhu, R. PBDM: A Flexible Delegation Model in RBAC. In Proceedings of the Eighth ACM Symposium on Access Control Models and Technologies (SACMAT '03), Como, Italy, 2–3 June 2003; Association for Computing Machinery: New York, NY, USA, 2003; pp. 149–157. [CrossRef]
34. Di Francesco Maesa, D.; Mori, P.; Ricci, L. Blockchain Based Access Control. In Proceedings of the Distributed Applications and Interoperable Systems, Neuchâtel, Switzerland, 19–22 June 2017; Chen, L.Y., Reiser, H.P., Eds.; Springer International Publishing: Cham, Switzerland, 2017; pp. 206–220. [CrossRef]
35. Xu, H.; He, Q.; Li, X.; Jiang, B.; Qin, K. BDSS-FA: A Blockchain-Based Data Security Sharing Platform With Fine-Grained Access Control. *IEEE Access* **2020**, *8*, 87552–87561. [CrossRef]
36. Pinno, O.J.A.; Gregio, A.R.A.; De Bona, L.C.E. ControlChain: Blockchain as a Central Enabler for Access Control Authorizations in the IoT. In Proceedings of the GLOBECOM 2017—2017 IEEE Global Communications Conference, Singapore, 4–8 December 2017; pp. 1–6. [CrossRef]
37. Ouaddah, A.; Abou Elkalam, A.; Ait Ouahman, A. FairAccess: A new Blockchain-based access control framework for the Internet of Things. *Secur. Commun. Netw.* **2016**, *9*, 5943–5964. [CrossRef]
38. Ali, G.; Ahmad, N.; Cao, Y.; Asif, M.; Cruickshank, H.; Ali, Q.E. Blockchain based permission delegation and access control in Internet of Things (BACI). *Comput. Secur.* **2019**, *86*, 318–334. [CrossRef]
39. Schardong, F.; Custódio, R. Self-Sovereign Identity: A Systematic Review, Mapping and Taxonomy. *Sensors* **2022**, 22, 5641. [CrossRef] [PubMed]
40. Lim, S.; Rhie, M.H.; Hwang, D.; Kim, K.H. A Subject-Centric Credential Management Method based on the Verifiable Credentials. In Proceedings of the 2021 International Conference on Information Networking (ICOIN), Jeju Island, South Korea, 13–16 January 2021; pp. 508–510. [CrossRef]
41. Mahalle, P.N.; Shinde, G.R. OAuth-Based Authorization and Delegation in Smart Home for the Elderly Using Decentralized Identifiers and Verifiable Credentials. In *Security Issues and Privacy Threats in Smart Ubiquitous Computing*; Mahalle, P.N., Shinde, G.R., Dey, N., Hassanien, A.E., Eds.; Springer: Singapore, 2021; pp. 95–109. [CrossRef]
42. Sovrin. Guardianship. 2021. Available online: https://sovrin.org/guardianship/ (accessed on 21 October 2021).
43. Sovrin Guardianship Task Force. On Guardianship in Self-Sovereign Identity. December 2019. Available online: https://sovrin.org/wp-content/uploads/Guardianship-Whitepaper2.pdf (accessed on 21 October 2021).
44. Davie, M.; Gisolfi, D.; Hardman, D.; Jordan, J.; O'Donnell, D.; Reed, D.; van Deventer, O. 0289: The Trust over IP Stack. 2019. Available online: https://github.com/hyperledger/aries-rfcs/tree/main/concepts/0289-toip-stack (accessed on 21 October 2021).
45. Hardman, D. Sample Guardianship Trust Framework. September 2019. Available online: https://github.com/decentralized-identity/aries-rfcs/blob/main/concepts/0103-indirect-identity-control/guardianship-sample/trust-framework.md (accessed on 21 October 2021).

46. Hardman, D. Sample Guardianship Schema. September 2019. Available online: https://github.com/decentralized-identity/aries-rfcs/blob/main/concepts/0103-indirect-identity-control/guardianship-sample/schema.md (accessed on 21 October 2021).
47. Dündar, Y.; Sertkaya, I. Self Sovereign Identity based mutual guardianship. *J. Mod. Technol. Eng.* **2020**, *5*, 189–211.
48. Bartolomeu, P.C.; Vieira, E.; Hosseini, S.M.; Ferreira, J. Self-Sovereign Identity: Use-cases, Technologies, and Challenges for Industrial IoT. In Proceedings of the 2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Zaragoza, Spain, 10–13 September 2019; pp. 1173–1180. [CrossRef]
49. Mukta, R.; Paik, H.Y.; Lu, Q.; Kanhere, S.S. CredTrust: Credential Based Issuer Management for Trust in Self-Sovereign Identity. In Proceedings of the 2022 IEEE International Conference on Blockchain (Blockchain), Espoo, Finland, 22–25 August 2022; pp. 334–339. [CrossRef]
50. Chadwick, D.; Longley, D.; Sporny, M.; Terbu, O.; Zagidulin, D.; Zundel, B. Verifiable Credentials Implementation Guidelines 1.0. February 2023. Available online: https://w3c.github.io/vc-imp-guide/#subject-holder-relationships (accessed on 5 September 2025).

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.