

OGC Testbed-15
Federated Cloud Provenance ER

Table of Contents

1. Subject	4
2. Executive Summary	5
2.1. Document contributor contact points	6
2.2. Foreword	6
3. References	7
4. Terms and definitions	8
4.1. Abbreviated terms	13
5. Overview	15
6. Provenance	17
6.1. Provenance vocabulary standards	17
6.2. Levels of Provenance and Resource Sharing	20
6.3. Provenance Security	24
7. Workflows	26
7.1. Workflow Management Services	26
7.2. Workflow description standards	27
8. Blockchain Technologies	28
8.1. Governance in DLTs, Blockchain, Hybrid Blockchain	28
8.2. Permissioned vs. Permissionless and Private versus Public	28
8.3. Hashing	29
8.4. Consensus	29
8.5. IPFS	29
8.6. BitTorrent File System (BTFS)	30
8.7. Hyperledger Fabric	31
8.8. Smart Contracts	32
9. Identity Management	34
9.1. Evolution of Online Identity Model	34
9.1.1. Centralized Identity	35
9.1.2. Federated Identity	35
9.1.3. User-Centric Identity	35
9.1.4. Self-Sovereign Identity	36
9.2. Centralized Public key infrastructure (PKI)	37
9.2.1. Digital Certificates	38
9.2.2. Authentication, Public keys, and Private Keys	39
9.2.3. Certificate Authorities	40
9.2.4. Certificate Revocation Lists	41
9.2.5. Problems with Centralized PKIs	41
9.3. Decentralized Public Key Infrastructure (DPKI)	44
9.3.1. Decentralized Identifiers	45

9.3.2. Verifiable Credentials	50
9.3.3. Verifiable Credentials Ecosystem	50
9.3.4. DPKI solutions	57
10. Solutions	58
10.1. Challenges	58
10.1.1. DLT and Blockchain heterogeneity	58
10.1.2. Location-based Addressing	59
10.1.3. Lack of Standard Provenance for Workflow	59
10.2. Siloed Trust System	60
10.3. High level description of the solution	60
10.4. Identity Management for Federated Cloud	61
10.5. Universal Resolver for Self-Sovereign Identifiers	62
10.6. Chaining DADs	63
10.7. Content Availability	64
10.8. Verifiable Credentials for Federated Cloud Analytics	64
10.9. What is stored in the Blockchain ?	65
11. Conclusions	66
11.1. Future Works	66
Appendix A: Revision History	68
Appendix B: Bibliography	69

Publication Date: 2020-02-12

Approval Date: 2020-02-12

Submission Date: 2019-12-20

Reference number of this document: OGC 19-015

Reference URL for this document: <http://www.opengis.net/doc/PER/t15-D022>

Category: OGC Public Engineering Report

Editor: Stephane Fellah

Title: OGC Testbed-15: Federated Cloud Provenance ER

OGC Public Engineering Report

COPYRIGHT

Copyright © 2020 Open Geospatial Consortium. To obtain additional rights of use, visit <http://www.opengeospatial.org/>

WARNING

This document is not an OGC Standard. This document is an OGC Public Engineering Report created as a deliverable in an OGC Interoperability Initiative and is not an official position of the OGC membership. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an OGC Standard. Further, any OGC Public Engineering Report should not be referenced as required or mandatory technology in procurements. However, the discussions in this document could very well lead to the definition of an OGC Standard.

LICENSE AGREEMENT

Permission is hereby granted by the Open Geospatial Consortium, ("Licensor"), free of charge and subject to the terms set forth below, to any person obtaining a copy of this Intellectual Property and any associated documentation, to deal in the Intellectual Property without restriction (except as set forth below), including without limitation the rights to implement, use, copy, modify, merge, publish, distribute, and/or sublicense copies of the Intellectual Property, and to permit persons to whom the Intellectual Property is furnished to do so, provided that all copyright notices on the intellectual property are retained intact and that each person to whom the Intellectual Property is furnished agrees to the terms of this Agreement.

If you modify the Intellectual Property, all copies of the modified Intellectual Property must include, in addition to the above copyright notice, a notice that the Intellectual Property includes modifications that have not been approved or adopted by LICENSOR.

THIS LICENSE IS A COPYRIGHT LICENSE ONLY, AND DOES NOT CONVEY ANY RIGHTS UNDER ANY PATENTS THAT MAY BE IN FORCE ANYWHERE IN THE WORLD. THE INTELLECTUAL PROPERTY IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE DO NOT WARRANT THAT THE FUNCTIONS CONTAINED IN THE INTELLECTUAL PROPERTY WILL MEET YOUR REQUIREMENTS OR THAT THE OPERATION OF THE INTELLECTUAL PROPERTY WILL BE UNINTERRUPTED OR ERROR FREE. ANY USE OF THE INTELLECTUAL PROPERTY SHALL BE MADE ENTIRELY AT THE USER'S OWN RISK. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR ANY CONTRIBUTOR OF INTELLECTUAL PROPERTY RIGHTS TO THE INTELLECTUAL PROPERTY BE LIABLE FOR ANY CLAIM, OR ANY DIRECT, SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM ANY ALLEGED INFRINGEMENT OR ANY LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR UNDER ANY OTHER LEGAL THEORY, ARISING OUT OF OR IN CONNECTION WITH THE IMPLEMENTATION, USE, COMMERCIALIZATION OR PERFORMANCE OF THIS INTELLECTUAL PROPERTY.

This license is effective until terminated. You may terminate it at any time by destroying the Intellectual Property together with all copies in any form. The license will also terminate if you fail to comply with any term or condition of this Agreement. Except as provided in the following sentence, no such termination of this license shall require the termination of any third party end-user sublicense to the Intellectual Property which is in force as of the date of notice of such termination. In addition, should the Intellectual Property, or the operation of the Intellectual Property, infringe, or in LICENSOR's sole opinion be likely to infringe, any patent, copyright, trademark or other right of a third party, you agree that LICENSOR, in its sole discretion, may terminate this license without any compensation or liability to you, your licensees or any other party. You agree upon termination of any kind to destroy or cause to be destroyed the Intellectual Property together with all copies in any form, whether held by you or by any third party.

Except as contained in this notice, the name of LICENSOR or of any other holder of a copyright in all or part of the Intellectual Property shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Intellectual Property without prior written authorization of LICENSOR or such copyright holder. LICENSOR is and shall at all times be the sole entity that may authorize you or any third party to use certification marks, trademarks or other special designations to

indicate compliance with any LICENSOR standards or specifications.

This Agreement is governed by the laws of the Commonwealth of Massachusetts. The application to this Agreement of the United Nations Convention on Contracts for the International Sale of Goods is hereby expressly excluded. In the event any provision of this Agreement shall be deemed unenforceable, void or invalid, such provision shall be modified so as to make it valid and enforceable, and as so modified the entire Agreement shall remain in full force and effect. No decision, action or inaction by LICENSOR shall be construed to be a waiver of any rights or remedies available to it.

None of the Intellectual Property or underlying information or technology may be downloaded or otherwise exported or reexported in violation of U.S. export laws and regulations. In addition, you are responsible for complying with any local laws in your jurisdiction which may impact your right to import, export or use the Intellectual Property, and you represent that you have complied with any regulations or registration procedures required by applicable law to make this license enforceable.

Chapter 1. Subject

The emergence of Federated Cloud processing and ‘Big Data’ have raised many concerns over the use to which data is being put. This led to new requirements for methodologies, and capabilities which can address transparency and trust in data provenance in the Cloud. Distributed Ledger Technologies (DLTs) and more specifically blockchains, have been proposed as a possible platform to address provenance. This OGC Testbed 15 Engineering Report (ER) is a study of the application of DLTs for managing provenance information in Federated Clouds.

Chapter 2. Executive Summary

Cloud computing has been widely adopted by the commercial, research and military communities. To support computing "on-demand" and "pay-as-you-go" models, cloud computing extends distributed and parallel system architecture by using abstraction and virtualization techniques. These environments are composed of heterogeneous hardware and software components from different vendors on which complex workflows can be executed using a federated orchestration of the execution of these workflows.

Assurance of the quality and repeatability of data results is essential in many fields (eScience, and healthcare for example) and requires cloud auditing and the maintenance of provenance information for the whole workflow execution. The use of heterogeneous components in cloud computing environment introduces the risks of accidental data corruption, processing errors, vulnerabilities such as security violation, data tampering or malicious forgery of provenance. Cloud systems are structured in a fundamentally different way from other distributed systems, such as grids, and therefore present new challenges for the collection of provenance data.

Current scientific workflows do not provide a standard way to share provenance. Existing workflow management systems integrating provenance repositories are typically proprietary and are not interoperable with other systems in mind. Federated Cloud Architectures exacerbate the challenge of tracking and sharing provenance information.

The sharing of provenance from scientific workflows would enable the rapid reproduction of results and enable the rapid computing of new and significant results using the history to generate new workflow definition using minor modifications to the original workflow. The ability to share provenance will greatly reduce duplication of workflows, improve the trust and integrity of data and analyses, improve reproducibility of scientific workflows and catalyze the discovery of new knowledge. While these may be relatively simple to achieve in a single, well-designed workflow management system that captures provenance, there is no readily available general-purpose solution, especially for cloud-based environment.

The scope of this study is to review the state-of-the-art of Provenance and Blockchain technologies, identify the challenges and requirements about using cloud computing provenance on a blockchain. Based on these analyses, the authors of this ER propose an architecture to share provenance information from federated cloud workflows that ensure the provenance information has not be tampered with so that user can trust the results produced by the workflow. This study is not about defining a model of provenance to reproduce workflows, though the authors will indicate some good candidates to address that challenge.

The findings of the study determine that W3C Self Sovereign Identifiers (SSIs) and Verifiable credentials are fundamental assets for interaction over the Internet and are the cornerstone of establishing the Web Of Trust needed to ensure provenance of information. SSI brings back full control of the identity to the owner and the use of DLTs and Blockchain to support Decentralized PKI provides a solid alternative that addresses the usability and security issues of the centralized PKI approach. SSIs and Verifiable credentials are still young technologies, but the development of these standards is moving at rapid pace, and will have profound impact on the current web technologies, by creating Web 4.0.

2.1. Document contributor contact points

All questions regarding this document should be directed to the editor or the contributors:

Contacts

Name	Organization	Role
Stephane Fella	Image Matters LLC	Editor
Anna Burzykowska	European Space Agency	Contributor

2.2. Foreword

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The Open Geospatial Consortium shall not be held responsible for identifying any or all such patent rights.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.

Chapter 3. References

The following normative documents are referenced in this document.

- [OGC 06-121r9, OGC® Web Services Common Standard](https://portal.opengeospatial.org/files/?artifact_id=38867&version=2) [https://portal.opengeospatial.org/files/?artifact_id=38867&version=2]

Chapter 4. Terms and definitions

For the purposes of this report, the definitions specified in Clause 4 of the OWS Common Implementation Standard [OGC 06-121r9](https://portal.opengeospatial.org/files/?artifact_id=38867&version=2) [https://portal.opengeospatial.org/files/?artifact_id=38867&version=2] shall apply. In addition, the following terms and definitions apply.

- **51% Attack**

When more than half of the computing power of a cryptocurrency network is controlled by a single entity or group, this entity or group may issue conflicting transactions to harm the network, should they have the malicious intent to do so.

- **Address**

Addresses (Cryptocurrency addresses) are used to receive and send transactions on the network. An address is a string of alphanumeric characters, but can also be represented as a scannable QR code.

- **Blockchain**

A particular type of data structure used in some distributed ledgers, which stores and transmits data in packages called 'blocks', connected to each other in a digital 'chain'. Blockchains employ cryptographic and algorithmic methods to record and synchronise data across a network in an immutable manner.

- **Blockchain transaction**

A Blockchain transaction can be defined as a small unit of task, stored in public records. These records are also known as 'blocks'. These blocks are executed, implemented and stored in blockchain only after validation by the entities in the blockchain network.

- **Central Ledger**

A central ledger refers to a ledger maintained by a central agency.

- **Claim**

A statement about an identity. This could be: a fact, such as a person's age; an opinion, such as a rating of their trustworthiness; or something in between, such as an assessment of a skill.

- **Confirmation**

A confirmation means that the blockchain transaction has been verified by the network. This happens through a process known as mining, in a proof-of-work system (e.g. Bitcoin). Once a transaction is confirmed, it cannot be reversed or double spent. The more confirmations a transaction has, the harder it becomes to perform a double spend attack.

- **Consensus**

Consensus is achieved when all participants of the network agree on the validity of the transactions, ensuring that the ledgers are exact copies of each other.

- **Credential**

A set of one or more claims made by an issuer. A verifiable credential is a tamper-evident credential that has authorship that can be cryptographically verified. Verifiable credentials can

be used to build verifiable presentations, which can also be cryptographically verified. The claims in a credential can be about different subjects.

- **Cryptocurrency**

A form of digital currency based on mathematics, where encryption techniques are used to regulate the generation of units of currency and verify the transfer of funds. Furthermore, cryptocurrencies operate independently of a central bank.

- **Cryptography**

A method for securing communication using code. The main example of cryptography in cryptocurrency is the symmetric-key cryptography used in the Bitcoin network. Bitcoin addresses generated for the wallet have matching private keys that allow for the spending of the cryptocurrency. The corresponding public key coupled with the private key allows funds to be unlocked. This is one example of cryptography in action.

- **Decentralization**

The transfer of authority and responsibility from a centralized organization, government, or party to a distributed network.

- **Decentralized Application (DApp)**

DApp is a decentralized application, running on a decentralized peer-to-peer network as opposed to running on centralized servers.

- **Decentralized Identifier (DID)**

A globally unique identifier that does not require a centralized registration authority because it is registered with distributed ledger technology (DLT) or other form of decentralized network.

- **Digital currencies**

Digital currencies are digital representations of value, denominated in their own unit of account. They are distinct from e-money, which is a digital payment mechanism, representing and denominated in fiat money.

- **Digital Identity**

A digital identity is an online or networked identity adopted or claimed in cyberspace by an individual, organization, or electronic device.

- **Distributed Ledger Technology (DLT)**

DLT refers to a novel and fast-evolving approach to recording and sharing data across multiple data stores (or ledgers). This technology allows for transactions and data to be recorded, shared, and synchronized across a distributed network of different network participants.

- **Ethereum**

Ethereum is the open-source, public, blockchain-based distributed computing platform and operating system, featuring smart contract functionality.

- **EVM**

The Ethereum Virtual Machine (EVM) is a Turing complete virtual machine that allows anyone to execute arbitrary EVM Byte Code. Every Ethereum node runs on the EVM to maintain

consensus across the blockchain.

- **Fork**

A fork creates an alternative version of a blockchain. The two chains run simultaneously on different parts of the network. They can be either accidental or intentional.

- **Genesis Block**

The very first block in a block chain.

- **Identity Provider**

An identity provider, sometimes abbreviated as IdP, is a system for creating, maintaining, and managing identity information for holders, while providing authentication services to relying party applications within a federation or distributed network. In this case the holder is always the subject. Even if the verifiable credentials are bearer credentials, it is assumed the verifiable credentials remain with the subject, and if they are not, they were stolen by an attacker. This specification does not use this term unless comparing or mapping the concepts in this document to other specifications. This specification decouples the identity provider concept into two distinct concepts: the issuer and the holder.

- **Immutable**

An inability to be altered or changed over time. This refers to a ledger's inability to be changed by a single administrator, all data once written onto a blockchain can be altered.

- **Internet of Things (IoT)**

Internet of Things is a network of objects, linked by a tag or microchip, that send data to a system that receives it.

- **InterPlanetary File System (IPFS)**

Distribution protocol that started as an open source project at Interplanetary Networks. The p2p method of storing and sharing hypermedia in a distributed file system aims to help applications run faster, safer and more transparently. IPFS allows objects to be exchanged and interact without a single point of failure. IPFS creates trustless node interrelations.

- **Ledger**

An append-only record store, where records are immutable and may hold more general information than financial records.

- **Mining**

Mining is the act of validating blockchain transactions. The necessity of validation warrants an incentive for the miners, usually in the form of coins.

- **Multi Signature**

Multi-signature (multisig) addresses allow multiple parties to require more than one key to authorize a transaction. The needed number of signatures is agreed at the creation of the address. Multi signature addresses have a much greater resistance to theft.

- **Node (Full Node)**

A computer connected to the blockchain network is referred to as a 'node'. Most nodes are not

full nodes and full nodes can be difficult to run due to their bulky size. A full node is a program can fully validate transactions and blocks bolstering the p2p network.

- **Oracle**

An oracle helps communicate data using smart contracts connecting the real world and blockchain. The oracle finds and verifies events and gives this information to the smart contract on the blockchain.

- **Participant**

An actor who can access the ledger: read records or add records to.

- **Peer**

An actor that shares responsibility for maintaining the identity and integrity of the ledger.

- **Peer to Peer (P2P)**

Peer-to-peer (P2P) refers to the decentralized interactions that happen between at least two parties in a highly interconnected network. P2P participants deal directly with each other through a single mediation point.

- **Permissioned Ledger**

A permissioned ledger is a ledger where actors must have permission to access the ledger. Permissioned ledgers may have one or many owners. When a new record is added, the ledger's integrity is checked by a limited consensus process. This is carried out by trusted actors — government departments or banks, for example — which makes maintaining a shared record much simpler than the consensus process used by unpermissioned ledgers. Permissioned block chains provide highly-verifiable data sets because the consensus process creates a digital signature, which can be seen by all parties. A permissioned ledger is usually faster than an unpermissioned ledger.

- **PoS/Pow Hybrid**

a combination of Proof of Stake (PoS) and Proof of Work (PoW) consensus protocols on a blockchain network. Blocks are validated from not only miners, but also voters (stakeholders) to form a balanced network governance.

- **Private Blockchain**

A closed network where blockchain permissions are held and controlled by a centralized entity. Read permissions are subject to varying levels of restriction.

- **Public Address**

A public address is the cryptographic hash of a public key. They act as email addresses that can be published anywhere, unlike private keys.

- **Private Key**

A private key is a string of data that allows you to access the tokens in a specific wallet. They act as passwords that are kept hidden from anyone but the owner of the address.

- **Proof-of-Authority**

A consensus mechanism in a private blockchain that grants a single private key the authority to

generate all of the blocks.

- **Proof of Stake**

A consensus distribution algorithm that rewards earnings based on the number of coins you own or hold. The more you invest in the coin, the more you gain by mining with this protocol.

- **Proof of Work**

A consensus distribution algorithm that requires an active role in mining data blocks, often consuming resources, such as electricity. The more ‘work’ you do or the more computational power you provide, the more coins you are rewarded with.

- **Protocol**

A set of rules that dictate how data is exchanged and transmitted. This pertains to cryptocurrency in blockchain when referring to the formal rules that outline how these actions are performed across a specific network.

- **Public Blockchain**

A globally public network where anyone participate in transactions, execute consensus protocol to help determine which blocks get added to the chain, and maintain the shared ledger.

- **Public Key Cryptography**

Public Key Cryptography is an asymmetric encryption scheme that uses two sets of keys: a public key that is widely disseminated, and a private key known only to the owner. Public key cryptography can be used to create digital signatures, and is used in a wide array of applications, such as HTTPS internet protocol, for authentication in critical applications, and also in chip-based payment cards.

- **SHA-256**

SHA-256 is a cryptographic algorithm used by cryptocurrencies such as Bitcoin. However, it uses a lot of computing power and processing time, forcing miners to form mining pools to capture gains.

- **Smart contract**

Smart contract is a computer protocol intended to digitally facilitate, verify, or enforce the negotiation or performance of a contract. In this paper ‘smart contract’ is mostly used in the sense of general purpose computation that takes place on a blockchain or distributed ledger. In this interpretation, a ‘smart contract’ is not necessarily related to the classical concept of a contract, but can be any kind of computer program or code-executed task on blockchain.

- **Solidity**

Solidity is Ethereum’s programming language for developing smart contracts.

- **Token**

A Token is a representation of a digital asset. It typically does not have intrinsic value but is linked to an underlying asset, which could be anything of value.

- **Unpermissioned ledgers**

Unpermissioned ledgers such as Bitcoin have no single owner — indeed, they cannot be owned.

The purpose of an unpermissioned ledger is to allow anyone to contribute data to the ledger and for everyone in possession of the ledger to have identical copies. This creates censorship resistance, which means that no actor can prevent a transaction from being added to the ledger. Participants maintain the integrity of the ledger by reaching a consensus about its state

- **Verifiable Claim**

A Verifiable Claim is machine-readable information that can be verified by a third party on the Web. Such a claim is effectively tamper-proof and its authorship can be cryptographically verified. Multiple claims may be bundled together into a set of claims

4.1. Abbreviated terms

- API Application Programming Interface
- CAS Content Addressable Storage
- DAO Decentralized Autonomous Organization
- DID Decentralized Identifier
- DIF Decentralized Identity Foundation
- DPKI Decentralized Public Key Infrastructure
- DLT Distributed Ledger Technology
- DNS Domain Name System
- ERC Ethereum Request for Comments
- ETH Ethereum
- FIM Federated Identity Management
- HTTP Hyper-Text Transfer Protocol
- IDMS Identity Management System
- IP Internet Protocol
- IPFS Inter-Planetary File System
- ISO International Organization for Standardization
- JSON JavaScript Object Notation
- JSON-LD JavaScript Object Notation for Linked Data
- OGC Open Geospatial Consortium
- PII Personally-Identifiable Information
- SSI Self-Sovereign Identity
- URI Uniform Resource Identifier
- URL Uniform Resource Locator
- W3C World Wide Web Consortium
- ZK Zero-Knowledge
- ZKP Zero-Knowledge Protocol

- W3C World Wide Web Consortium
- XML eXtensible Markup Language

Chapter 5. Overview

Until recently, data analysts designed their algorithms with the assumption that the analyzed data were gathered into a centralized repository, such as in a cloud data center or a data lake (a storage repository that holds a vast amount of raw data in its native format until it is needed). A paradigm shift is now occurring with the exponential growth of Big Data due to the rise of the Internet of Things (IoT), social media, mobility, and other data sources. This growth defies the scalability of centralized approaches to store and analyze data in a single location. For example, some sensors generate and store data locally, as moving the data to a centralized location makes it impractical due to bandwidth and cost constraints. In other cases, data centralization may not be possible due to security concerns for data in transit, governance, privacy, or regulatory compliance issues that limit the movement of data beyond certain geographic boundaries.

Analytics in a centralized repository is becoming challenging as data becomes more and more distributed. If we cannot bring data together for analysis, then analytics needs to be taken to the data. This occurs often in controlled, well-defined and well-secured places at the edge, in the fog or core, and in the cloud or enterprise data centers. Further, intermediate results may need to be fused and analyzed together as well. This is where federated cloud and federated analytics enters the picture. There are a number of challenges to overcome:

- How to redesign the data analytic algorithms to reason and learn in a federated manner?
- How to distribute the analytics close to where the data is collected?
- How to aggregate and analyze together the intermediate results to drive higher-order learning at scale?
- How to ensure the quality and repeatability of data results of distributed workflow in federated environment?
- How to audit the execution of distributed workflow in a federated environment to analyze the validity of analysis or identify faults in execution?
- How to verify the integrity of all the participants and all the data sources in the analytics process? In particular, how to get the same assurances of trust, transparency and traceability (the “Three Ts” of data analytics) that you would have in a centralized world?

How do you get there? One answer is to combine the unique capabilities of federated analytics and blockchain technology, which adds a distributed ledger to the federated analytics solution. Identity Management and Provenance information play a central role in the solution. This is the topic of the study documented in this OGC ER.

Section 6 and 7 provide background information and related works about provenance and workflow related technologies, in particular in a federated cloud environment.

Section 8 provides background information about distributed ledger technologies, including blockchain. This section also describes their characteristics and benefits.

Section 9 provides a detailed analysis of the management of identity, which is a fundamental asset of interaction and demonstrate how DLTs and blockchain technologies can solve many of the fundamental issues of usability and security currently encountered in existing identity management solutions.

Section 10 provides an outline of a solution that addresses the challenges described above.

Section 11 summarizes the conclusion of this study and outlines future works to be investigated.

Chapter 6. Provenance

6.1. Provenance vocabulary standards

Provenance provides vital information for evaluating quality and trustworthiness of information on the Web. Therefore semantically interchangeable provenance information must be accessible and there must be an agreement on where and how this information is to be located [1]. The mission statement of the W3C Provenance Incubator Group stated that the provenance of information is critical to making determinations about whether information is trusted, how to integrate diverse information sources, and how to give credit to originators when reusing information. Broadly defined, provenance encompasses the initial sources of information used as well as any entity and process involved in producing a result. In an open and inclusive environment such as the Web, users find information that is often contradictory or questionable. People make trust judgements based on provenance that may or may not be explicitly offered to them.

Provenance is a well-established term in the context of art or digital libraries. In these cases, provenance respectively refers to the documented history of an art object, or the documentation of processes in a digital object's life cycle [2]. The "e-science community" interest for provenance [3] is also growing, as it is considered as a crucial component of workflow systems [4] that can help scientists ensure reproducibility of their scientific analyses and processes. In law, the concept of provenance refers to the "chain of custody" or the paper trail of evidence. This concept logically extends to the documentation of the history of change of data in a knowledge system [5].

In the context of this study, the term **provenance** refers to information about entities, activities, and people involved in producing a piece of data or thing, which can be used to form assessments about its quality, reliability, or trustworthiness [6].

Around the year 2006, consensus began to emerge on the benefits of having a community-defined data model and uniform representation for "data provenance, process documentation, data derivation, and data annotation", as stated in [7].

There are many areas of research and development that have studied relevant aspects of provenance. They can be classified into the following broad categories [8],[4],[7],[9]:

- Interoperability for different provenance systems and tools to aid in the integration of provenance.
- Information management infrastructure to manage growing volume of provenance data
- Provenance analytics and visualization for mining and extracting knowledge from provenance data, which has been largely unexplored
- Data provenance security and inference control

A series of challenges for provenance were launched [7]. The first Provenance Challenge [10] was to test the hypothesis that heterogeneous systems (mostly in the e-science/cyberinfrastructure space), each individually capable of producing provenance data by observing the execution of data-intensive processes, could successfully exchange such provenance observations with each other, without loss of information. The Open Provenance Model (OPM) [7] was proposed as a common

data model for the experiment. The second challenge aimed at allowing disparate groups to gain a better understanding of the similarities, differences, core concepts, and common issues across systems. The third challenge aimed at exchanging provenance information encoded with OPM and providing additional profiles. The fourth challenge was to apply OPM to scenarios and demonstrate novel functionality that can only be achieved by the presence of an interoperable solution for provenance. Some of the approaches to address these challenges use Semantic Web technologies [11].

The notion of causal relationships, or dependencies, involving artifacts (e.g., data items), processes, and agents plays a central role in OPM. Using the OPM, one can assert that an artifact A was produced or consumed by a process P, such as “the orthoimage was produced by using a Digital Elevation Model M, and aerial image I and control points C using orthorectification algorithm P.” Here M, I, and C are artifacts, and P is a process. One can also assert a derivation dependency between two artifacts, A1 and A2, without mentioning any mediating process, i.e., “A2 was derived from A1.” Agents, including humans, software systems, etc., can be mentioned in OPM as process controllers. For example, “the orthorectification process was controlled by software S managed by agent X. OPM statements attempt to explain the existence of artifacts. Since such statements may reflect an incomplete view of the world, obtained from a specific perspective, the OPM adopts an open world assumption, whereby the statements are interpreted as correct but possibly incomplete knowledge: “A2 was derived from A1” asserts a certain derivation, but does not exclude that other, possibly unknown artifacts, in addition to A1, may have contributed to explaining the existence of A2. Other features of the OPM, including built-in rules for inference of new provenance facts, are described in detail in [10].

In September, 2009, the W3C Provenance Incubator Group was created. The group’s mission was to “provide a state-of-the art understanding and develop a roadmap in the area of provenance for Semantic Web technologies, development, and possible standardization.” The group produced its final report in December 2010 [12]. The report highlighted the importance of provenance for multiple application domains, outlined typical scenarios that would benefit from a rich provenance description, and summarized the state of the art from the literature, as well as in the Web technology available to support tools that exploit a future standard provenance model. As a result, the W3C Provenance Working Group was created in 2011. The group released its final recommendations for PROV in June 2013 [13].

The Core PROV-O standard defines the following core elements (see [Figure 1](#)) [13] (see):

- **Entities:** Physical, digital, conceptual, or other kinds of thing are called entities. Examples of such entities are a web page, a chart, and a spellchecker.
- **Activities:** Activities generate new entities. For example, writing a document brings the document into existence, while revising the document brings a new version into existence. Activities also make use of entities.
- **Agents:** An agent takes a role in an activity such that the agent can be assigned some degree of responsibility for the activity taking place. An agent can be a person, a piece of software, an inanimate object, an organization, or other entities that may be ascribed responsibility.

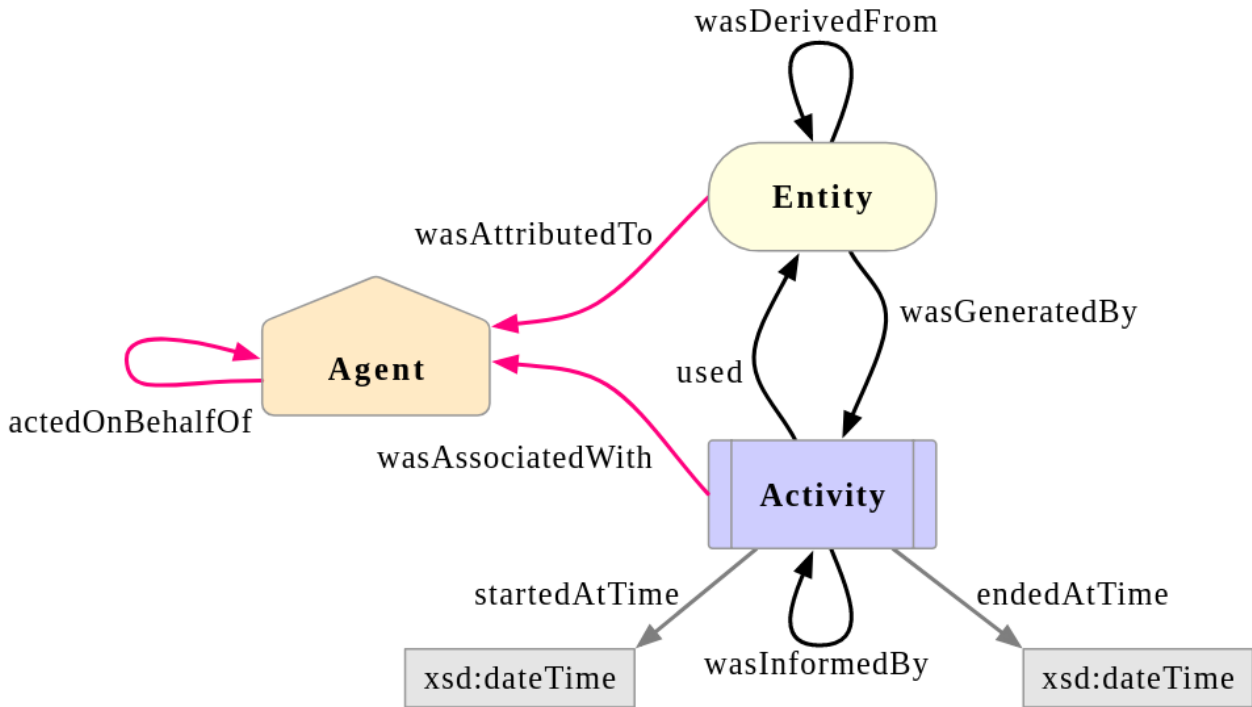


Figure 1. Prov-O Core Model

One of the advantages of PROV-O is that it is based on semantic web standards, thus provenance information can be interpreted by machines without ambiguity using well-defined semantics specified by the PROV-O ontology. The model can be extended, in principle, using the standard OWL extension mechanism (subclass, subproperty, and so forth) to address the needs of multiple disciplines. The PROV-O specification defines additional terms that extend the core concepts of the specification. Figure 2 depicts Entities as yellow ovals, Activities as blue rectangles, and Agents as orange pentagons. The domain of **prov:atLocation** (**prov:Activity** or **prov:Entity** or **prov:Agent** or **prov:InstantaneousEvent**) is not illustrated.

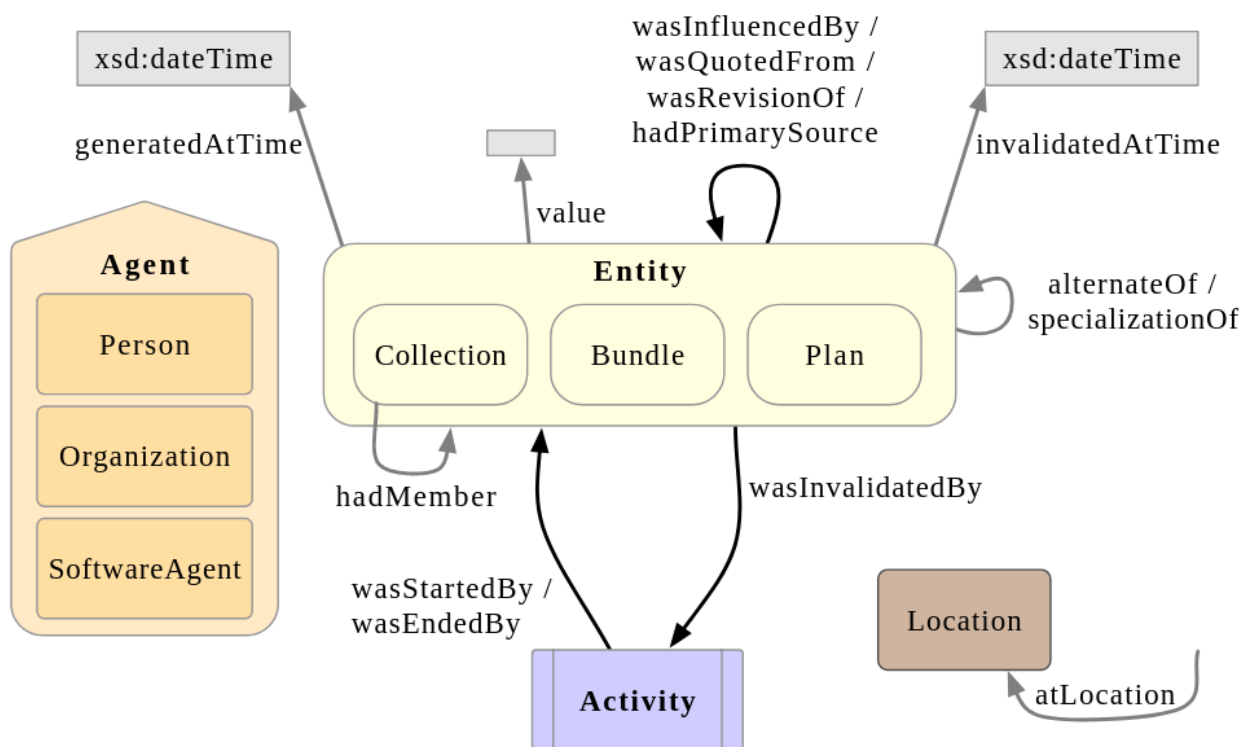


Figure 2. The expanded terms build upon those in the PROV-O core model.

Most of the provenance storage solutions are currently based on a database controlled by a central authority. This means that if the centralized authority is compromised, the data provenance can also be compromised and be tampered or destroyed. A reliable secure decentralized solution could be more suitable to address this issue.

6.2. Levels of Provenance and Resource Sharing

There are a number of studies that have investigated the role of automated workflows and published best practices to support workflow design, preservation, understandability, and reuse. A summary of the recommendations and their justifications has been summarized by Khan et al [14] by studying workflows from different domains (see Table 1). Their study classifies the recommendations into broad categories related to workflow design, retrospective provenance, the computational environment required/used for an analysis, and better findability and understandability of all shared resources. The recommendations have been informed by a wide corpus of literature [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [15], [25], [26], [27], [28]. Their findings were used to define the **CWLProv** model, which defines the provenance model for the Common Workflow Language (CWL).

Table 1. Summarized recommendations and justifications from the literature covering best practices on reproducibility, accessibility, interoperability, and portability of workflows cite:[khan2019sharing]

Requirement No.	Category	Recommendations	Justifications
R1-parameters	<i>Retrospective Provenance</i>	Save and share all parameters used for each software executed in a given workflow (including default values of parameters used).	Affects reproducibility of results because different inputs and configurations of the software can produce different results. Different versions of a tool might upgrade the default values of the parameters.
R2-automate	<i>Prospective Provenance</i>	Avoid manual processing of data, and if using shims, then make these part of the workflow to fully automate the computational process.	This ensures the complete capture of the computational process without broken links so that the analysis can be executed without the need for performing manual steps.
R3-intermediate	<i>Data Sharing</i>	Include intermediate results where possible when publishing an analysis.	Intermediate data products can be used to inspect and understand shared analysis when re-enactment is not possible.

Requirement No.	Category	Recommendations	Justifications
R4-sw-version	<i>Retrospective Provenance</i>	Record the exact software versions used.	This is necessary for reproducibility of results because different software versions can produce different results.
R5-data-version	<i>Retrospective Provenance</i>	If using public data (reference data, variant databases), then it is necessary to store and share the actual data versions used.	This is needed because different versions of data, e.g., human reference genome or variant databases, can result in slightly different results for the same workflow.
R6-annotation	<i>Prospective Provenance</i>	Workflows should be well-described, annotated, and offer associated metadata. Annotations such as user-contributed tags and versions should be assigned to workflows and shared when publishing the workflows and associated results.	Metadata and annotations improve the understandability of the workflow, facilitate independent reuse by someone skilled in the field, make workflows more accessible, and hence promote the longevity of the workflows.
R7-identifier	<i>Findability and Understandability</i>	Use and store stable identifiers for all artifacts including the workflow, the datasets, and the software components.	Identifiers play an important role in the discovery, citation, and accessibility of resources made available in open access repositories.
R8-environment	<i>Execution Environment</i>	Share the details of the computational environment.	Such details support analysis of requirements before any re-enactment or reproducibility is attempted.
R9-workflow	<i>Prospective Provenance</i>	Share workflow specifications/descriptions used in the analysis.	The same workflow specifications can be used with different datasets, thereby supporting reusability.
R10-software	<i>Execution Environment</i>	Aggregate the software with the analysis and share this when publishing a given analysis.	Making software available reduces dependence on third-party resources and as a result minimizes “workflow decay”.

Requirement No.	Category	Recommendations	Justifications
R11-raw-data	<i>Data Sharing</i>	Share raw data used in the analysis.	When someone wants to validate published results, availability of data supports verification of claims and hence establishes trust in the published analysis.
R12-attribution	<i>Retrospective Provenance</i>	Store all attributions related to data resources and software systems used.	Accreditation supports proper citation of resources cite:used.
R13-provenance	<i>Retrospective Provenance</i>	Workflows should be preserved along with the provenance trace of the data and results .	A provenance trace provides a historical view of the workflow enactment, enabling end users to better understand the analysis retrospectively.
R14-diagram	<i>Prospective Provenance</i>	Data flow diagrams of the computational analysis using workflows should be provided .	These diagrams are easy to understand and provide a human-readable view of the workflow.
R15-open-source	<i>Findability and Understandability</i>	Open source licensing for methods, software, code, workflows, and data should be adopted instead of proprietary resources.	This improves availability and legal reuse of the resources used in the original analysis, while restricted licenses would hinder reproducibility.
R16-format	<i>Findability and Understandability</i>	Data, code, and all workflow steps should be shared in a format that others can easily understand, preferably in a system-neutral language.	System-neutral languages help achieve interoperability and make an analysis understandable.
R17-executable	<i>Execution Environment</i>	Promote easy execution of workflows without making significant changes to the underlying environment.	In addition to helping reproducibility, this enables adapting the analysis methods to other infrastructures and improves workflow portability.

Requirement No.	Category	Recommendations	Justifications
R18-resource-use	<i>Execution Environment</i>	Information about compute and storage resources should be stored and shared as part of the workflow.	Such information can assist users in estimating the resources needed for an analysis and thereby reduce the amount of failed executions.
R19-example	<i>Data Sharing</i>	Example input and sample output data should be preserved and published along with the workflow-based analysis.	This information enables more efficient test runs of an analysis to verify and understand the methods used.

These recommendations can be clustered into broad themes as shown in [Figure 3](#).

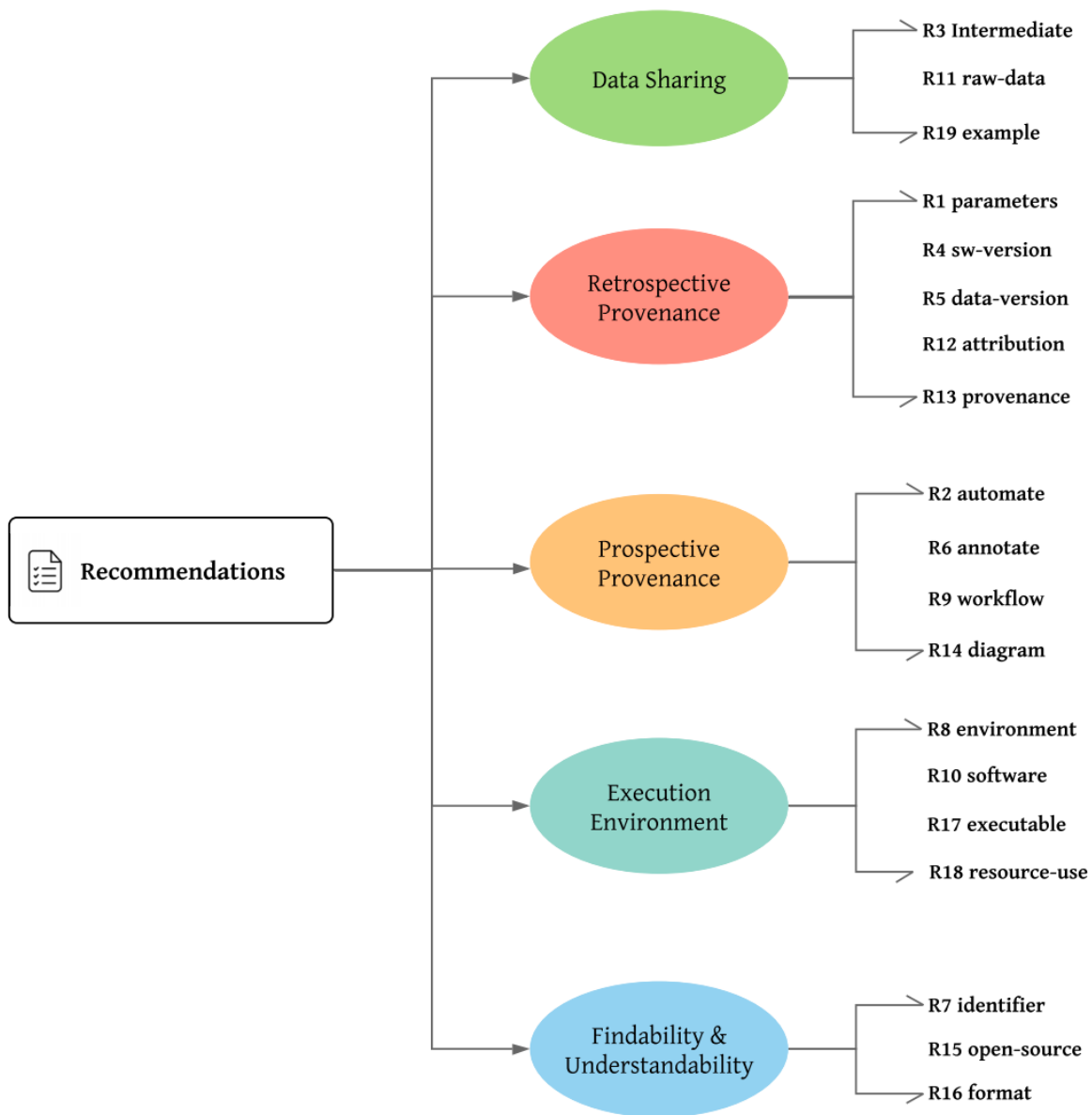


Figure 3. Recommendations from [Table 1](#) classified into categories.

6.3. Provenance Security

While considerable work has been done for provenance of workflow and documents, much less work has been done on securing the provenance information. Secure provenance is of paramount importance to federated cloud computing, yet it is still challenging today. Data provenance needs to be secured because it may contain sensitive/private information. Cloud service providers do not guarantee confidentiality of the data stored in dispersed geographical locations. Unless provenance information is secured and under appropriate access control policies for confidentiality and privacy, the information simply cannot be trusted. [29]. In the past few years, several studies have recognized the importance of securing provenance [29], [30], [31],[32].Lee and al. [31] provide a survey of the provenance security challenges in the cloud.

To guarantee the trustworthiness of data provenance, the data provenance scheme must satisfy the following general data security properties [32]:

- **Confidentiality:** "Data provenance of a sensitive piece of data (that is, the source data) may reveal some private information. Therefore, it is necessary to encrypt not only the source data but also the data provenance. Moreover, a query to and/or a response from the data provenance store may reveal some sensitive information. Thus, both the query and its response must be encrypted in order to guarantee confidentiality on the communication channel. Last but not least, if data provenance is stored in the outsourced environment, such as the cloud then the data provenance scheme must guarantee that neither the stored information nor the query and response mechanism must reveal any sensitive information while storing data provenance or performing search operations" [32].
- **Integrity:** "The data provenance is immutable. Therefore, the integrity must be ensured by preventing any kind of unauthorized modifications in order to get the trustworthy data provenance. The integrity guarantees that data provenance cannot be modified during the transmission or on the storage server without being detected" [32].
- **Unforgeability:** "An adversary may forge data provenance of the existing source data with fake data. Unforgeability refers to the source data being tightly coupled with its data provenance. In other words, an adversary cannot forge the fake data with existing data provenance (or vice versa) without being detected" [32].
- **Non-Repudiation:** Once a user takes an action, as a consequence, the data provenance is generated. A user must not be able to deny the ownership of the action once data provenance has been recorded. The non-repudiation ensures that the user cannot deny his action if he/she has taken any actions [32].
- **Availability:** "The data provenance and its corresponding source data might be critical" [32]. Therefore, the provenance of the data or the source data must be available at anytime from anywhere. "For instance, the life critical data of a patient is subject to high availability, considering emergency situations that can occur at any time. The availability of the data can be ensured by a public storage service such as provided by the cloud service provider" [32] or Content Addressable Storage (such as IPFS).

In other words, secure trustworthy provenance mechanisms ensure that the integrity of provenance chains are tamper evident, their contents are confidential, and auditors can verify their authenticity without having to know the contents.

Researchers have proposed several data provenance related efforts. The Provenance-Aware Storage System (PASS) was the first scheme to address the collection and maintenance of provenance data at the operation system level [33]. A file provenance system [34] was proposed to collect provenance data by intercepting file system calls below the virtual file system, which requires changes to operating systems. For cloud data provenance, S2Logger [35], was developed as an end to end data tracking tool which provides both file-level and block-level provenance in kernel space. In addition to data provenance techniques and tools, the security of provenance data and user privacy has also been explored. Asghar et al. [32] proposed a secure data provenance solution in the cloud, which adopts a two folder encryption method to improve privacy but at a higher computation cost. In SPROVE [36] provenance data confidentiality and integrity are protected using encryption and digital signature, but does not support provenance data querying capability. The kernel-level logging tool Progger [37] provides log tamper-evidence at the expense of user privacy. There are also efforts which use provenance data for managing cloud environments, such as, discovery of usage patterns for cloud resources, popularized resource reuse and fault management [38].

Chapter 7. Workflows

Scientific workflows design and management have become increasingly popular for compute-intensive and data-intensive scientific applications. The vision and promise of scientific workflows includes rapid, easy workflow design, reuse, scalable execution, and other advantages, e.g., to facilitate “reproducible science” through provenance (e.g., data lineage) support [39]. However important research challenges still remain. There is an urgent need for a common format and standard to define workflows and enable sharing of analysis results provenance information using a workflow environment. Cloud systems are structured in a fundamentally different way from other distributed systems, such as grids, and therefore present new challenges for the collection of provenance data.

7.1. Workflow Management Services

The *freezing* and packaging of a runtime environment that includes all the software components and their dependencies used in data analysis workflow is considered today as best practices and has been widely adopted in cloud computing environments where *images* and *snapshots* are used and shared by researchers [40]. To distribute system-wide software, various lightweight and container-based virtualization and package managers have emerged such as **Docker** and **Singularity**.

Docker [41] is a lightweight container-based virtualization technology that facilitates the automation of application development by archiving software systems and environments to improve portability of the applications on many common platforms including Mac OS X, Linux, Microsoft Windows and cloud instances. One of Docker’s main features is the ability to find, download, deploy and run container images that were created by other developers quickly. Within the context of Docker, the place where images are stored is called a **registry**, and Docker Inc. offers a public registry called the **Docker Hub** [<https://hub.docker.com/>]. You can think of the registry along with the docker client as the equivalent of **Node’s NPM** [<https://www.npmjs.com/package/node>], **Perl’s CPAN** [<https://www.cpan.org/>] or **Ruby’s RubyGems** [<https://rubygems.org/>].

Singularity [42] is a cross-platform open source container engine specifically supporting High Performance Computing (HPC) resources. Singularity can import Docker format software images. Singularity enables users to have full control of their environment. Singularity containers can be used to package entire scientific workflows, software and libraries, and even data. This means that you do not have to ask your cluster admin to install anything for you - you can put it in a Singularity container and run.

The sharing and preservation of runtime environment packaging is becoming regular practice in the workflow domain and is supported today by all the leading platforms managing cloud infrastructure and computing services. These cloud providers include Digital Ocean [43], Amazon Elastic Compute Cloud [44], Google Cloud Platform [45] and Microsoft Azure [46]. The instances launched on these platforms can be saved as snapshots to be analyzed or can be recreated to restore the computing state at analysis time.

7.2. Workflow description standards

The **Common Workflow Language (CWL)** [47] has emerged as a workflow definition standard for the heterogeneous workflow environments. CWL is an open standard for describing analysis workflows and tools in a way that makes them portable and scalable across a variety of software and hardware environments, from workstations to cluster, cloud, and HPC environments. CWL is designed to meet the needs of data-intensive science, such as Bioinformatics, Medical Imaging, Astronomy, High Energy Physics, and Machine Learning. CWL is developed by a multi-vendor working group consisting of organizations and individuals aiming to enable scientists to share data analysis workflows [48]. CWL has been widely adopted by a large number of organizations and been implemented in a large set of open source tools and **workflow management systems**.

While a common standard for describing analysis workflow is an important step toward interoperability of workflow management systems, it is also important to share and publish results of these workflow executions in an transparent, comprehensive, interoperable and secure manner. This is essential to reduce duplication of workflows, improve the trust and integrity of data and analyses, improve reproducibility of scientific workflows and catalyze the discovery of new knowledge. Currently, there is no common format defined and agreed upon for interoperable and secure provenance, workflow archiving or sharing.

CWLProv [14] has been proposed very recently as a format to represent any workflow-based computational analysis to produce workflow output artifacts that satisfy various levels of provenance. CWLProv is based on open source community-driven standards, interoperable workflow definitions CWL, structured provenance using the W3C PROV Model [6], and resource aggregation and sharing as workflow-centric Research Object (RO) [16] generated along the final outputs of a given workflow enactment.

Chapter 8. Blockchain Technologies

This section provides some background information about blockchain technologies. Blockchain is, in layman terms, a series of connected blocks that together form a chain. More technically, a blockchain is a shared, trusted and append-only ledger which contains transactions that have been made between users in the network. Blockchain is a combination of private key cryptography, peer-to-peer networking with an open ledger and incentivizing protocols. This ledger is distributed among participants in the peer-to-peer system where peers in the network store a copy of the ledger. The fact that the ledger is distributed throughout the network means that the peers have to reach consensus and agree on the order of the blocks. This is critical since it is essential that every peer in the network has the same view of the blockchain.

8.1. Governance in DLTs, Blockchain, Hybrid Blockchain

Governance in DLTs, Blockchain and Hybrid Blockchains can be described as follows [49]:

- **Distributed Ledger Technology (DLT):** "Governance in DLTs is mostly centralized in one or a few validator nodes that are identified and other nodes that might have read access with the permission of the validator nodes. Governance is closed and the network of nodes is permissioned and new nodes can only join with permission from the validator nodes" [49].
- **Hybrid Blockchains:** "Governance in a semi-public or public permissioned Blockchain and is often defined by the validators nodes that are identified, usually, public institutions like governments agencies, educational institutions or corporates. However, read access to the ledger is open to everyone, which is different from DLTs where you need to be invited to have read access" [49].
- **Blockchain:** "The governance in a blockchain such as Bitcoin is decentralized through the global distribution of all nodes each having all the data of the blockchain, the free software that allows anyone to participate in "Bitcoin" and the relative decentralization of mining (which has become more centralized over time) to reach a consensus on the truth of the blockchain. In Bitcoin and similar blockchain efforts everything is aimed at maximizing decentralization" [49].

8.2. Permissioned vs. Permissionless and Private versus Public

To meet the different requirements for applications and companies, in the recent years, different types of blockchains have been developed. Data stored in the blockchain can vary in importance and sensibility and thus it is important to control the access to this information. For example, Bitcoin blockchain is a *public* and *permissionless* blockchain that allows anyone to read and send transactions to the network. A *permissionless* blockchain means that everyone can contribute to the consensus process to validate new block of transactions to be added to the blockchain.

Open networks must use proof-based algorithms to establish trust because members of the network are inherently untrustworthy. Thus, by providing proof that is acceptable to a majority of the remaining network, nodes can be added to the list. Since the proof might be computationally

expensive, cryptocurrencies provide the incentive of receiving coins in the currency appropriate to the work.

If the primary motivation of such a costly proof scheme is to create trust between untrustworthy parties, one obvious alternative is to work only with trustworthy nodes. **Permissioned** networks are composed of nodes identified by cryptographic keys. Other members of the network provide the permission to these nodes to join the network. Consensus is reached in this situation by simply making sure that the source and purpose of the transaction are valid between some parties on the network. There is no need of incentive to check transactions. The simple checking of membership of the parties in the network is sufficient. This improves performance and alleviates any concern about the work, stake, or cost required to participate. Example of permissioned blockchain is the open source [Hyperledger Fabric](https://www.hyperledger.org/projects/fabric) [https://www.hyperledger.org/projects/fabric] distributed ledger software.

8.3. Hashing

To maintain the reliability and integrity of the blockchain and avoid the recording fraudulent data or double spending transactions, the technology relies on one of its key component: **hashing**. **Hashing** is a mathematical algorithm (such as SHA-256 used by Bitcoin) that takes an input of any length and turns it into a cryptographic fixed output. Examples of such inputs can include a short piece of information such as a message or smart contracts, a block of transactions or an address of a content in a content-addressable system such as IPFS.

8.4. Consensus

Consensus protocols are one of the most important and revolutionary aspects of blockchain technology. These protocols create an irrefutable system of agreement between various devices across a distributed network, whilst preventing exploitation of the system.

Blockchain consensus protocols are what keep all the nodes on a network synchronized with each other, while providing an answer to the question: How do we all make sure that we agree on what the truth is?

‘Consensus’ means that the nodes on the network agree on the same state of the blockchain. This, in a sense makes it a self-auditing ecosystem. This is an absolutely crucial aspect of the technology, carrying out two key functions. Firstly, consensus protocols allow a blockchain to be updated, while ensuring that every block in the chain is true and in many cases keeping participants incentivized. Secondly, it prevents any single entity from controlling or derailing the whole blockchain system. The aim of consensus rules is to guarantee a single chain is used and followed.

8.5. IPFS

The InterPlanetary File System (IPFS) is likely the foremost P2P file system at the moment, represents the state of the art in the quest for a distributed web. IPFS takes ideas from previous P2P systems such as distributed hash tables (DHTs), BitTorrent, Git, and and Self-Certified Filesystems (SFS), and tries to simplify those ideas and take them even farther. Nodes in an IPFS network store objects (files and other data structures) in local storage and connect to each other to transfer these objects [50].

Nodes in the IPFS network are identified by a "NodeId", which is the cryptographic hash of a public key. When a node in the network requests a file or other network objects, the network's routing system finds peers who can serve the requested objects (known as "blocks"), gets the corresponding network addresses, and connects the requester to the discovered peers. When two peers first connect, they exchange public keys and check to make sure the connection is secure. If the check fails, the connection ends. Once the peers have connected, they use an IPFS protocol called BitSwap, which is based on BitTorrent, to "barter" and exchange blocks. This bartering process is meant to prevent freeloader nodes from exploiting the file system. Once the exchange is over, the requester node now has full copies of the blocks it received and can then share them with the next requester [50].

IPFS uses a [Mergle DAG](https://docs.ipfs.io/guides/concepts/merkle-dag/) [https://docs.ipfs.io/guides/concepts/merkle-dag/] (directed acyclic graph) where links between objects are cryptographic hashes of the content the link refers to. This means that all content in the IPFS, including links, is identified by its hash checksum. This, in turn, helps prevent data tampering and eliminates data duplication. It also means that unlike HTTP, which is location-addressed, IPFS is content-addressed.

Objects in IPFS are immutable and permanent, and Git-like versioning is built into the system. Old versions of objects, like every other object in IPFS, can simply be retrieved by their hash checksum.

Although its underlying store is immutable and content-addressed, IPFS does support mutable paths through a decentralized naming system called InterPlanetary Name Space (IPNS). IPNS takes advantage of the mutable state routing system in IPFS to store object hashes as metadata values that point to objects, can be changed, and, if the end user so chooses, can also point to previous versions of the same object. One drawback of the IPNS system is that it does not result in human friendly paths, so another layer needs to be added on top of IPNS if human friendly paths are desired.

8.6. BitTorrent File System (BTFS)

BitTorrent File System (BTFS) [51] is both a protocol and network implementation that provides a content-addressable, peer-to-peer mechanism for storing and sharing digital content in a decentralized file system using BitTorrent protocol. BTFS provides a foundation platform for Decentralized Applications, known as DApps.

BitTorrent, the largest P2P network in the world, still relies on centralized torrent file distribution. These torrent repositories are prone to security breaches, outages, and censorship. There have been numerous instances of attacks on torrent hosting web servers reducing service reliability. With a decentralized repository of torrent files leveraging the version control properties of BTFS, users can more reliably access torrent files.

The BitTorrent File System (BTFS), created by the makers of BitTorrent, is an upcoming distributed file system implementation that began as a fork of the IPFS implementation. BTFS promises to take what IPFS has accomplished and present an improved version that is more ready for widespread adoption. The makers of BTFS plan to integrate the existing BitTorrent P2P file exchange system into BTFS, which could help BTFS gain the wide use that IPFS has largely failed to attain [51].

By leveraging the massive existing infrastructure of BitTorrent user nodes (close to 100 million), BTFS is aimed at becoming the largest distributed storage network as well as the world's largest

distributed media sharing network [51].

One of the primary advantages BTFS claims to have over IPFS is its integration of native token economics. These tokens are intended as an incentive for network nodes to contribute by storing data and not simply leeching of existing nodes. BTFS is also set to release with a set of developer tools that will make working with the file system more user friendly than other alternatives. BTFS claims a public version of the system will be available by 2020.[51]

8.7. Hyperledger Fabric

Hyperledger Fabric [52] is an open source enterprise-grade permissioned DLT platform, designed for use in enterprise contexts, that delivers some key differentiating capabilities over other popular distributed ledger or blockchain platforms. Hyperledger Fabric is maintained by the Linux Foundation and evangelized by IBM. Fabric is currently one of the most widely adopted blockchain technologies by the biggest enterprises, much more so than the “big name” blockchains. Companies such as Oracle, Walmart, Airbus, Accenture, Daimler, Thales, The National Association of Realtors, Deutsche Borse Group, and Sony Global Education are all **members** [<https://www.hyperledger.org/members>] of the community that develops and maintains Fabric.

One key point of differentiation is that Hyperledger was established under the Linux Foundation, which itself has a long and very successful history of nurturing open source projects under **open governance** that grow strong sustaining communities and thriving ecosystems. Hyperledger is governed by a diverse technical steering committee, and the Hyperledger Fabric project by a diverse set of maintainers from multiple organizations. Fabric has a development community that has grown to over 35 organizations with nearly 200 developers since its earliest commits.

Fabric has a highly **modular** and **configurable** architecture, enabling innovation, versatility and optimization for a broad range of industry use cases including banking, finance, insurance, healthcare, human resources, supply chain and digital product delivery.

"Fabric is the first distributed ledger platform to support smart contracts authored in general-purpose programming languages such as Java, Go and Node.js, rather than constrained domain-specific languages (DSL) and uses Docker container technology for deployment of smart contracts (what Fabric calls “chaincode”)" [52]. This means that most enterprises already have the skill set needed to develop smart contracts with no additional training to learn a new language or DSL is needed.

The Fabric platform is also **permissioned**, meaning that, unlike with a public permissionless network, the participants are known to each other, rather than anonymous and therefore fully untrusted. This means that while the participants may not fully trust one another (they may, for example, be competitors in the same industry), a network can be operated under a governance model that is built off the trust that does exist between participants, such as a legal agreement or framework for handling disputes.

One of the most important of the platform’s differentiators is its support for **pluggable consensus protocols** that enable the platform to be more effectively customized to fit particular use cases and trust models. For instance, when deployed within a single enterprise, or operated by a trusted authority, fully byzantine fault tolerant consensus might be considered unnecessary and an excessive drag on performance and throughput.

Fabric can leverage consensus protocols that **do not require a native cryptocurrency** to incent costly mining or to fuel smart contract execution. Avoidance of a cryptocurrency reduces some significant risk/attack vectors, and absence of cryptographic mining operations means that the platform can be deployed with roughly the same operational cost as any other distributed system.

Hyperledger Fabric delivers a uniquely elastic and extensible architecture, distinguishing it from alternative blockchain solutions. The combination of these differentiating design features makes Fabric one of the best distributed ledger platforms available today both in terms of high degrees of confidentiality, resiliency, flexibility and scalability.

8.8. Smart Contracts

The concept of a Smart Contract was introduced in 1994 by Nick Szabo [53], a legal scholar, and cryptographer. Szabo came up with the idea that a decentralized ledger could be used for smart contracts, otherwise called self-executing contracts, blockchain contracts, or digital contracts. In this format, contracts could be converted to computer code, stored and replicated on the system and supervised by the network of computers that run the blockchain. Contracts would be activated automatically when certain conditions are met.

A smart contract, or simply a contract, in the context of blockchain is a small piece of code that is executed in response to a transaction. Business logic executed on the network is done through contracts. Contracts can have many uses, although some ledgers may limit the type of code that can be executed for either architectural reasons or security. Smart contracts are automatically executable lines of code that are stored on a blockchain which contain predetermined rules. When these rules are met, the code executes on its own and provides the output. In the simplest form, smart contracts are programs that run according to the format that has been set up by their creator. Smart contracts are most beneficial in business collaborations in which they are used to agree upon the decided terms set up by the consent of both the parties. This reduces the risk of fraud, as there is no third-party involved. The costs are also reduced.

In the context of blockchain, a smart contract is defined as machine processable business agreement embedded into the transaction database and executed with transactions. The function of the contract is to define the rules defining the flow of value and state of a transaction needed in business transaction. The contract is smart because it executes the terms of contract using computerized protocol. The core idea behind smart contracts is to codify various contractual clauses (such as acceptance criteria, delineation of property rights, and so forth) to enforce compliance with the terms of the contract and ensure a successful transaction. Smart contracts are designed to guarantee one party that the other will fulfill their agreement. One of the benefits of smart contracts is to reduce the costs of verification and enforcement. Smart contracts are required to be observable (meaning that participants can see or prove each other's actions pertaining to the contract), verifiable (meaning that participants can prove to other nodes that a contract has been performed or breached), and private (meaning that knowledge of the contents/performance of the contract should involve only the necessary participants required to execute it). Bitcoin has basic support for smart contracts. However, it lacks some essential capabilities such as Turing-completeness, lack of state, and so on.

Launched in 2015, **Ethereum** blockchain is the world's leading public programmable blockchain supporting Smart contracts. It replaces Bitcoin's more restrictive script language with one that

enables developers to build their own decentralized applications. On this platform, smart contracts are implemented with the [Solidity](https://solidity.readthedocs.io/en/v0.6.0/) [https://solidity.readthedocs.io/en/v0.6.0/] scripting language, which is Turing Complete. Solidity is currently the most prominent public smart contracts framework that allows anyone to write smart contracts and decentralized applications by creating their own arbitrary rules for ownership, transaction formats, and state transition functions.

In the open source **Hyperledger Fabric**, smart contracts, a.k.a. **ChainCode** program can be written in Go, node.js, or Java. Chaincode is installed on peers and require access to the asset states to perform reads and writes. Chaincode runs in a secured Docker container isolated from the endorsing peer process. The chaincode is then instantiated on specific channels for specific peers. Chaincode initializes and manages the ledger state through transactions submitted by applications. A chaincode typically handles business logic agreed to by members of the network, so it similar to a “smart contract”. A chaincode can be invoked to update or query the ledger in a proposal transaction. Given the appropriate permission, a chaincode may invoke another chaincode, either in the same channel or in different channels, to access its state. Note that, if the called chaincode is on a different channel from the calling chaincode, only read query is allowed. That is, the called chaincode on a different channel is only a Query, which does not participate in state validation checks in subsequent commit phase.

Smart contracts are one of the most successful applications of blockchain technology. Using smart contracts in place of traditional ones can reduce the transaction costs and trusted intermediaries significantly and improve automation and security. They are also tamper-proof, as no one can change what has been programmed.

Chapter 9. Identity Management

Data in Clouds is geographically dispersed and is frequently accessed by a number of actors. Actors are active elements inside or outside the network, including cloud services, peers, client applications, administrators, and so forth. In such a shared and distributed environment, data moves from one point to another through communication networks. The number of data transactions increases as the number of users and volume of data increases. The growing interactions with this dispersed data increases the chances of lost data, data alteration and/or unauthorized access.

To determine the exact permissions over the resource and access to information of the ledger, **digital identity** plays a central role. Identity is the fundamental asset of how parties are interacting, thus playing a central role in securing access and ensuring the integrity of information. Digital Identity for people, organization, devices and secure, encrypted, privacy-preserving storage and computation of data are critical and play a central role in the establishment of the Web of Trust. This section describes the evolution of online identity models from centralized toward Self-Sovereign Identity (SSI), explains Public Key Infrastructures (PKIs) and demonstrate how DLTs and emerging SSI standards can be used as a foundation of an Identity Layer for Internet and Web of Trust based on Decentralized Public Key Infrastructure (DPKI).

9.1. Evolution of Online Identity Model

The Internet was built without an Identity Layer. That is, there was no standard way of identifying people and organizations. The addressing system was based solely on identifying machine physical endpoints, not people or organizations. In his excellent article “The Path to Self-Sovereign Identity” [54], Christopher Allen provides a clear analysis of the online identity landscape and describes the evolutionary path composed of 4 broad stages since the advent of the Internet: Centralized identity, Federated identity, User-Centric identity, and Self-Sovereign identity (see [Figure 4](#)).

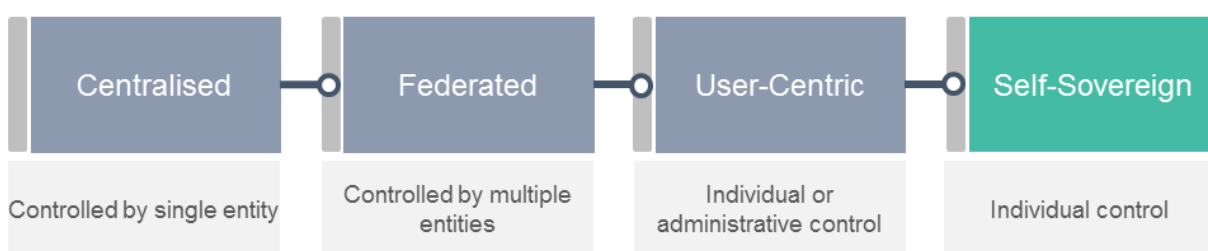


Figure 4. The evolution of online identity

The evolution of internet identity is the result of trying to satisfy three basic requirements [55]:

1. **Security** - the identity information must be protected from unintentional disclosure;
2. **Control** - the identity owner must be in control of who can see and access their data and for what purposes;
3. **Portability** - the user must be able to use their identity data wherever they want and not be tied into a single provider.

The following describes the different evolution stages of identity model in more detail.

9.1.1. Centralized Identity

Due to the lack of identity layer on Internet, several web sites started to provide their own identity management by providing username/password-protected accounts. These systems were centralized, owned and controlled by a single entity, such as an eCommerce website or a social network (Google, Facebook). The user does not own their identity record as it can be taken away at any time due to policy violations, web site shutdown or censorship for example. This effectively erases a person's online identity, which may have been used for years and may be of significant value to them, and impossible to replace.

As the Internet expands rapidly to new types of devices (Internet of Things), online web sites and services, the maintenance of identity on every endpoint becomes unsustainable. Users are requested to provide the same identification information over and over again, remember the plethora of user accounts and passwords can be subject to hacking and data breaches if private information is not secured enough.

9.1.2. Federated Identity

To address some of the problems of centralization, the usage of federated identity management provides a degree of portability to a centralized identity. This is done by enabling a user to login into one service using the credentials of another (e.g. Facebook or Google account in the consumer internet). At a more complex level, it can allow different services to share details about the user. Federation is common within large businesses, where single sign-on mechanisms allow a user to access multiple separate internal services such as Human Resources, accounting, etc., with a single username and password. Although federation provides a semblance of portability, the concentrated control of management of identities to a small number of corporations increases by order of magnitude the risk of hacking. Further, the implications to a user of having their centrally federated account deleted or compromised are much more profound if that account is their key to many other 3rd party services [55]. In addition, the cost and growing economic inefficiency to collect, store, and protect personal data is reaching a tipping point today.

9.1.3. User-Centric Identity

The Identity Commons (2001-Present) established to consolidate new work on digital identity with a focus on decentralization. Their most important contribution may have been the creation, in association with the Identity Gang, of the Internet Identity Workshop (IIW) (2005-Present) working group. The IIW community focused on a new term that countered the server-centric model of centralized authorities: **user-centric identity**. The term suggests that users are placed in the middle of the identity process. The work of the IIW has supported many new methods for creating digital identity, including OpenID (2005), OpenID 2.0 (2006), OpenID Connect (2014), OAuth (2010), and FIDO (2013). As implemented, user-centric methodologies tend to focus on two elements: user consent and interoperability. By adopting these elements, a user can decide to share an identity from one service to another and thus de-balkanize their digital self [54].

User-centric identity is most frequently manifested in the form of independent personal data stores at one end of the spectrum, and large social networks at the other end. However the entire

spectrum still relies on the user selecting an individual identity provider and agreeing to their often one-sided adhesion contracts [55]. The user-centric identity communities intended to give users complete control of their digital identities. Unfortunately, powerful institutions such as Facebook or Google co-opted their efforts and kept them from fully realizing their goals. Much as with the Liberty Alliance, final ownership of user-centric identities today remains with the entities that register them [54].

OpenID offers an example. A user can theoretically register his own OpenID, which can then use autonomously. However, this takes some technical know-how, so the casual Internet user is more likely to use an OpenID from one public web site as a login for another. Users are at the mercy of selecting a long-lived and trustworthy site to gain many of the advantages of a self-sovereign identity, which could take it away at any time by the registering entity.

Facebook Connect (2008) appeared a few years after OpenID, leveraging lessons learned, and thus was several times more successful largely due to a better user interface. Unfortunately, Facebook Connect veers even further from the original user-centric ideal of user control, not only because they are the only provider but also because Facebook has a history of arbitrarily closing accounts, as was seen in their recent real-name controversy [56]. As a result, people using their “user-centric” Facebook Connect identity to connect to other sites may be even more vulnerable than OpenID users to losing that identity in multiple places at one time [54].

9.1.4. Self-Sovereign Identity

Self-sovereign identity (SSI) is the last step in the digital identity evolution. SSI is independent from any individual silo, and provides all three required elements: individual control, security, and full portability. SSI addresses the problems of the centralized external controls from the three previous phases. The individual (or organization) to whom the identity pertains completely owns, controls and manages their identity. There is no external party who can claim to “provide” the identity for them or to take away their identity because it is intrinsically theirs. The individual’s digital existence is independent of any single organization [55].

In a 2016 article [57], Phil Windley describes self-sovereign identity as an “Internet for identity” which, like the Internet itself, has three virtues: no one owns it, everyone can use it, and anyone can improve it. Similar to the internet, the move to self-sovereign identity is a move from a silo mentality to a layer mentality. Instead of having every organization maintaining their own siloed identity information store with possibly a suite of APIs to connect to other such silos, each organization can have one single connection to the Internet’s identity layer, and immediately benefit from all the organizations that are already present.

Figure 5 illustrates the migration from centralized to decentralized peer-to-peer self-sovereign identity. In the current identity model, every identity is given by an organization. For example, to do business with Amazon, you need to create an account to use their service and you are subject to the terms and conditions to use their service. This can be changed retroactively and the account can be taken down at any time if you violate their terms. To change the identity model, people need to have full control of their identities, which basically turn the centralized model inside-out, resulting in a self-sovereign identity model. Each organization and person establishes their identities independently and is seen as a peer in a peer-to-peer network that is decentralized and not controlled by anyone.

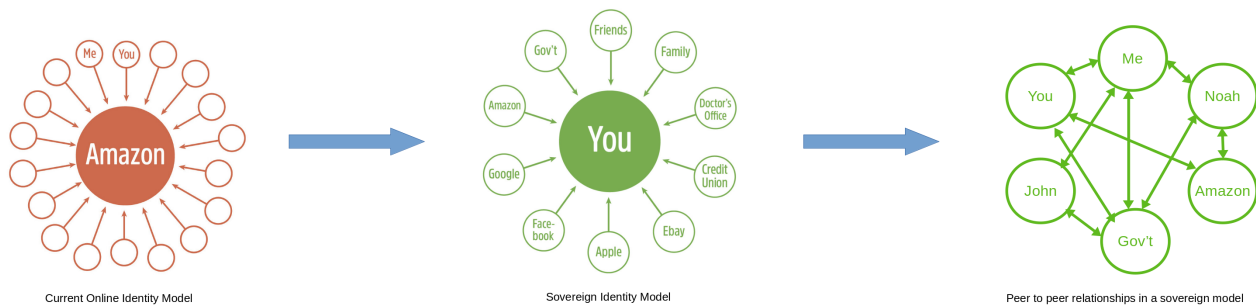


Figure 5. Evolution from centralized to self-sovereign identity model

To enable the missing identity layer of the Internet, a new and trusted infrastructure is needed that enables identity owners to share and control their identity but also has verifiable claims with full permission and consent. In the context of this study, the verifiable claims can be used to identify parties of a federated cloud workflow and to guarantee the integrity of information produced by a given party with reference to provenance information. This infrastructure needs to be decentralized, distributed around the world, permanent and not controlled by any single company, organization or government. DLTs are the breakthrough that makes this possible. They enable people, organizations, institutions and government to collaborate by forming a decentralized network, where information is replicated in different locations to address security, fault-tolerance and tampering of information. By combining peer-to-peer sharing of verifiable claims and distributed key management, DLTs make SSI possible. The discovery, routing of requests, exchange of data and recording events can exist pervasively, without control by any organization [55]. The open, decentralized systems enable individuals to fully own and manage their own identities.

The following section describes the different approaches, technologies and emergent standards used to manage identity in federated clouds, DLTs and blockchains.

9.2. Centralized Public key infrastructure (PKI)

Public Key Infrastructure (PKI) is a system of hardware, software, people, policies, documents, and procedures. PKI includes the creation, issuance, management, distribution, usage, storage, and revocation of digital certificates. These certificates are then used to authenticate the identities of various actors across the data transfer process. They also assure that the data being moved between these actors is secured and encrypted in a way that both parties can decrypt. This way, information is only being sent to and received from known and trusted sources, and both parties are assured of the information's integrity.

Currently the most commonly employed approach for Identity Management is based on **Web PKI**, a centralized trust infrastructure based on **public key infrastructures (PKIs)**. Web PKI is a Certificate Authority (CA) based system that adopts a centralized trust infrastructure. Communications over the internet are secured through the safe delivery of public keys and the corresponding private keys.

PKI trust is established by a certificate authority, which is an organization or governing body that can issue certificates and verify the identity of the certificate requestor. For example, AWS offers multiple PKI certificate authority services that can help organization to easily and securely manage their certificate infrastructure.

The open source **Hyperledger Fabric**, which is used to implement permissioned blockchains, adopts the traditional PKI hierarchical model. Identity verification come from a trusted authority called **membership service provider (MSP)**. An MSP is a component that defines the rules that govern the valid identities for this organization. For example, the default MSP implementation in Hyperledger Fabric uses X.509 certificates as identities. MSPs turn verifiable identities into the members of a blockchain network.

The following describes, in more detail, the four key elements to PKI:

- Digital Certificates
- Public and Private Keys
- Certificate Authorities
- Certificate Revocation Lists

9.2.1. Digital Certificates

A digital certificate is a document which holds a set of attributes relating to the holder of the certificate. The most common type of certificate is the one compliant with the X.509 standard, which allows the encoding of a party's identifying details in its structure. For example, Mary Morris in the Manufacturing Division of Mitchell Cars in Detroit, Michigan might have a digital certificate with a SUBJECT attribute of C=US, ST=Michigan, L=Detroit, O=Mitchell Cars, OU=Manufacturing, CN=Mary Morris /UID=123456. Mary's certificate is similar to her government identity card — it provides information about Mary which she can use to prove key facts about her. There are many other attributes in an X.509 certificate, but our study concentrates on just these for now.

```

Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number: 1 (0x1)
    Signature Algorithm: md5WithRSAEncryption
    Issuer: C=ZA, ST=Western Cape, L=Cape Town, O=Thawte Consulting cc,
           OU=Certification Services Division,
           CN=Thawte Server CA/Email=server-certs@thawte.com
    Validity
      Not Before: Aug  1 00:00:00 1996 GMT
      Not After : Dec 31 23:59:59 2020 GMT
    Subject: C=ZA, ST=Western Cape, L=Cape Town, O=Thawte Consulting cc,
           OU=Certification Services Division,
           CN=Thawte Server CA/Email=server-certs@thawte.com
    Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
      RSA Public Key: (1024 bit)
        Modulus (1024 bit):
          00:d3:a4:50:6e:c0:ff:56:6b:e6:cf:5d:b6:ea:0c:
          68:75:47:a2:aa:c2:da:84:25:fc:a8:f4:47:51:da:
          85:b5:20:74:94:86:1e:0f:75:c9:e9:08:61:f5:06:
          6d:30:6e:15:19:02:e9:52:c0:62:db:4d:99:9e:e2:
          6a:0c:44:38:cd:fc:bc:e3:64:09:70:c5:fc:bl:6b:
          29:b6:2f:49:c0:3b:d4:27:04:25:10:97:2f:e7:90:
          6d:c0:28:42:99:d7:4c:43:de:c3:f5:21:6d:54:9f:
          5d:c3:58:e1:c0:e4:d9:5b:b0:b8:dc:b4:7b:df:36:
          3a:c2:b5:66:22:12:d6:87:0d
        Exponent: 65537 (0x10001)
    X509v3 extensions:
      X509v3 Basic Constraints: critical
      CA:TRUE
    Signature Algorithm: md5WithRSAEncryption
    07:fa:4c:69:5c:fb:95:cc:46:cc:85:83:4d:21:30:8e:ca:d9:
    a8:6f:49:1a:e6:da:51:e3:60:70:6c:84:61:11:a1:1a:c0:48:
    3e:59:43:7d:4f:95:3d:a1:8b:b7:0b:62:98:7a:75:8a:dd:88:
    4e:4e:9e:40:db:a8:cc:32:74:b9:6f:0d:c6:e3:b3:44:0b:d9:
    8a:6f:9a:29:9b:99:18:28:3b:d1:e3:40:28:9a:5a:3c:d5:b5:
    e7:20:1b:8b:ca:a4:ab:8d:e9:51:d9:e2:4c:2c:59:a9:da:b9:
    b2:75:1b:f6:42:f2:cf:c7:f2:18:f9:89:bc:a3:ff:8a:23:2e:
    70:47

```

Figure 6. X509 Digital Certificate Sample

9.2.2. Authentication, Public keys, and Private Keys

Authentication and message integrity are important concepts in secure communications. Authentication requires that parties who exchange messages are assured of the identity that created a specific message. For a message to have “integrity” means that it cannot have been modified during its transmission. Traditional authentication mechanisms rely on digital signatures that, as the name suggests, allow a party to digitally sign its messages. Digital signatures also provide guarantees on the integrity of the signed message.

Technically speaking, digital signature mechanisms require each party to hold two cryptographically connected keys: a **public key** that is made widely available and acts as authentication anchor, and a **private key** that is used to produce digital signatures on messages. Recipients of digitally signed messages can verify the origin and integrity of a received message by checking that the attached signature is valid under the public key of the expected sender (see [Figure 7](#)).

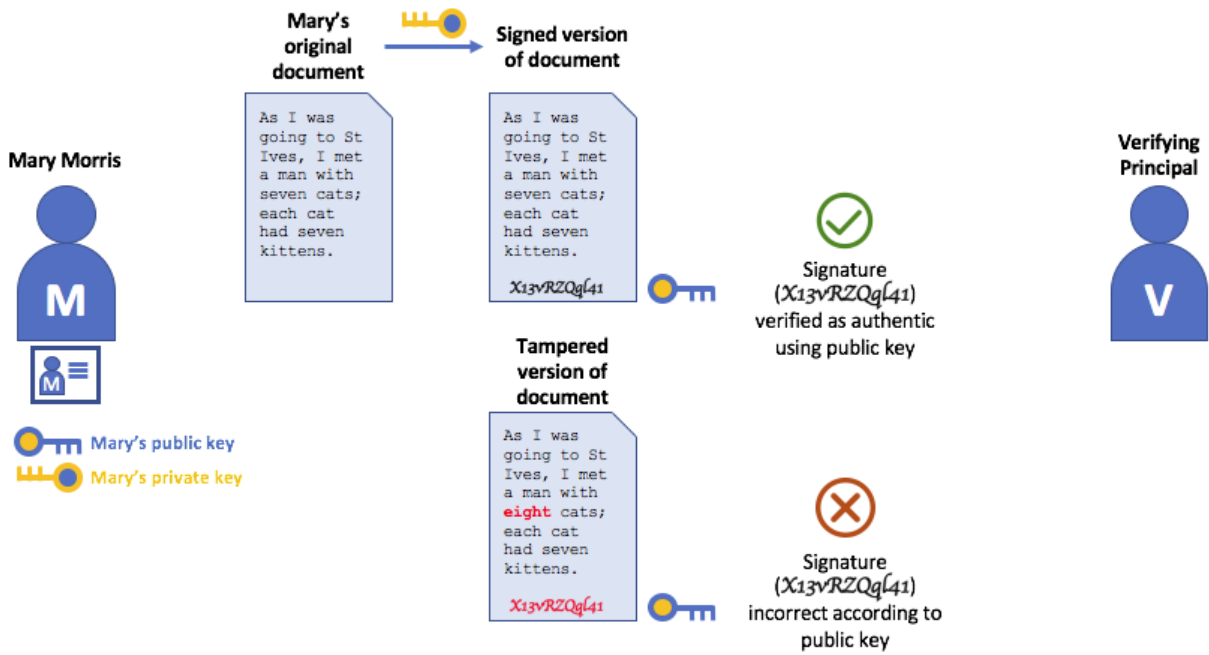


Figure 7. Authentication with Public/Private Keys

9.2.3. Certificate Authorities

Digital identities (or simply identities) are encoded in the form of cryptographically validated digital certificates that comply in the most common case with X.509 standard and are issued by a Certificate Authority (CA).

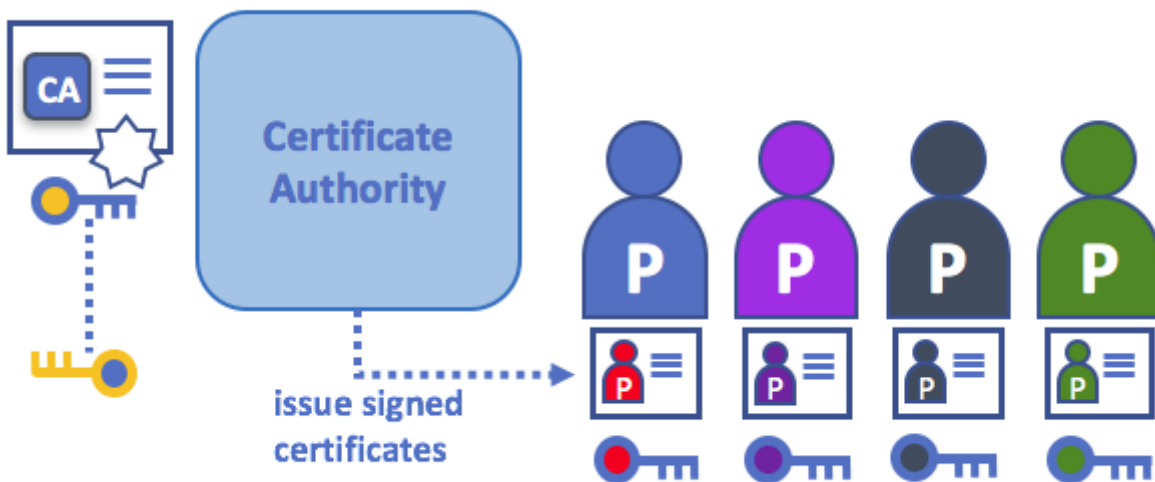


Figure 8. Certificate Authority

They are two types of CAs: **Root CAs** and **Intermediate CAs**. To securely distribute the millions of certificates to internet users, Root CAs (Symantec, Geotrust, etc) use Intermediate CAs to spread this process out. Each Intermediate CAs have their certificates issued by the root CA or another intermediate authority, allowing the creation of a “chain of trust” (see Figure 9) for any certificate that is issued by any CA in the chain. This ability to track back to the Root CA not only allows the function of CAs to scale while still providing security — allowing organizations that consume certificates to use Intermediate CAs with confidence. This limits the exposure of the Root CA, which,

if compromised, would endanger the entire chain of trust. If an Intermediate CA is compromised, on the other hand, there will be a much smaller exposure.

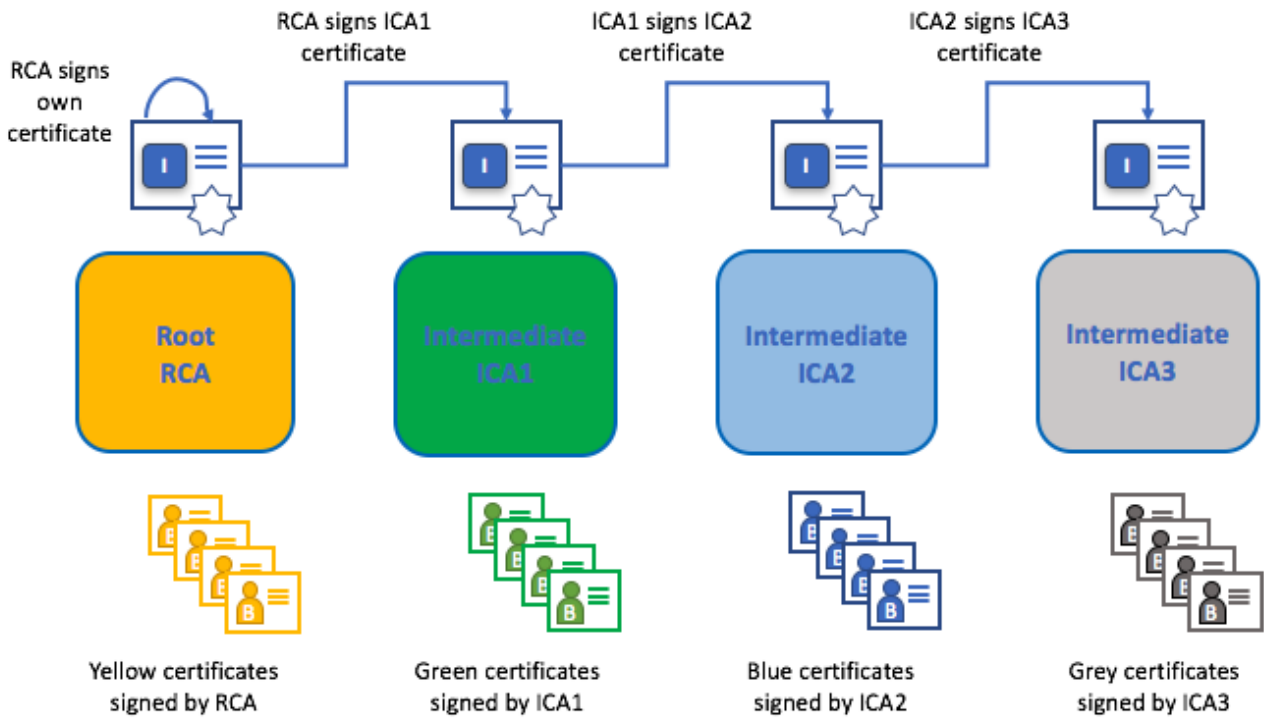


Figure 9. Chain of Trust

9.2.4. Certificate Revocation Lists

A Certificate Revocation List (CRL) is a list of references to certificates that a CA knows to be revoked for one reason or another. This is similar to a list of stolen credit cards. When a third party wants to verify another party's identity, the first step is to check the issuing CA's CRL to ensure that the certificate has not been revoked. A verifier does not have to check the CRL, but if they don't they run the risk of accepting a compromised identity (see Figure 10)

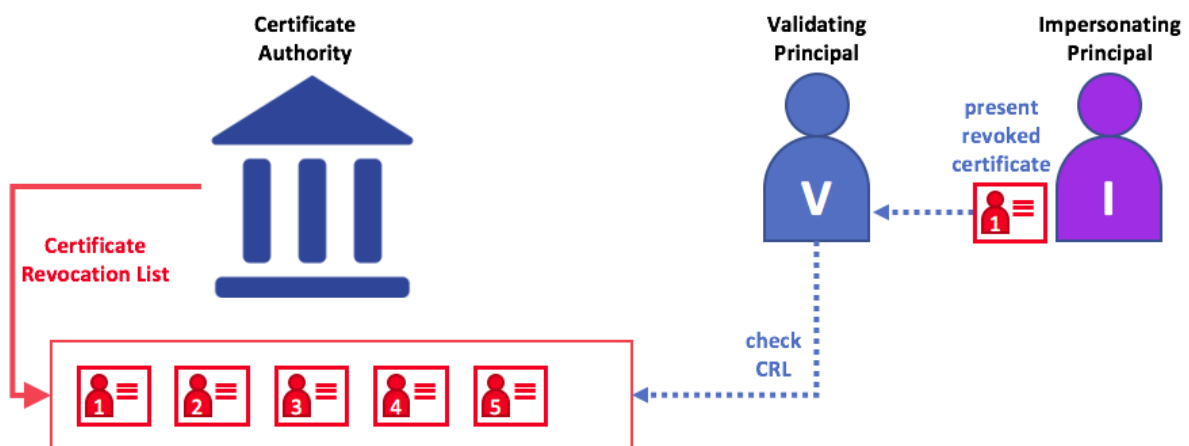


Figure 10. Certificate Revocation Lists

9.2.5. Problems with Centralized PKIs

The centralized PKIs such as the CA-based system have their problems and limitations generally

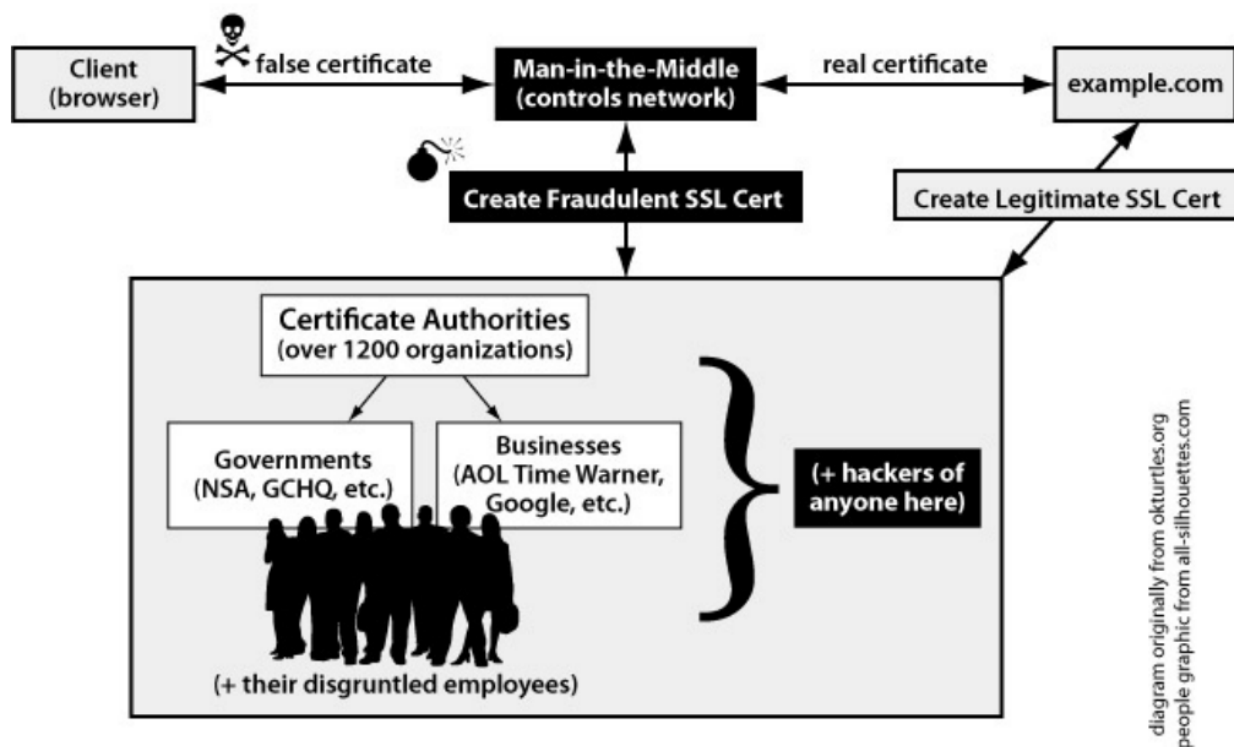
because they rely on a central trusted party. In a centralized PKI system, user identity is defined by trusted third parties, sometimes private companies and sometimes governments, not by the user. This leaves the door open for attackers to conduct MITM (Man-in-the-Middle) attacks, that can take different forms —Address Resolution Protocol (ARP) spoofing, Internet Protocol (IP) address spoofing, Domain Name System (DNS) spoofing, HyperText Transfer Protocol Secure (HTTPS) spoofing, and Man in the Browser (MITB), and more. The large number of trusted CAs is an attractive target for cyber-criminals. If a CA can be subverted, then the security of the entire system is lost, potentially subverting all the entities that trust the compromised CA. Each of these CAs have the ability to create alternative identities for the user. Numerous incidents have already shown that there is a risk increase of MITM attacks when the user place too much trust in CAs.

The Internet Engineering Task Force (IETF) responsible for Web PKI standards has created a memo/whitepaper describing current PKI issues [58]. A group of researchers focused on Rebooting the Web of Trust (including Vitalik Buterin, creator of Ethereum) assessed independently its weaknesses in their white paper titled “Decentralized Public Key Infrastructure” [59]. The researchers agreed that the current implementation of Web PKI has problems that should not be ignored.

The white paper outlines the following issues with the current centralized PKI approach:

PKI systems are responsible for the secure delivery of public keys. However, the commonly used X.509 PKI, PKIX, undermines both the creation and the secure delivery of these keys. In X.509 PKIX, web services are secured through the creation of the keys signed by CAs. However, the complexity of generating and managing keys and certificates in PKIX has caused web hosting companies to manage the creation and signing of these keys themselves, rather than leaving it to their clients. This creates major security concerns from the outset, as it results in the accumulation of private keys at a central point of failure (the web hosting company), making it possible for anyone with access to that repository of keys to compromise the security of the connections to those websites in a way that is virtually undetectable [59].

The design of X.509 PKIX also permits any of approximately 1200 CAs around the world to impersonate any website. This is further complicated by the risk of coercion or compromise of a CA. Due to these dangers, users cannot be certain that their communications are not being compromised by a fraudulent certificate allowing a MITM attack. These attacks are extremely difficult to detect. Companies such as Google that produce web browsers can sometimes recognize attacks on their own websites, but they cannot prevent attacks on arbitrary websites [59] (Figure 11)



Fraudulent Certificates Allow Man-in-the-Middle Attacks

Figure 11. Man-in-the-Middle Attack

The white paper also determined that even if third-party authorities could be trusted, the current PKI system has major usability problems. A group from Brigham Young University investigated the usability of [Mailvelope](https://www.mailvelope.com/en) [https://www.mailvelope.com/en], a browser extension that supports GPG-encrypted communication through third-party websites like Gmail. Their research demonstrated a 90% failure rate in secure communication attempts among the participants. Public key management, the study found, was the main reason that users were unable to use the software correctly. Even TextSecure/Signal — a messaging system promoted as being secure - is reported to have usability problems due to its inability to smoothly handle public key changes [59]. If a user deletes and reinstalls the app, their friends are warned that their public key "fingerprint" has changed. This scenario is indistinguishable from a MITM attack, and few users are likely to understand or bother verifying that they received the correct public key [59].

Due to the traditional PKI's usability challenges, much of Web traffic today is unsigned and unencrypted. This is particularly evident on the major social networks. Given PKI's complexity, social networks do not encrypt their user's communications in any way, other than relying on PKIX by sending communications over HTTPS. Because messages are not signed, there is no way to be sure that a user really said what they said, or whether the text displayed is the result of a database compromise. Similarly, user communication is stored in a manner that anyone with access to those databases can read — compromising user privacy and burdening social networks with large liability risks [59].

The out-of-date PKI design poses serious usability and security risks because a single point of failure can be used to open any encrypted online communication. The answer is not to abandon PKI, but to find an alternative that returns control of online identities to the entities they belong to, based on a decentralized architecture. This alternate approach is called **Decentralized Public Key Infrastructure (DPKI)**. By doing so, DPKI addresses many usability and security challenges that

plague traditional PKI. DPKI provides advantages at each stage of the PKI life cycle. This is the topic of next section.

9.3. Decentralized Public Key Infrastructure (DPKI)

In 1995, Phil Zimmerman proposed the encryption program Pretty Good Privacy (PGP) [60], which is a decentralized trust system that was created before blockchain existed. However, PGP has issues with establishing trust relations between all parties. But today, with DLTs technologies, there is no need for the third-parties. The emergence of distributed ledger technology (DLT) and blockchain technology provides the opportunity for fully decentralized identity management. DLTs provide a new approach to building a more competent, secure PKI system. **Decentralized Public Key Infrastructure, or DPKI**, is an alternative approach to designing better PKI systems. In 2015, Allen et al. [59] asserted that unlike the traditional approach, DPKI ensures no single third-party can compromise the integrity and security of the system as a whole. DPKI could have a bigger impact on global cybersecurity and cyber-privacy than the development of the SSL/TLS protocol for encrypted Web traffic (now the largest PKI in the world).

In DPKI, trust is decentralized through the use of technologies that make it possible for geographically and politically disparate entities to reach consensus on the state of a shared database. DPKI is based on Distributed Ledger Technologies (typically blockchain key-value datastores). In a decentralized identity system, entities (discrete identifiable units such as, but not limited to, people, organizations, and things) are free to use any shared root of trust. Globally distributed ledgers, decentralized P2P networks, or other systems with similar capabilities, provide the means for managing a root of trust without introducing a centralized authority or a single point of failure. In combination, DLTs and decentralized identity management systems enable any entity to create and manage their own identifiers on any number of distributed, independent roots of trust [61].

In blockchain-powered DPKI, the new third parties become miners or validators. The trust is established and maintained based on consensus protocols. The miners or validators will have to follow the rules of the protocol, that would financially reward and punish these third-parties to effectively preventing misbehavior in the blockchain and limiting their roles. Bitcoin, devised by Satoshi Nakamoto, is the first such successful protocol. It is based on proof-of-work, in which the energy expenditure of "miners" is used to secure the database. Other consensus protocols could be used such as proof-of-stake (PoS).

Figure 12 illustrates in a nutshell, the difference between PKI and DPKI. In a PKI based system Alice and Bob need to establish a way to exchange and store their public keys. Conversely, in a blockchain-based web of trust model, the storage of public keys is managed on the public ledger. As participants in a global identity network, Alice and Bob create their unique DIDs, attach their public keys and write them to the public ledger. Now any person or organization that can discover these Decentralized Identifiers (DIDs) will be able to acquire access to the associated public keys for verification purposes.

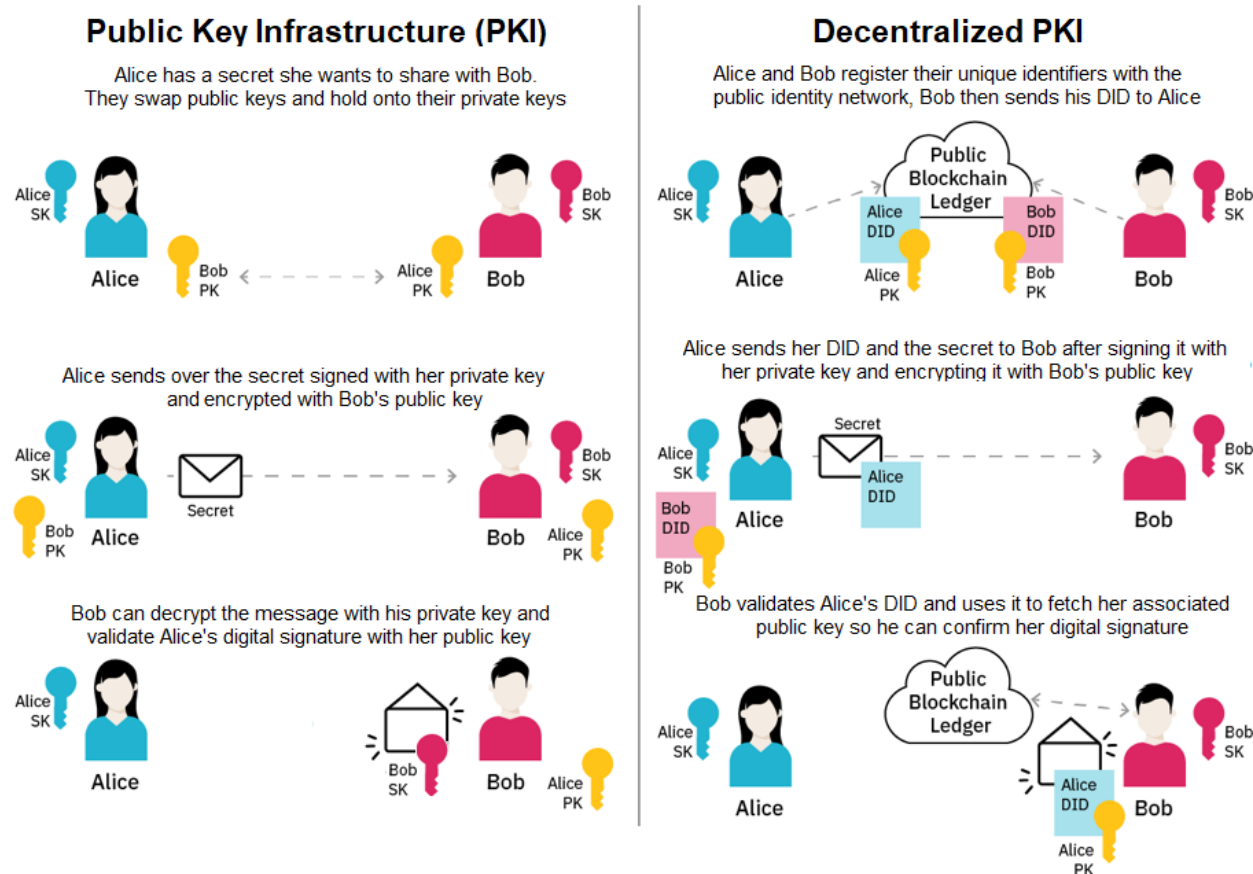


Figure 12. PKI versus DPKI (source: <https://www.ibm.com/blogs/blockchain/2018/06/self-sovereign-identity-why-blockchain/>)

There are two emergent and important standards that play a central role in the enablement of DPKI, the W3C Decentralized Identifiers (DIDs) [61] and the W3c Verifiable Credentials Data Model [62]. The next sections, which are derived from the W3C Primer for Decentralized Identifiers [63], describe these foundational specifications in more detail.

9.3.1. Decentralized Identifiers

The W3C Decentralized Identifier (DID) [61] is a new type of identifier that is globally unique, resolvable with high availability, and cryptographically verifiable. DIDs are typically associated with cryptographic material, such as public keys, and service endpoints, for establishing secure communication channels. DIDs are useful for any application that benefits from self-administered, cryptographically verifiable identifiers such as personal identifiers, organizational identifiers, and identifiers for Internet of Things scenarios. The W3C Verifiable Credentials heavily utilize Decentralized Identifiers to identify people, organizations, and things and to achieve a number of security and privacy-protecting guarantees.

At a high-level DIDs are a new type of globally unique identifier intended for verifiable digital identity that is "self-sovereign" i.e., fully under the control of the identity owner and not dependent on a centralized registry, identity provider, or certificate authority. DIDs are the core component of an entirely new layer of decentralized digital identity and public key infrastructure (PKI) for the Internet. DIDs resolve to DDOs (DID descriptor objects)—simple JSON-LD documents that contain all the metadata needed to prove ownership and control of a DID. Specifically, a DDO contains a set of key descriptions, which are machine-readable descriptions of the identity owner's public keys, and

a set of service endpoints, which are resource pointers necessary to initiate trusted interactions with the identity owner. Each DID uses a specific DID method, defined in a separate DID method specification, to define how a DID is registered, resolved, updated, and revoked on a specific distributed ledger or network [63].

In 2016 the developers of the DID specification agreed with a suggestion from Christopher Allen that DIDs could be adapted to work with multiple blockchains by following the same basic pattern as the URN specification (Figure 13):

DID Syntax

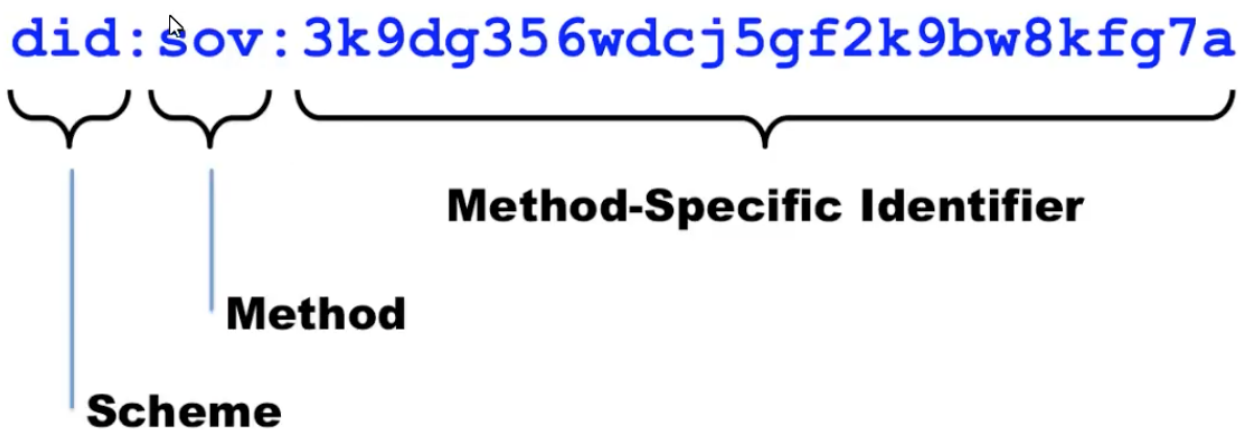


Figure 13. DID syntax

The key difference is that with DIDs the method (or namespace) component identifies a DID method (ipid in Figure 14), and the DID method specification specifies the format of the method-specific identifier.

DID Syntax

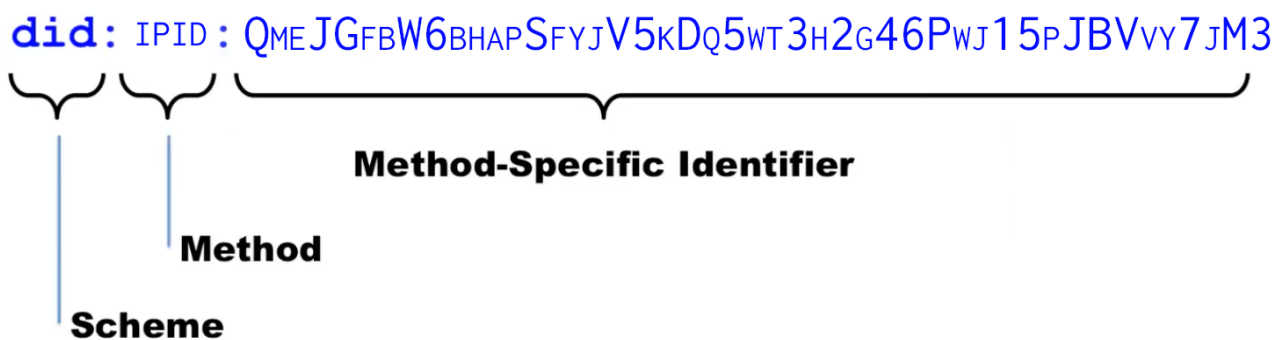


Figure 14. DID using IPID Method

DID Methods

DID methods define how DIDs work with a specific blockchain. All DID method specifications must define the format and generation of the method-specific identifier. Note that the method specific identifier string must be unique in the namespace of that DID method. Defining how a DID and DID document are created, resolved, and managed on a specific blockchain or “target system” is the role of a DID method specification. DID method specifications are to the generic DID specification as URN namespace specifications (UUID, ISBN, OID, LSID, etc.) are to the generic IETF URN specification. [Table 2](#) summarizes the DID method specifications currently in development and defined the DID Method Registry [64].

Table 2. DID method specifications currently in development.

Method Name	Status	DLT or Network
did:abt:	PROVISIONAL	ABT Network
did:btc:	PROVISIONAL	Bitcoin
did:stack:	PROVISIONAL	Bitcoin
did:erc725:	PROVISIONAL	Ethereum
did:example:	PROVISIONAL	DID Specification
did:ipid:	PROVISIONAL	IPFS
did:life:	PROVISIONAL	RChain
did:sov:	PROVISIONAL	Sovrin
did:uport:	DEPRECATED	Ethereum
did:ethr:	PROVISIONAL	Ethereum
did:v1:	PROVISIONAL	Veres
did:com:	PROVISIONAL	commercio.network
did:dom:	PROVISIONAL	Ethereum
did:ont:	PROVISIONAL	Ontology
did:vvo:	PROVISIONAL	Vivvo
did:aergo:	PROVISIONAL	Aergo
did:icon:	PROVISIONAL	ICON
did:iwt:	PROVISIONAL	InfoWallet
did:ockam:	PROVISIONAL	Ockam
did:ala:	PROVISIONAL	Alastria
did:op:	PROVISIONAL	Ocean Protocol
did:jlinc:	PROVISIONAL	JLINC Protocol
did:ion:	PROVISIONAL	Bitcoin
did:jolo:	PROVISIONAL	Ethereum
did:bryk:	PROVISIONAL	bryk
did:peer:	PROVISIONAL	peer
did:selfkey:	PROVISIONAL	Ethereum

Method Name	Status	DLT or Network
did:meta:	PROVISIONAL	Metadium
did:tys:	PROVISIONAL	DID Specification
did:git:	PROVISIONAL	DID Specification
did:tangle:	PROVISIONAL	IOTA Tangle
did:emtrust:	PROVISIONAL	Hyperledger Fabric
did:ttm:	PROVISIONAL	TMChain
did:wlk:	PROVISIONAL	Weelink Network
did:pistis:	PROVISIONAL	Ethereum
did:holo:	PROVISIONAL	Holochain
did:web:	PROVISIONAL	Web
did:io:	PROVISIONAL	IoTeX
did:vaultie:	PROVISIONAL	Ethereum
did:moac:	PROVISIONAL	MOAC
did:omn:	PROVISIONAL	OmniOne
did:work:	PROVISIONAL	Hyperledger Fabric
did:vid:	PROVISIONAL	VP
did:ccp:	PROVISIONAL	Quorum
did:jnctn:	PROVISIONAL	Jnctn Network

DID Documents

DID infrastructure can be thought of as a global key-value database in which the database is all DID-compatible blockchains, distributed ledgers, or decentralized networks. DIDs and DID documents can be adapted to any modern blockchain, distributed ledger, or other decentralized network capable of resolving a unique key into a unique value. It does not matter whether the blockchain is public, private, permissionless, or permissioned. In this virtual database, the key is a DID, and the value is a DID document. The purpose of the DID document is to describe the public keys, authentication protocols, and service endpoints necessary to bootstrap cryptographically-verifiable interactions with the identified entity [63].

A DID document is a valid JSON-LD object that uses the DID context – the vocabulary of property names that is encoded in Resource Description Framework (RDF) - defined in the DID specification. This includes six components (all optional):

1. **The DID itself**, so the DID document is fully self-describing.
2. **A set of cryptographic material**, such as public keys, that can be used for authentication or interaction with the DID subject.
3. **A set of cryptographic protocols** for interacting with the DID subject, such as authentication and capability delegation.
4. **A set of service endpoints** that describe where and how to interact with the DID subject.

5. **Timestamps** for auditing.
6. **An optional JSON-LD signature** if needed to verify the integrity of the DID document.

The following shows a sample DID Document.

Sample DDO stored using did method spec stored on IPFS

```
{
"@context": "/ipfs/QmfS56jDfrXNaS6Xcsp3RjiXd2wyY7smeEAwyTAnL1RhEG",
"id": "did:ipid:QmeJGfbW6bhapSfyjV5kDq5wt3h2g46Pwj15pJBVvy7jm3",
"owner": [{
  "id": "did:ipid:QmeJGfbW6bhapSfyjV5kDq5wt3h2g46Pwj15pJBVvy7jm3",
  "type": ["CryptographicKey", "EdDsaPublicKey"],
  "curve": "ed25519",
  "expires": "2017-02-08T16:02:20Z",
  "publicKeyBase64": "lji9qTtkCydxtex/bt1zdLxVMMbz4SzWvlqgOBmURoM="
}, {
  "id": "did:ipid:QmeJGfbW6bhapSfyjV5kDq5wt3h2g46Pwj15pJBVvy7jm3/key-2",
  "type": ["CryptographicKey", "RsaPublicKey"],
  "expires": "2017-03-22T00:00:00Z",
  "publicKeyPem": "-----BEGIN PUBLIC KEY-----\r\nMIIBOgIBAAJBAAkSUT9/Q2uBfGRau6/XJyZhcF5abo7b37I5hr3EmwGykdyk8GSyJK3T0rjyl0sdJsGbFmgQaRyV\r\n-----END PUBLIC KEY-----"
}],
"control": [{
  "type": "OrControl",
  "signer": [ "did:eth:0xd3382e07f2173270ef43817ab1b4e1cdeb36f23b",
"did:sov:8uQhQMGzWxR8vw5P3UWH1j" ]
}],
"service": {
  "did": "did:eth:0x641073322a9aa53fcf025587f86226fe358da1ef2c2e4dcb989d610e9dbf6b9a",
},
"created": "2017-09-24T17:00:00Z",
"updated": "2018-09-24T02:41:00Z",
"signature": {
  "type": "RsaSignature2016",
  "created": "2016-02-08T16:02:20Z",
  "creator": "did:ipid:QmeJGfbW6bhapSfyjV5kDq5wt3h2g46Pwj15pJBVvy7jm3",
  "signatureValue":
"IOmA4R7TfhkYTYW87z64003GYFlDw0yqie9Wl1kZ50BYNAK0wG5u0sPRK8/2C4STOWF+83cMcbZ3CBMq2/gi25s="
}
}
```

DIDs and Privacy by Design

Privacy is an essential component of any identity management solution. Privacy is especially critical for a global identity system that uses immutable public blockchains. Thankfully DID architecture can incorporate Privacy by Design at the very lowest levels of infrastructure and thus become a powerful, new, privacy-preserving technology if deployed using best practices such as [63]:

- **Pairwise-pseudonymous DIDs.** While DIDs can be used as well-known public identifiers, they can also be used as private identifiers issued on a per-relationship basis. So rather than a person having a single DID, like a cell phone number or national ID number, they can have thousands of pairwise-unique DIDs that cannot be correlated without their consent, yet can still be managed as easily as an address book.
- **Off-chain private data** Storing any type of PII on a public blockchain, even encrypted or hashed, is dangerous for two reasons: 1) the encrypted or hashed data is a global correlation point when the data is shared with multiple parties, and 2) if the encryption is eventually broken (e.g., quantum computing), the data will be forever accessible on an immutable public ledger. So, the best practice is to store all private data off-chain and exchange it only over encrypted, private, peer-to-peer connections.
- **Selective disclosure.** The decentralized PKI (DPKI) that DIDs make possible opens the door to individuals gaining greater control over their personal data in two ways. First, it enables data to be shared using encrypted digital credentials (see below). Second, these credentials can use zero-knowledge proof cryptography for data minimization. For example, you can disclose that you are over a certain age without disclosing your exact birthdate.

DIDs are only the base layer of decentralized identity infrastructure. The next higher layer – where most of the value is unlocked – is **Verifiable Credentials**. DIDs can be used to identify various entities in the Verifiable Credentials ecosystem such as issuers, holders, subjects, and verifiers. More generally, DIDs can be used as identifiers for people, devices, and organizations.

9.3.2. Verifiable Credentials

Currently transmitting over the internet credentials such as driver's licenses, proofs of age, certification of authenticity of physical or digital assets, education qualifications, and healthcare data in a way that is verifiable yet protects individual privacy is difficult. These credentials are composed of statements called **verifiable claims**.

Starting in 2013, the W3C Credentials Community Group started to work on solutions for this problem space followed shortly thereafter by the Rebooting Web of Trust Community and W3C Verifiable Claims Working Group. These groups, composed of 150+ individuals and organizations, are currently focused on the creation, storage, transmission, and verification of digital credentials via the Internet. The Verifiable Claims Working Group at the W3C is developing standards for expressing and exchanging "claims" that have been verified by a third party and to make them easier and more secure on the Web. The Verifiable Credentials Data Model 1.0 specification [62], which is now a W3C Recommendation, provides a mechanism to express these sorts of credentials on the Web in a way that is cryptographically secure, privacy respecting, and machine-verifiable.

A **verifiable claim** is a qualification, achievement, quality, or piece of information about an entity's background such as a name, government ID, payment provider, home address, or university degree. Such a claim describes a quality or qualities, property or properties of an entity which establish its existence and uniqueness [62].

9.3.3. Verifiable Credentials Ecosystem

The Verifiable Credentials ecosystem is composed of four primary roles:

- The **Issuer**, who issues verifiable credentials about a specific **Subject**.
- The **Holder** stores credentials on behalf of a **Subject**. **Holders** are typically also the **Subject** of a credential.
- The **Verifier** requests a profile of the **Subject**. A **profile** contains a specific set of credentials. The **verifier** verifies that the credentials provided in the profile are fit-for-purpose.
- The **Identifier Registry** is a mechanism that is used to issue identifiers for **Subjects**.

A visual representation of the ecosystem is shown in [Figure 15](#).

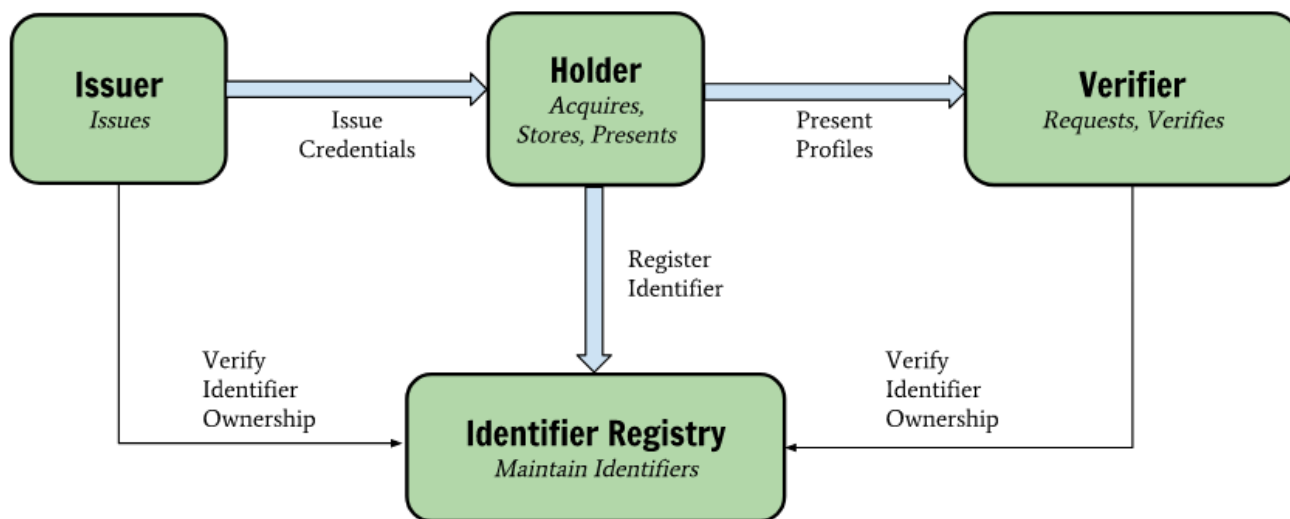


Figure 15. Verifiable Credentials ecosystem

Claims, Credentials, and Profiles

Credentials Data exchanged between parties differs based on the roles participating, but is fundamentally composed of **Claims**, **Credentials**, and **Profiles**. The encoding of the credential data information is based on Linked Data standards, in particular JSON-LD. JSON-LD is a JSON-based format used to serialize Linked Data. The syntax is designed to easily integrate into deployed systems already using JSON, and provides a smooth upgrade path from JSON to JSON-LD. It is primarily intended to be a way to use Linked Data in Web-based programming environments, to build interoperable Web services, and to store Linked Data in JSON-based storage engines. Linked Data standards are based on the RDF model, which allows the expression of any piece of information in form of triples (also called statement) consisting of (subject predicate object) and the processing by machine.

The following descriptions are a high-level introduction to the data model and gloss over specifics. Readers that would like to explore the data model in more depth are urged to read the Verifiable Claims Working Groups' Data Model Specification [62].

A **claim** is statement about a subject, expressed as a subject-property-value relationship [Figure 16](#):

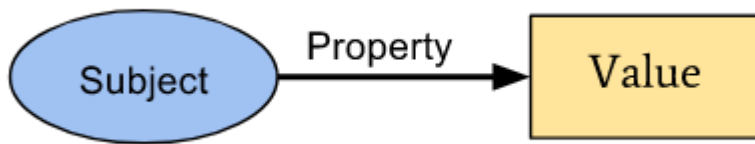


Figure 16. Claim Model

The Linked Data model used to describe claims allows the expression of a large variety of statements that can be grounded in ontologies using the Web Ontology Language (OWL) or RDF Schema. For example, whether or not someone is over the age of 21 may be expressed as follows [Figure 17](#):

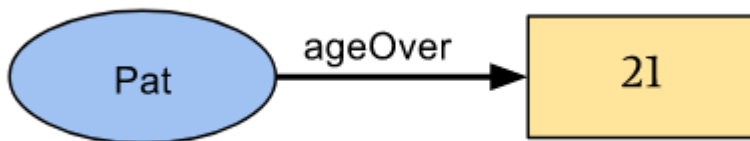


Figure 17. Claim example 1

These claims may be merged together to express a Linked Data graph of information about a particular subject. The example below extends the data model above by adding claims that state that Pat knows Sam and that Sam is a student [Figure 18](#):



Figure 18. Claim graph

When an Issuer sends data to a Holder, it bundles a set of claims into a data structure called a **credential** and digitally signs the data structure ([Figure 19](#)). A credential is a set of one or more claims made by the same entity. Credentials might also include an identifier and metadata to describe properties of the credential, such as the issuer, the expiry date and time, a representative image, a public key to use for verification purposes, the revocation mechanism, and so on. The metadata might be signed by the issuer. A verifiable credential is a set of tamper-evident claims and metadata that cryptographically prove who issued it.

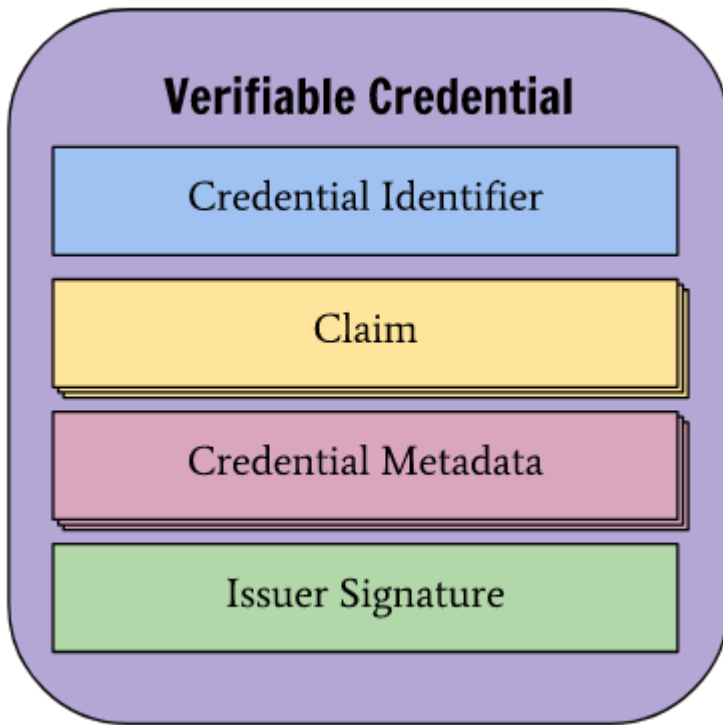


Figure 19. Verifiable Credential Data Structure

When a Verifier asks for data from a Holder, the Holder typically bundles a set of credentials into a data structure called a Presentation and digitally signs the data structure (Figure 20): A verifiable presentation expresses data from one or more verifiable credentials, and is packaged in such a way that the authorship of the data is verifiable. If verifiable credentials are presented directly, they become verifiable presentations. Data formats derived from verifiable credentials that are cryptographically verifiable, but do not of themselves contain verifiable credentials, might also be verifiable presentations.

The data in a presentation is often about the same subject, but might have been issued by multiple issuers. The aggregation of this information typically expresses an aspect of a person, organization, or entity.

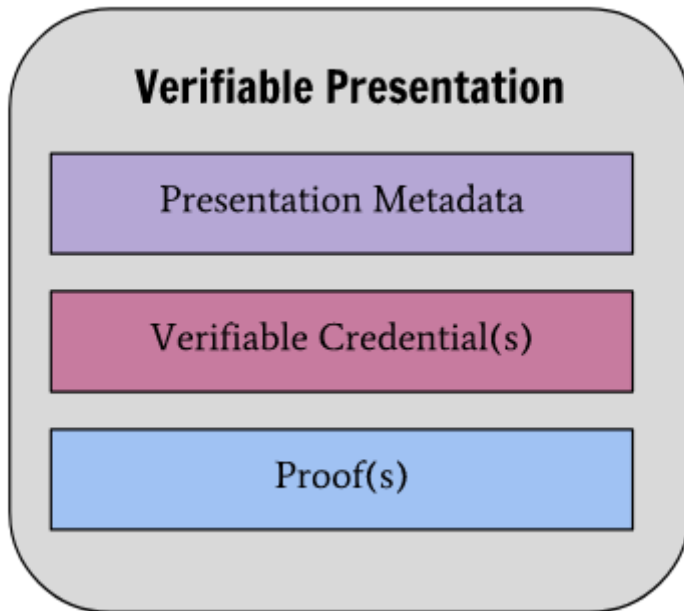


Figure 20. Verifiable Presentation

Figure 21 below shows a more complete depiction of a verifiable presentation, which is normally composed of at least four information graphs. The first graph expresses the verifiable presentation itself, which contains presentation metadata. The verifiable Presentation property in the graph refers to one or more verifiable credentials (each a self-contained graph), which in turn contains credential metadata and claims. The third graph expresses the credential graph proof, which is usually a digital signature. The fourth graph expresses the presentation graph proof, which is usually a digital signature.



Figure 21. Verifiable Presentation

In the example below, Pat receives an alumni verifiable credential from a university, and Pat stores the verifiable credential in a digital wallet.

A simple example of a verifiable credential

```
{
  "@context": [
    "https://www.w3.org/2018/credentials/v1",
    "https://www.w3.org/2018/credentials/examples/v1"
  ],
  "id": "http://example.edu/credentials/1872",
  "type": ["VerifiableCredential", "AlumniCredential"],
  "issuer": "https://example.edu/issuers/565049",
  "issuanceDate": "2010-01-01T19:73:24Z",
  "credentialSubject": {
    "id": "did:example:ebfeb1f712ebc6f1c276e12ec21",
    "alumniOf": {
      "id": "did:example:c276e12ec21ebfeb1f712ebc6f1",
      "name": [{
        "value": "Example University",
        "lang": "en"
      }, {
        "value": "Exemple d'Université",
        "lang": "fr"
      }]
    }
  },
  "proof": {
    "type": "RsaSignature2018",
    "created": "2017-06-18T21:19:10Z",
    "proofPurpose": "assertionMethod",
    "verificationMethod": "https://example.edu/issuers/keys/1",
    "jws": "eyJhbGciOiJIUzI1NiIsImI2NCI6ZmFsc2UsImNyaXQiOlsiYjY0Ii19..TCYt5XsITJX1CxPCT8yAV-TVkIEq_PbChOMqsLfRoPsnsgw5WEuts01mq-pQy7UJiN5mgRxD-WUcX16dUEMG1v50aqzpqh4Qktb3rk-BuQy72IFLOqV0G_zS245-kronKb78cPN25DG1cTwLtjPAYuNzVBAh4vGHSrQyHUdBBPM"
  }
}
```

9.3.4. DPKI solutions

This section provides a review of existing tools that support DPKI with the DID and Verifiable Credentials.

[Hyperledger Indy](https://www.hyperledger.org/projects/hyperledger-indy) [https://www.hyperledger.org/projects/hyperledger-indy] provides the tools, libraries and reusable components for providing digital identities rooted in blockchains or other distributed ledgers so that they are interoperable across administrative domains, applications and any other silo. The Sovrin network is a deployment of Hyperledger Indy that is compatible with any Hyperledger Aries identity agent.

[Hyperledger Aries](https://www.hyperledger.org/projects/aries) [https://www.hyperledger.org/projects/aries] provides a shared, reusable, interoperable tool kit designed for initiatives and solutions focused on creating, transmitting and storing verifiable digital credentials. Aries provides infrastructure for blockchain-rooted, peer-to-peer interactions. It includes a shared cryptographic wallet for blockchain clients as well as a communications protocol for allowing off-ledger interaction between those clients. This project consumes the cryptographic support provided by [Hyperledger Ursa](https://www.hyperledger.org/projects/ursa) [https://www.hyperledger.org/projects/ursa], to provide secure secret management and decentralized key management functionality.

Chapter 10. Solutions

This section outlines the important components, technologies and standards needed to build a robust, scalable, secure, trustworthy federal cloud analytic solution with secure provenance information about workflow.

Today, the Internet is composed of a network of interconnected entities, traditionally referring to human users and computers. With the emergence of IoT, the number of addressable connected entities and devices is in the tens of billions, with an estimate of 75 billions connected IoT devices in 2025 [65]. By adding furthermore cloud computing resources such as complex workflow composed of software services running different algorithms in geographic distributed computing environment, this broaden even further the number of identifiable entities to manage and track.

The seamless provenance of physical assets or data assets through any value supply chain has major implications on the risk and value properties of the processed data. The participating entities in such dynamic workflow process have substantial interest in the authenticity, traceability and trustworthiness in any individual step of these complex workflows.

Currently most of the data integration and analysis is handled predominantly in the cloud. However, the exponential growth of digital data due to the proliferation of digital devices requires also an exponential increase in computing capacity for data integration and analysis. The reliance of transporting data back and forth between 'the edge' of the Internet, where data is produced, and remote cloud data centers is not efficient. It is often more efficient to process the data on the edge (known as fog computing) in a hierarchical-tree-like bottom-up fashion by benefiting the low-latency processing of data close to the sources and sinks of the data. The goal of the proposed solution is to enable a large number of entities to cooperatively participate both as consumers and producers of information in a federation of computing processes executing complex workflows where each step can be trusted, audited and reproduced.

To be able to accomplish this vision, a **robust decentralized infrastructure where information about entities and data can be retrieved, trusted and verified anytime and anywhere** is needed. This is where DLTs and Blockchain technologies come in. The highest purpose of the blockchain can be seen as a kind of "truth machine" [66] as it relies on highly decentralized networks of information and validation.

10.1. Challenges

10.1.1. DLT and Blockchain heterogeneity

One of the challenges to overcome is the fact that are many DLTs and Blockchains that can be used as *root of trust*. Some are globally distributed ledgers such as Bitcoin, Ethereum, some DLTs are permissioned with controlled access, and others are very specialized blockchain designed for a specific domain space (supply-chain, advertisement, reputation, etc.). Implemented in a proprietary manner, DLT behaves similar to a traditional database where one entity retains pull-the-plug and change-the-rules authority. This model still has the fundamental problems of siloed identity, which created the identity mess in the first place as described in [Identity Management](#) section.

10.1.2. Location-based Addressing

Another challenge is the assignment of **decentralized stable and globally unique identifiers** to refer to any entities involved in a complex federated workflow including persons, entities, transactions, data, physical objects. The current Web architecture is based on a location-based addressing mechanism based on HTTP protocol (URLs). When a web page, dataset, service is accessed, it is identified by a HTTP link that points to a particular location on the web, corresponding to a server or cluster of servers somewhere on the web. This has a number of implications.

- Whoever controls that location controls the content. This puts a lot of responsibilities and power on the shoulders of whoever controls the location. Among the responsibilities of the content provider includes the need to remain accessible 24/7, accommodate the workload for large number of users, protect the integrity of the information by preventing anyone tampering with it, never remove the information from the server location - if they destroy it or even move it, the link is broken and no one can access the information.
- The owners of the server have the power to dictate who is allowed to see information, move the content without telling anyone, destroy the content, charge people money to access information, collect data about everyone who access the information, replace information with something else or something more malicious, turning location to a trap.
- Even if thousands of copies of a file have been downloaded by users, HTTP points to a single location. A location-addressed approach forces users to pretend that the data are in only one location, even the content has been duplicated in many computers that may be located closer to them.

Location-addressing has worked on the web for 25 years, but it's starting to get painful and it's about to get much worse. As long as we continue to rely on it, the web will continue to be unstable, insecure, and prone to manipulation or exploitation. It is time for a distributed, permanent web. One solution to this problem is to use **Content Addressable Content**. **Content-addressable storage**, also referred to as associative storage or abbreviated **CAS**, is a mechanism for storing information that can be retrieved based on its content, not its storage location. It is typically used for high-speed storage and retrieval of fixed content (e.g. BTFS, IPFS), such as documents stored for compliance with government regulations. Roughly speaking, content-addressable storage is the permanent-storage analogue to content-addressable memory.

10.1.3. Lack of Standard Provenance for Workflow

Data provenance plays a big part in scientific workflows. It documents the execution of the workflow and stores the information about the data pieces involved in the workflow. Current workflow management systems do not have a standard mechanism to automatically share provenance information. The ability of sharing provenance will greatly reduce duplication of workflows, improve the trust and integrity of data and analyses, improve reproducibility of scientific workflows and catalyze the discovery of new knowledge. The issue is that there are so many ways to describe provenance depending on the platform of execution (cloud workflow, web service workflow), that it makes it difficult to come out with a universal model that accommodates all the specifics of each system. PROV-O provides a core minimum ontology composed of three concepts (Agent, Activity, Entity). These core concepts need to be extended and adapted for each target workflow management system (for example CWLProv for Federated Cloud Workflow). These standards keep evolving over time and it will be hard to reach any consensus anytime soon. Provenance information become less

important if you can provide a solution that guarantees reliable trust verification for all the artifacts produced by workflow using a Web of Trust infrastructure. Verifiable claims about entities produced by workflows could include reference to provenance information encoded in the most adequate standard.

To have a truly global solution, easy to use and still safe from hacking and sovereign interference, the solution must address the following characteristics:

- preservation of privacy
- security from tampering
- reliable trust verification
- assurance of risk
- independence from any vendor-defined naming API
- one-to-one mappable onto each entity.

10.2. Siloed Trust System

The number of possible connections between any given set of entities in the cloud is an impossibly large number. Human, machine, or software agents increasingly need to interact in a trusted manner with a different group of these interconnected objects to achieve their goals in both the digital and physical worlds. This requires a straightforward and ubiquitous method to address, verify, and connect these elements together. Unfortunately, the current centralized PKI approach is not able to scale with the growth of IoT and has severe usability and security limitations. The Internet does not have a built-layer for managing identity and trust in decentralized way. Human or object identities are stored in multiple centralized or federated systems such as government, ERP, IoT, or manufacturing systems. From a cryptographic trust verification standpoint, each of these centralized authorities serves as its own root of trust. An entity trailing along a value chain is interacting with multiple systems. Consequently, a new actor in any given value chain has no method to independently verify credentials of a human or attributes of either a physical object or data item (provenance, audit trail). This results in the existence of complex validation, quality inspection, and paper trail processes, and enormous hidden trust overhead costs are added to all value chains and services [67].

10.3. High level description of the solution

The solution outlined in this section describes the general principles that needs to be adhered to accomplish the goal of establishing a web of trust. The solution is not exhaustive and complete as some important standards still need to be defined by the community.

The proposed solution to address the challenges outlined in the previous section is based on DLTs and a new layer of decentralized identity infrastructure, which provides a straightforward, ubiquitous, universal method and associated interoperable protocols to address, verify, and connect all the entities participating in complex workflow ecosystems. Entities include living organisms, physical objects, locations, events, machines, IoT devices, digital assets, datasets or agents. The solution is based on the emerging Decentralized identifier (DID) standard supported by W3C and the Decentralized Identity Foundation (DIF), which defines a new and open standard type of globally unique identifier that offers a model for lifetime-scope portable digital identity that does not depend on any centralized authority

and that can never be taken away by third-parties [61]. The core idea of the approach is to use DIDs to identify *every entity* involved in the complex workflows. DIDs will identify and manage objects, machines, or agents through their digital twins and will be expanded to digital assets, locations, and to events. Data assets are referred to as **Decentralized Autonomous Data (DAD)** items [68],[67].

The objective of the suggested approach is to capture unforgeable audit trail referring to the DIDs of all the entities involved in each step of the processing workflow. The data flow provenance is captured via a self-referential blockchain, which consists of a list of signed DAD items where each subsequent DAD item includes the DID(s) or dDID and associated signature of the previous item, forming a Merkle tree.

DID can utilize the W3C Verifiable Credentials standard to prove verified aspects of their identity when signing claims about other DIDs. Such a use case could enable verified organizations to sign a claim with their DID that they have produced a digital asset according given specifications, certify software, service, hardware and standard-based provenance document that could be reused to reproduce the data asset.

By integrating decentralized identifiers, verifiable claims, DLTs, federated analytics can be achieved in a fully decentralized, peer-to-peer collaboration mode, without including a third party to coordinate the process. Everything is trusted, transparent and traceable and organizations can now control their own digital identities, rather than relegating control to their vendors. This increases security, accuracy, and trust for all participants in the global supply chain.

10.4. Identity Management for Federated Cloud

A paradigm shift is now occurring with the exponential growth of Big Data due to the rise of the Internet of Things (IoT), social media, mobility, and other data sources, as it defies the scalability of centralized approaches to store and analyze data in a single location. Participants in complex federated analytics workflows may include, not only different organizations, but also devices from the Internet of Things (IOT). To be able to trust each participant of the workflow, it is necessary to have a scalable, secure and robust identity management solution. As discussed in the [Identity Management](#) section, centralized identification systems based on PKI suffer from serious security and usability challenges. Testbed-15 proposes a solution to address this challenge is to use a DPKI infrastructure leveraging DLTs and Blockchain to build a Web of Trust without centralized root of trust. To bootstrap the web of trust between participants, each participant and artifacts produced by the workflow should be assigned with a resolvable, permanent decentralized identifier using the DID. The DID would be registered on a blockchain using a method specification that allows the storage of the DID document on the DLTs. For example, IPFS can be used as a method to store the document in a distributed way. The access of the document would be resolvable anytime from anywhere due to the replication of the document over the P2P network. The DID of each participant would be used to publish and sign verifiable claims using the W3C Verifiable Credentials standards.

Within the solution describes above, DIDs would also be used to reference artifacts and provenance information produced by federated analytic workflows. By doing so, they can be referred to in verifiable claims that can provide information about the author, provenance information, time stamps and certification information. The credential data model authors are expected to use machine-readable vocabularies through the use of Linked data ontologies (for example schema.org) that the different parties of the federated cloud agree upon.

10.5. Universal Resolver for Self-Sovereign Identifiers

A **DID resolver** is a software or hardware component with an API for resolving DIDs of at least one DID method. It executes the read operation for the DID method corresponding to the DID being resolved to obtain the authoritative DID document [61].

The **DIF Universal Resolver** supports different “methods” for registering identifiers on decentralized systems such as the Bitcoin blockchain, Sovrin, Ethereum, IPFS, and others. It provides a unified interface which can be used to resolve any kind of decentralized identifier. This enables higher level data formats (such as Verifiable Claims) and protocols (such as DIF’s Hub protocol) to be built on top of the identifier layer, no matter which DLT, blockchain or other system has been used to register the identifier. Internally, this is achieved by the use of “drivers” for each supported identifier type. A driver can be easily added via a Docker container, a Java API, or a remote HTTP GET call, as illustrated in the following diagram [Figure 22](#):

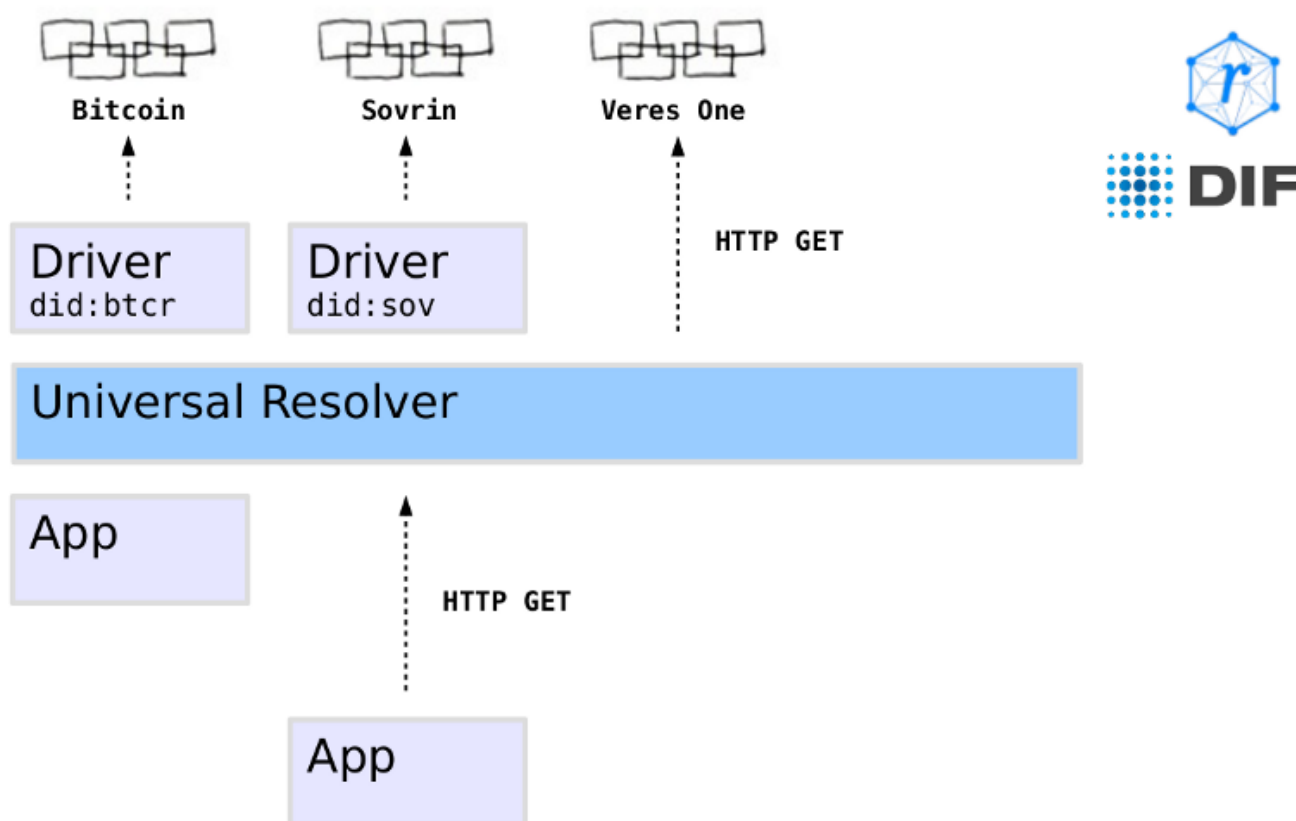


Figure 22. DIF Universal Resolver

Currently a number of driver specification documents have been defined for Sovrin, Bitcoin, Blockstack, uPort, Kolocom, IPFS, IPNS and Veres One Distributed Ledgers. Drivers can internally function in different ways — some drivers may simply call a remote web service (can make the driver usable even on mobile devices), while others may have direct access to a full node of the blockchain they use which provides higher security guarantees for the resolution result.

10.6. Chaining DADs

Shawn Conway introduced in his seminal paper "A DID for Everything" [67] a mechanism to capture attribution, verification and provenance for entities and Data Items using DADs. "The provenance of data in a data flow through a processing system that transforms data can be established by forming a literal block chain of the data. When using DAD items to represent the data, the chain of DADs can be represented simply in a self-contained manner. At each step in the data flow of the originating DAD, where the contained data is transformed in any way, a new DAD is generated by the controlling entity of the transformation. This entity assigns a new DID or derived DID (dDID) to this DAD. The new DAD payload includes the DID(s) of the DAD prior to transformation as well as the signature of the prior DAD. This links the new DAD to the prior DADs. The signature of the prior DAD provides both a hash that establishes the content integrity of the prior DAD as well a non-repudiation of the controller of the embedded prior DID. All the DADs in the data flow need to be stored some place indexed by their DIDs. Given this storage, any single DAD can then be used to recall the string of prior DADs back to the originating DAD or DADs. A special case is when an entity merely wishes to establish custody of data without changing or transforming it. The simplest way to do this, is for the entity to add a copy of the DAD as a link in the DAD chain without changing the underlying data. This forms an assertion that they control that link. If they do not transform the data then merely signing is enough to assert control over the link or equivalent custody of the the data for that link of the chain. A one to many data stream is just a branch or fork in the chain into multiple chains." [67].

Figure 23 illustrates a complex Directed Acyclic Graph (DAG) Decentralized Autonomous Data Flow.

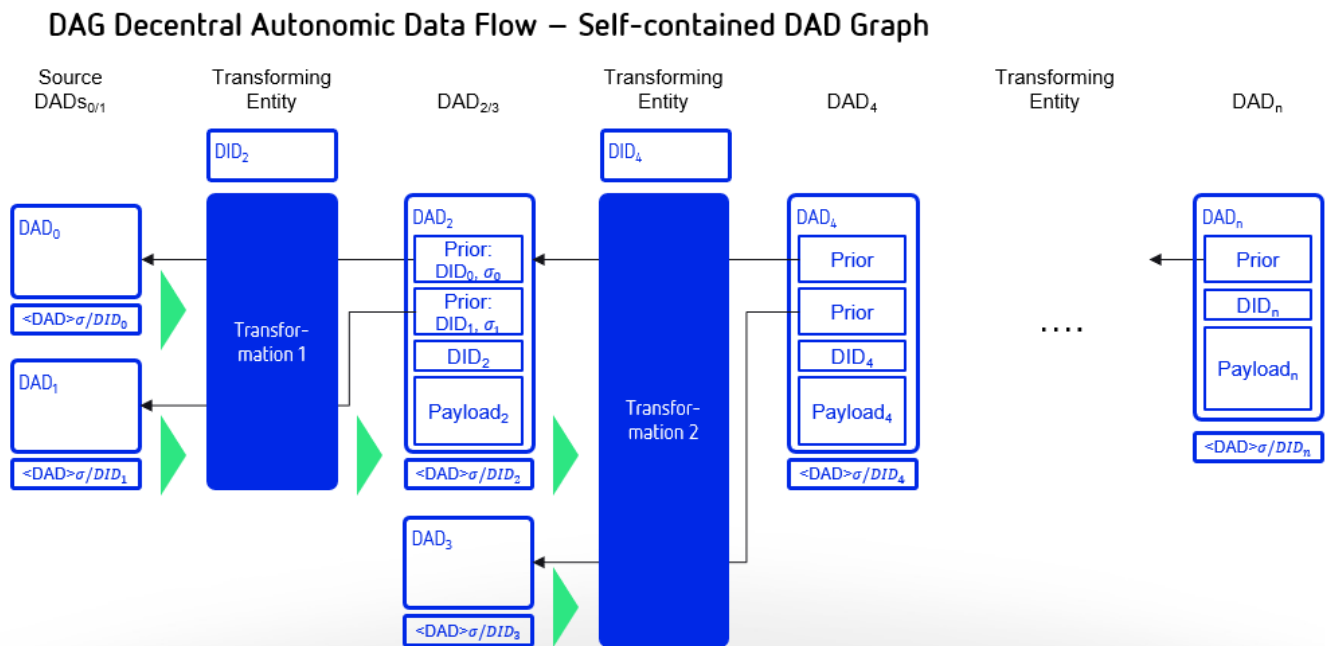


Figure 23. DAG Data Flow - Self-contained DAG Graph.

Provenance of both the controller of the transformation step and the integrity of the associated data can be determined because each DAD embeds a DID and is signed by the associated private key belonging to the DID. Full traceability can be established back to the originating DAD or DADs, preserving both data integrity and proof of control because each DAD embeds a DID and is signed by the associated private key belonging to the DID. This is a critically enabling capability for decentralized computing infrastructure [67].

10.7. Content Availability

The data provenance and its corresponding source data might be critical; therefore, it must be **available at anytime from anywhere**.

Why is this important in the context of this study? One of the main reasons is to have robust and permanent **availability** of provenance information. IPFS is an example of Content-addressable storage that could be used for storing such information. The content could be encrypted to control the access of the information when sensitive information needs to be protected. As long as the data have been duplicated on other peers of the network, the information will remain permanently on the network and can be accessed anytime by using its content hash address.

10.8. Verifiable Credentials for Federated Cloud Analytics

The W3C published a number of use cases for Verifiable Credentials [69] related to education, retail, finance, healthcare, professional credentials, legal identity and devices. However, there is no use case related to supply chain of physical and digital assets. Each variable credential is composed of a series of claims (i.e., RDF statements) asserted by an authoritative entity (the **Issuer**) about another entity (the **Subject**), owned by yet another entity (the **Holder** who most of the time is identical to the Subject but not always) and verified by a final entity (the **Inspector/Verifier**). Using the W3C Verifiable Credentials standards, verifiable claims can be either self-issued by an entity such as a machine to provide a proof about authenticity and integrity of data or they can be issued by a third party. Verifiable Credentials will automate authentication and authorization for consumers and consequently reduce the cost of service delivery for providers. Verifiable Credentials are claims made against an identity (DID) representing an asset such as an algorithm, machine, certificate of authenticity, provenance documentation, etc. that can be verified by a third party verifier. These claims are cryptographically secure, privacy-respecting and machine-verifiable.

In federated cloud systems, any node might want to transact with any other node, engaging with each other in a dynamically defined, on-demand way. To ensure efficient transactions any new entity involved in a value chain must be able to independently verify other counter parties. This is achieved by using DIDs which anchor the verifiable claims on a distributed ledger technology, so they can move the cryptographic root of trust from a centralized to a decentralized, interoperable infrastructure.

Because the claims are expressed in Linked Data, parties sharing credentials need to have a common understanding of the vocabulary used to make the claims. The Verifiable Claim needs to refer to a schema (ontology) that is agreed by different parties. Example of ontology is schema.org or OpenBadge, which could be used to make assertions about people and other web assets. One important aspect of relying on schemas to provide the semantic meaning of data within a verifiable credential, is that the meaning of the credential properties should not change. It is not enough for entities within the ecosystem to have a shared understanding of the data in the present, it may be necessary for them to have an understanding of the credential at the time it was issued and signed. This depends on the trust framework within which the credential was issued and the needs of the parties involved. A verifiable data registry can provide immutable storage of schemas.

10.9. What is stored in the Blockchain ?

The storage of information in a blockchain network is an important issue to be considered from an identity management, asset management, scalability, privacy, security and cost. There are concerns about storing private data (even encrypted), and there is a need to avoid potential security and privacy problems due to possible vulnerability to advanced quantum machines in the future. However, many of the available blockchain platforms do not require private data storing on a blockchain. Another concern is that some private data security approaches might compromise the scalability of the blockchain network. The preferred solution is therefore to use blockchain to store and lookup decentralized identifiers (or their hashes) of the assets, organizations, agents, services, people, and consent of data sharing between the asset owners. Since the proofs of data are stored on the blockchain rather than the identity/asset data themselves, the scalability, cost and storage size are not operational challenges. As a result, private data of any kind (including hashed personal data) and private proof of existence are generally speaking not stored on the ledger.

Chapter 11. Conclusions

Self Sovereign Identifiers and Verifiable credentials are still young technologies. SSI brings back full control of the identity to the owner and the use of DLTs and Blockchain to support Decentralized PKI provides a solid alternative that addresses the usability and security issues of the centralized PKI approach. SSI are fundamental assets for interaction over the Internet and are the cornerstone of establishing the Web Of Trust.

Verifiable credentials provide significant advantages compared to existing hub-and-spoke federation models and data silos as they are not limited to specific data types, mechanisms, or contracts imposed by a central hub. Any participant of this Web of Trust can present identity information of any type to anyone else in the world, and the recipient can unpack and verify it instantly, with no need for a multitude of complex APIs and commercial contracts. No private information is ever stored on the ledger, in any form as the Verifiable claims, along with all private data, are stored off-ledger by each self-sovereign identity owner, wherever the owner decides. The use of SSIs and verifiable claims enables highly advanced privacy-enhancing techniques, such as zero-knowledge proofs (for selective disclosure) and anonymous revocation, to be made available to the world.

The W3C standard is just a beginning to address the challenges of digital credentials. The data model is now a W3C recommendation, but the actual format of digital files is not yet standardized. Interoperability is not as good as it should be, yet, because it permits divergence on some crucial aspects that need to be addressed. Verification processes need to go beyond digital signatures to complex criteria. Rich schemas, which define common semantics for representing claims in credentials is currently under work and need elaboration. A full threat model for credentials is currently under discussion, and different approaches to a robust privacy strategy that can comply with regulatory requirements across legal jurisdictions, and avoid abuse in the surveillance economy, are competing for standardization.

Implemented in a proprietary manner, DLT behaves similar to a traditional database where one entity retains pull-the-plug and change-the-rules authority. This model still has the fundamental problems of siloed identity, which created the identity mess in the first place. Implemented properly, by defining a standard identity layer for Internet, DLTs can remove reliance on centralized silos, enabling the revolutionary power of self-sovereignty. Any person, organization, or thing can actually own their digital identity and control it independently from any silo. Any person, organization, or thing can instantly verify the authenticity of “claims,” including who (or what) something claims to be. Any person, organization, or thing can have complete control of how, what and when information is shared, without the added risk of correlation and without creating troves of breachable data.

11.1. Future Works

There are several projects currently ongoing to explore Earth Observation (EO) data provenance and traceability using blockchain technology. [One of them](https://guardtime.com/blog/european-space-agency-selects-guardtime-for-data-provenance) [https://guardtime.com/blog/european-space-agency-selects-guardtime-for-data-provenance], which implemented as a proof-of-concept, has been conceived at the European Space Agency (ESA) and aims to develop and prototype a set of new technologies to enable secured and traceable exploitation of data from space missions focusing on the EO segment. The high-level objectives of the ESA data provenance project included:

- demonstration of the use of blockchain technology for verification of integrity and time of EO data

products and their provenance throughout the supply chain,

- demonstration of the software compatibility with a variety of EO missions and data formats,
- demonstration of the interoperability of cryptographic proofs extracted and transported for data provenance verification and amending,
- demonstration of deployment based on Copernicus data architecture.

The technical implementation of the ESA data provenance project consisted of several steps: a survey of state-of-the-art DLT technology and analysis of potential requirements; design and development of trusted data sharing processes for the supply chain including necessary software components; integration of the developed software onto the selected data acquisition and distribution infrastructures; and validation and demonstration on identified use cases. The implemented solution provided the concept of a provenance chain which relies on hashes calculated from EO products and processors using the KSI (Keyless Signature Infrastructure) - a blockchain technology designed for automated verification of digital signatures based on record of data integrity and time. KSI only relies on cryptographic properties of hash functions and the availability of widely published verification codes which are stored in the Guardtime blockchain [70]. The results of the ESA data provenance project, when published, could inform future work by OGC testbeds.

Future works on blockchain, DLT and provenance for federated clouds should focus on the following aspects, among others:

- Experimentation of using DIDs, DADs and Verifiable Credentials with Federated Cloud and DLTs using open source software such as, for example, Hyperledger Indy, Aries, Sovrin blockchains.
- Design and development of ontologies for describing claims about assets produced by federated analytics cloud. The ontology will be defined a profile of the Verifiable Claims.
- Interoperability for different provenance systems and tools to aid in the integration of provenance information

Appendix A: Revision History

Table 3. Revision History

Date	Editor	Release	Primary clauses modified	Descriptions
May 29, 2019	S. Fella	.1	all	Initial Outline
June 20, 2019	S. Fella	.2	all	IPFS, BTFS
July 31, 2019	S. Fella	.3	all	Provenance section +PKI
August 28, 2019	S. Fella	.4	all	blockchain section
Oct 23, 2019	S. Fella	.5	all	DPKI
Nov 20, 2019	S. Fella	.6	all	Challenges
Dec 8, 2019	S. Fella	.7	all	Identity Management and finalization
Feb 5, 2020	G. Hobona	.8	10.9 and 11.1	Edits from BDLT DWG and ESA added

Appendix B: Bibliography

1. Zhao, J., Hartig, O.: Towards Interoperable Provenance Publication on the Linked Data Web. In: LDOW (2012).
2. Lavoie, B.F., Gartner, R.: Preservation metadata. OCLC (2005).
3. Simmhan, Y.L., Plale, B., Gannon, D.: A survey of data provenance in e-science. *ACM Sigmod Record*. 34, 31–36 (2005).
4. Gil, Y., Deelman, E., Ellisman, M., Fahringer, T., Fox, G., Gannon, D., Goble, C., Livny, M., Moreau, L., Myers, J.: Examining the challenges of scientific workflows. *Computer*. 40, 24–32 (2007).
5. Biltgen, P., Ryan, S.: *Activity-based Intelligence: Principles and Applications*. Artech House (2016).
6. Groth, P., Moreau, L.: PROV-overview. An overview of the PROV family of documents. (2013).
7. Moreau, L., Clifford, B., Freire, J., Futrelle, J., Gil, Y., Groth, P., Kwasnikowska, N., Miles, S., Missier, P., Myers, J., others: The open provenance model core specification (v1. 1). *Future generation computer systems*. 27, 743–756 (2011).
8. Davidson, S.B., Freire, J.: Provenance and scientific workflows: challenges and opportunities. In: *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. pp. 1345–1350. ACM (2008).
9. Thuraisingham, B., Cadenhead, T., Kantarcioglu, M., Khadilkar, V.: *Secure data provenance and inference control with semantic web*. Auerbach Publications (2014).
10. Moreau, L., Ludäscher, B., Altintas, I., Barga, R.S., Bowers, S., Callahan, S., Chin Jr, G., Clifford, B., Cohen, S., Cohen-Boulakia, S., others: The first provenance challenge. *Concurrency and computation: practice and experience*. 20, 409–418 (2008).
11. Golbeck, J., Hendler, J.: A Semantic Web approach to the provenance challenge. *Concurrency and Computation: Practice and Experience*. 20, 431–439 (2008).
12. Gil, Y., Cheney, J., Groth, P.T., Hartig, O., Miles, S., Moreau, L., Silva, P. Pinheiro da: Provenance xg final report. *W3C Incubator Group Reports*. (2010).
13. Lebo, T., Sahoo, S., McGuinness, D., Belhajjame, K., Cheney, J., Corsar, D., Garijo, D., Soiland-Reyes, S., Zednik, S., Zhao, J.: Prov-o: The prov ontology. *W3C recommendation*. 30, (2013).
14. Khan, F.Z., Soiland-Reyes, S., Sinnott, R.O., Lonie, A., Goble, C., Crusoe, M.R.: Sharing interoperable workflow provenance: A review of best practices and their practical application in CWLProv. *GigaScience*. 8, giz095 (2019).
15. Sandve, G.K., Nekrutenko, A., Taylor, J., Hovig, E.: Ten simple rules for reproducible computational research, (2013).
16. Belhajjame, K., Zhao, J., Garijo, D., Gamble, M., Hettne, K., Palma, R., Mina, E., Corcho, O., Gómez-Pérez, J.M., Bechhofer, S., others: Using a suite of ontologies for preserving workflow-centric research objects. *Journal of Web Semantics*. 32, 16–42 (2015).
17. Corcho, O., Garijo Verdejo, D., Belhajjame, K., Zhao, J., Missier, P., Newman, D., Palma, R., Bechhofer, S., García Cuesta Esteban, Gomez-Perez, J.M., others: Workflow-centric research objects: First class citizens in scholarly discourse. (2012).
18. Garijo, D., Kinnings, S., Xie, L., Xie, L., Zhang, Y., Bourne, P.E., Gil, Y.: Quantifying reproducibility in computational biology: the case of the tuberculosis drugome. *PloS one*. 8, e80278 (2013).

19. Garijo, D., Gil, Y., Corcho, O.: Abstract, link, publish, exploit: An end to end framework for workflow sharing. *Future Generation Computer Systems*. 75, 271–283 (2017).
20. Gymrek, M., Farjoun, Y.: Recommendations for open data science. *GigaScience*. 5, 22 (2016).
21. Kanwal, S., Khan, F.Z., Lonie, A., Sinnott, R.O.: Investigating reproducibility and tracking provenance—A genomic workflow case study. *BMC bioinformatics*. 18, 337 (2017).
22. Littauer, R., Ram, K., Ludäscher, B., Michener, W., Koskela, R.: Trends in use of scientific workflows: Insights from a public repository and recommendations for best practice. *International Journal of Digital Curation*. 7, 92–100 (2012).
23. Mohan, A., Lu, S., Kotov, A.: Addressing the shimming problem in big data scientific workflows. In: 2014 IEEE International Conference on Services Computing. pp. 347–354. IEEE (2014).
24. Nekrutenko, A., Taylor, J.: Next-generation sequencing data interpretation: enhancing reproducibility and accessibility. *Nature Reviews Genetics*. 13, 667 (2012).
25. Spjuth, O., Bongcam-Rudloff, E., Hernández, G.C., Forer, L., Giovacchini, M., Guimera, R.V., Kallio, A., Korpelainen, E., Kańduła, M.M., Krachunov, M., others: Experiences with workflows for automating data-intensive bioinformatics. *Biology direct*. 10, 43 (2015).
26. Stodden, V., Miguez, S.: Best practices for computational science: Software infrastructure and environments for reproducible and extensible research. Available at SSRN 2322276. (2013).
27. Stodden, V., McNutt, M., Bailey, D.H., Deelman, E., Gil, Y., Hanson, B., Heroux, M.A., Ioannidis, J.P.A., Tauber, M.: Enhancing reproducibility for computational methods. *Science*. 354, 1240–1241 (2016).
28. Zhao, J., Gomez-Perez, J.M., Belhajjame, K., Klyne, G., Garcia-Cuesta, E., Garrido, A., Hettne, K., Roos, M., De Roure, D., Goble, C.: Why workflows break—Understanding and combating decay in Taverna workflows. In: 2012 IEEE 8th International Conference on E-Science. pp. 1–9. IEEE (2012).
29. Hasan, R., Sion, R., Winslett, M.: Introducing secure provenance: problems and challenges. In: Proceedings of the 2007 ACM workshop on Storage security and survivability. pp. 13–18. ACM (2007).
30. Lu, R., Lin, X., Liang, X., Shen, X.S.: Secure provenance: the essential of bread and butter of data forensics in cloud computing. In: Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security. pp. 282–292. ACM (2010).
31. Lee, B., Awad, A., Awad, M.: Towards secure provenance in the cloud: a survey. In: Proceedings of the 8th International Conference on Utility and Cloud Computing. pp. 577–582. IEEE Press (2015).
32. Asghar, M.R., Ion, M., Russello, G., Crispo, B.: Securing data provenance in the cloud. In: Open problems in network security. pp. 145–160. Springer (2012).
33. Muniswamy-Reddy, K.-K., Holland, D.A., Braun, U., Seltzer, M.I.: Provenance-aware storage systems. In: USENIX Annual Technical Conference, General Track. pp. 43–56 (2006).
34. Sultana, S., Bertino, E.: A file provenance system. In: Proceedings of the third ACM conference on Data and application security and privacy. pp. 153–156. ACM (2013).
35. Suen, C.H., Ko, R.K.L., Tan, Y.S., Jagadpramana, P., Lee, B.S.: S2logger: End-to-end data tracking mechanism for cloud data provenance. In: 2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications. pp. 594–602. IEEE (2013).
36. Hasan, R., Sion, R., Winslett, M.: Sprov 2.0: A highly-configurable platform-independent library for secure provenance. In: ACM Conference on Computer and Communications Security (CCS). p. 122

- (2009).
37. Ko, R.K.L., Will, M.A.: Progger: An efficient, tamper-evident kernel-space logger for cloud data provenance tracking. In: 2014 IEEE 7th International Conference on Cloud Computing. pp. 881–889. IEEE (2014).
 38. Imran, M., Hlavacs, H.: Applications of provenance data for cloud infrastructure. In: 2012 Eighth International Conference on Semantics, Knowledge and Grids. pp. 16–23. IEEE (2012).
 39. Cuevas-Vicenti, Víctor, Dey, S., Köhler, S., Riddle, S., Ludäscher, B.: Scientific workflows and provenance: Introduction and research opportunities. *Datenbank-Spektrum*. 12, 193–203 (2012).
 40. Cohen-Boulakia, S., Belhajjame, K., Collin, O., Chopard, J., Froidevaux, C., Gaignard, A., Hinsén, K., Larmande, P., Le Bras, Y., Lemoine, F., others: Scientific workflows for computational reproducibility in the life sciences: Status, challenges and opportunities. *Future Generation Computer Systems*. 75, 284–298 (2017).
 41. Merkel, D.: Docker: lightweight linux containers for consistent development and deployment. *Linux Journal*. 2014, 2 (2014).
 42. Kurtzer, G.M., Sochat, V., Bauer, M.W.: Singularity: Scientific containers for mobility of compute. *PloS one*. 12, e0177459 (2017).
 43. Digital Ocean - Cloud Computing, Simplicity at scale, <https://www.digitalocean.com>, (2019).
 44. Services, A.W.: Amazon EC2, <https://aws.amazon.com/ec2/>, (2019).
 45. Google Cloud including GCP & G Suite., <https://cloud.google.com/>, (2019).
 46. Microsoft Azure Cloud Computing & Services, <https://azure.microsoft.com/en-us/>, (2019).
 47. Amstutz, P., Crusoe, M.R., Tijanić, N., Chapman, B., Chilton, J., Heuer, M., Kartashov, A., Leehr, D., Ménager, H., Nedeljkovich, M., others: Common workflow language, v1. 0. (2016).
 48. www.commonwl.org: Common Workflow Language, <https://www.commonwl.org/>, (2019).
 49. Preukschat, A.: Decentralized Identifiers (DIDs) v1.0, <https://medium.com/@AlexPreukschat/a-simple-guide-to-understanding-the-difference-between-blockchain-and-distributed-ledger-believers-79cddccc8708>, (2018).
 50. Benet, J.: IpfS-content addressed, versioned, p2p file system. arXiv preprint arXiv:1407.3561. (2014).
 51. BTFS BitTorrent File System, <https://www.bittorrent.com/btfs/>, (2019).
 52. Hughes, A.: A Blockchain Platform for the Enterprise: Hyperledger Fabric, <https://hyperledger-fabric.readthedocs.io/en/release-1.4/whatis.html>, (2019).
 53. Szabo, N.: Smart contracts: building blocks for digital markets. *EXTROPY: The Journal of Transhumanist Thought*, 18, 2 (1996).
 54. Allen, C.: The Path to Self-Sovereign Identity, <http://www.lifewithalacrity.com/2016/04/the-path-to-self-sovereign-identity.html>, (2016).
 55. Tobin, A., Reed, D.: The inevitable rise of self-sovereign identity. The Sovrin Foundation. 29, (2016).
 56. Hasine, W.B., Galperin, E.: Changes to Facebook’s “real Names” Policy Still Don’t Fix the Problem. *Electronic Frontier Foundation*. 18, (2015).
 57. Windley, P.: An Internet for Identity, https://www.windley.com/archives/2016/08/an_internet_for_identity.shtml, (2016).

58. Housley, R.: Problems with the Public Key Infrastructure (PKI) for the World Wide Web, <https://tools.ietf.org/html/draft-iab-web-pki-problems-01>, (2016).
59. Allen, C.: Decentralized Public Key Infrastructure - A White Paper from Rebooting the Web of Trust, <https://danubetech.com/download/dpki.pdf>, (2015).
60. Zimmermann, P.: PGP—Pretty Good Privacy. Public Key Encryption for the Masses, User’s Guide. 1, (1997).
61. Decentralized Identifiers (DIDs) v1.0, <https://www.w3.org/TR/did-core/>, (2019).
62. Sporny, M., Longley, D., Chadwick, D.: Verifiable Credentials Data Model 1.0. W3C, W3C Candidate Recommendation, March. (2019).
63. Hughes, A.: A Primer for Decentralized Identifiers, <https://w3c-ccg.github.io/did-primer/>, (2019).
64. Sporny, M., Reed, D.: DID Method Registry. Draft Community Group Report 05 December 2019. (2019).
65. statista.com: Internet of Things (IoT) connected devices installed base worldwide from 2015 to 2025, <https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/>, (2016).
66. Vigna, P., Casey, M.J.: The Truth Machine: The Blockchain and the Future of Everything. Picador (2019).
67. Conway, S., Hugues, A., Ma, M., Jack, P.: A DID for Everything. Rebooting the Web of Trust Technical Workshop. (2018).
68. Smith, V., Samuel with Gupta: Decentralized Autonomic Data (DAD) and the three R’s of Key Management. Rebooting the Web of Trust VI: (2018).
69. McCarron, S., Andrieu, H., Stone, M., Siegman, T., Kellog, G., Thidodeau, T.: Verifiable Credentials Use Cases. W3C Working Group Note. (2019).
70. Buldas, A., Laanoja, R., Truu, A.: Keyless signature infrastructure and PKI: hash-tree signatures in pre- and post-quantum world. International Journal of Services Technology and Management. 23, 117 (2017).