AUTHENTIC CHAINED DATA CONTAINERS (ACDC)

Samuel M. Smith Ph.D.

v1.03 2021/01/14

Abstract—The properties of Authentic Chained Data Containers (ACDCs) which may be composed from Authentic Data Containers (APCs) with Authentic Provenance Chains (APCs) are defined and explored. These are further developed from the standpoint of rich authorizations. A preliminary evaluation of VCs (Verifiable Containers) as a proposed concrete implementation of ADCs that may be expanded to include the semantics of APCs and rich authorizations is provided. An evaluation of the relative suitability of ZCAP-LD is also provided.

Index Terms— Verifiable Credentials, Authentic Data Containers, Authentic Provenance Chains, Delegation, Authorization.

1 INTRODUCTION

This paper provides a rationale for standardized semantics for what we call an *authentic chained data container (ACDC)*. An ACDC may be built up from an authentic data container (ADC) that contains an *authentic provenance chain* (APC). The term *authentic data container* (ADC) is an important abstract concept that hopefully may leverage a pre-existing open concrete implementation standard specification. The current targeted concrete implementation specification is the W3C Verifiable Credential (VC) standard [39]. We believe that *authentic data containers* (ADCs) with *authentic provenance chains* (APCs) that together compose an *authentic chained data container* (ACDC) are essential components of verifiable or authentic data supply chains which in turn are essential to what may be called the authentic data economy [11].

The dictionary definition of *authentic* is *having an origin supported by unquestionable evidence; authenticated; verified*. Likewise the definition of *authenticity* is *the quality of being authentic; genuineness*. For digital data which is easily duplicated and transmitted as bits of information we need a secure way of establishing or proving authenticity of data transmitted over the internet. The only practical mechanisms we have for securely proving authenticity of digital data are cryptographic proofs. The simplest most widely accepted type of cryptographic proof is a digital signature using asymmetric or (public, private) cryptographic key pairs. A digital signature on some data item (digital is assumed) provides a verifiable non-repudiable proof of integrity of that data item. Proof of integrity means proof that the data has not been tampered with (i.e. has not changed since it was signed). Proof of non-repudiation means proof that the signer of the data must have signed the data with the private key. Anyone in possession of both the data item and the public key may verify this proof.

Typically as a result, when applied to digital data, the term *verifiable* implies that some form of cryptographic proof, such as a digital signature, is attached to the data. Because a signature makes a non-repudiable attribution to the holder of the private key, we may say that the data is authentic in the strong sense that the signature is unquestionable evidence that the origin of the data item is the signer who holds the private key. By the act of creating the digital signature the signer is making a non-repudiable cryptographic commitment to the contents of the data item so signed. Recall that the definition of *repudiate* is *to reject as having no authority or binding force*. Therefore non-repudiable means that the signer may not reject or deny its commitment to the data item.

It is noteworthy that verifiable in this usage does not imply anything about the accuracy, credence, or truthfulness (veracity) of the data item being signed, but merely that the signer committed to the contents of that data item. Verifiable in the cryptographic sense means *verifiable attribution* (authenticity) not verifiable accuracy (veracity). Some other process is needed to provide verifiable veracity or credence. One reason to use the adjective authentic and not verifiable in the term authentic data container is to unambiguously describe what is verifiable. In this case it is verifiable authenticity. Verifiable authenticity is a different use of the term verifiable than the common one, and as a result the source of much confusion when using it with a non-security audience to modify other terms like verifiable credential . The dictionary definition of *verify* is *to prove the truth of, as by evidence or testimony; confirm; substantiate*. Because verifiable is an adjective, use of the term with respect to data, fosters ambiguity as to what is verifiable, the data itself or the origin of the data. To restate a digital signature on some data merely proves the origin or authenticity of the data and not its truth or veracity and using verifiable in this former sense is the wrong sense for all but security experts. Verified is more commonly used to describe a data item whose veracity has been established.

Thus the term authentic data may unambiguously be used to describe data that has the property of cryptographically verifiable attribution to the source of that data. In this case the source is the holder of the private key or keys used to create the signature or signatures attached to the data item. Because the term holder has another important meaning in this realm, a more precise term for holder of the private key is *controller*. The controller is the entity that may prove control over a public private key pair by signing some challenge with the private key.

There are other more esoteric forms of cryptographic proofs that have properties besides integral non-repudiable attribution to the source. These include zero knowledge proofs (ZKPs) [44]. The important property of a ZKP is that the value of attributes in the data item may be provably disclosed to the verifier without enabling the verifier to prove those attribute values to a third party. This provides enhanced data privacy protection. But for the purposes of this paper digital signatures will be used as the baseline of comparison. It is anticipated that generally speaking whereever a digital siganture is used some other cryptographic proof such as a ZKP may be used when desirable as long as it provides proof of integrity and non-repudiable proof of attribution to the source. A ZKP is a valuable extension of the essential proof of authenticity provided by a digital signature that may better limit the disclosure of information for enhanced privacy protection.

1.1 Authentic Data Containers

A data container is a syntactical way of bundling together data items. An authentic data container therefore is a data container that provides cryptographic verifiability of the integrity and source of its bundled data. In other words that data is both tamper-proof and is non-repudiably attributable to its source. Which also means it was not forged by a malicious party. If we trust the source of the data then we may have reason to trust the veracity of the data. Either way we can hold the source of the data container accountable for the contents of that container. If the data is to have any value, its authenticity must first be established and then one may benefit from establishing its veracity. But if the data is not authentic then it may be very difficult indeed to establish its veracity or otherwise convey any value.

The concept of a standardized authentic data container (ADC) makes an intentional analogy to standardized shipping containers [27–29]. Standardized shipping containers revolutionized supply chain logistics for physical goods. They provided secure traceable tamper proof transport of goods in an interoperable transportation ecosystem. Shipping containers have labeled physical seals and locks that enable attribution of the shipping container to its source and protection from tampering with its physical contents. Likewise a standardized authentic data container provides similar protections for its digital contents. A highly interoperable secure digital data supply chain is thereby facilitated by authentic data containers and the ecosystem of such supply chains forms the basis for an authentic data ecosystem or economy.

1.2 Authentic Provenance Chains

In a full supply chain, goods are not merely transported end-to-end from a "source" (producer) to a final destination (consumer) of those goods. The goods start out as raw materials that must be transformed and assembled into a finished product that may then be stored in a distribution center before final transport to the end consumer. Each of these steps may involve a change of custody with respect to the entity in control of the goods. Indeed one may generalize a supply chain as a series of transformation, aggregation, and transfer steps applied to raw materials that eventually end up as finished goods delivered to a consumer. Each of the steps along the supply chain path may involve a change of custody. These steps are perfectly analogous to those in a digital data supply chain. Moreover the management of a physical goods supply chain may benefit from being digitally mirrored or twined by a digital data supply chain. The mirroring data supply chain tracks the transformations, aggregations, transfers, and changes of custody. In either case, the security of the data supply chain is essential to its value. With respect to data authenticity, whenever there is a transfer of control or custody of the data then the source attribution also changes. This means a new proof must be provided by the controller of controller of the associated key pairs. Securing the whole data supply chain requires securing each of the steps, i.e providing authentic provenance of the data or equivalently the authentic data containers that transport that data from the custody of one entity to the next.

This brings us to the second concept, that of, authentic provenance chains (APCs). An authentic (data) provenance chain (APC) provides a proof of the changes in verifiable integral attribution (authenticity) of all the transformations, aggregations, and transfers across changes of custody of any resultant data item from each of its sources to its final destination. This may be recursively applied at each intermediary step in the chain. More discussion of the concept of data supply chain provenance may be found in earlier white papers on this subject [2; 34; 35].

In order for the provenance proof itself to be cryptographically verifiable each entity along the data supply chain needs to make a cryptographic commitment to the source(s) of data it received and the resultant data it passes along to the next entity in the chain. This binds the data at any point to its proximate origins and then by recursion to is ultimate origins. A convenient way of creating such a proof is to use a signed data structure that contains signed references to data sources as well as the data itself. By recursively walking back the signed references to data sources one may trace and then prove the provenance of the resultant data from sources to sink. A standard syntax and semantic for authentic (data) provenance chains conveyed by authentic data containers would foster an ecosystem of highly interoperable tooling for data supply chains. This is the primary objective of this work.

1.3 Secure Attribution Tree or Graph

All chaining semantics are valueless without secure attribution to each contributing source of information in the chain. An aggregation of more than one chain forms a graph. Special cases of graphs are trees and directed acyclic graphs (DAGs). A generalized chaining semantic allows graphs especially the special cases of trees and DAGs.

1.4 Automated Decision Making

Data in a data supply chain may be used for many different purposes. The abstract concepts of an authentic data container and an authentic provenance chain of data inside a container are completely independent of the function or purpose of that data. In general however, data is used to make decisions. Indeed the primary value of data products are to enable better decisions. Digital data enables automation of the decision making process. And trustworthy decisions are best fostered by authentically provenanced data. Once the authenticity of data is established one may then layer on a process to establish the veracity of that data, often as a function of the reputation or credibility of the source of that data. The following diagram shows the data processing and decision making activity value pyramid.

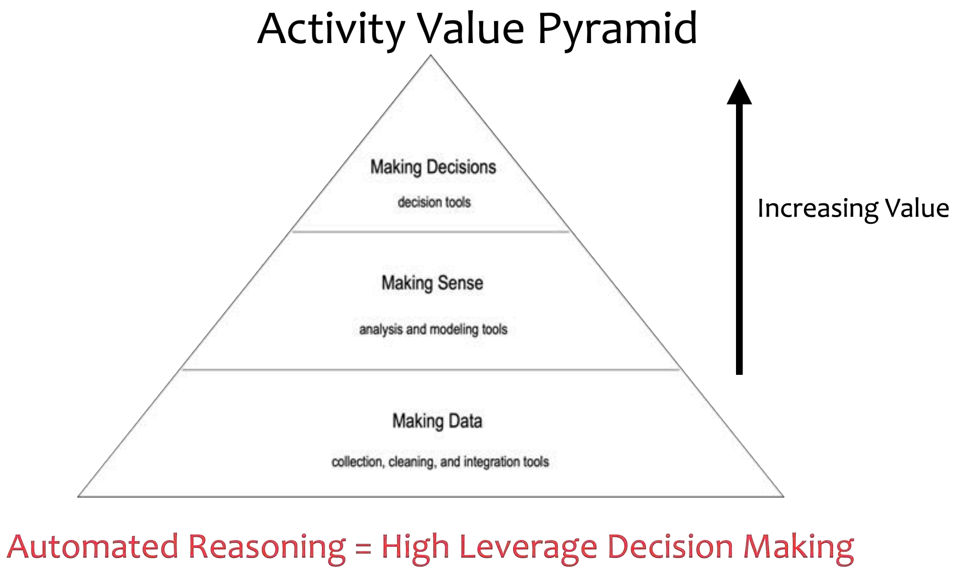
Activity Value Pyramid
Making Decisions
decision tools
Making Sense
analysis and modeling tools
Making Data
collection, cleaning, and integration tools
Increasing Value
Automated Reasoning = High Leverage Decision Making

Figure 1.1. Activity Value Pyramid

A very important type of decision-making may be generalized as rich *authorization*. Rich *authorization* benefits from an authentic provenance chain. The dictionary definition of the word *authorization* is *permission or power granted by an authority; sanction*. And *sanction* means *authoritative permission or approval, as for an action*. A decision making task may be succinctly described as finding the action or actions that best satisfy a set of objectives and constraints. In that light authorizations satisfy a class of constraints on actions. They are fundamental to decision making in any environment with legal, contractual, regulatory or other governance constraints, i.e. constraints imposed by an authority or entity with rights or power. A license is a common type of authorization. A delegated authorization is a type of chained authorization. But authorization is tightly bound to authenticity. The authority under which an authorization is issued must be securely determined before the authorization has any functional value. When the operative semantics of a data item are under the control of the signer, then a signature conveys a verifiable non-repudiable commitment to those operative semantics. Those operative semantics could be an authorization.

To generalize further, we live in the information age where the organizations with the largest market value are information centric. Indeed every organization now derives value from data processing and automating those data processing workflows. All decision-making workflows within organizations may realize tremendous gains in productivity when securely automated. When the data crosses entity boundaries, secure attribution becomes vital to the value of the resultant data products. The goal thereby is to enable secure automated decision making workflows especially when the data crosses entity boundaries. Secure attribution is provide by authentic data containers and authentic provenance chains. Moreover using decentralized mechanisms to prove authenticity in ADCs and APCs enables automated decentralized decision making workflows. Currently the realization of market value for information is concentrated in large multi-national corporations. Broadening that base through decentralized automated authentic data supply chains will unleash new innovation and sources of value.

Given a standard specification for authentic data containers (ADCs) with authentic provenance chains (APCs) we may be able to support in general authentic data supply chains in an authentic data economy. The remaining question therefore is how to implement a standard specification for ADCs with APCs.

2 Verifiable Credential Hypothesis

Our hypothesis is that leveraging an existing standard may be more productive than creating a new one. The standard under consideration as the basis for ADCs with APCs is the W3C verifiable credential (VC) standard [39]. The verifiable in verifiable credential refers to the verifiability of a cryptographic proof of the origin of the credential. Usually this proof is provided by a digital signature from an asymmetric key pair. In other words its the same meaning of verifiable as defined above for authentic data container. The VC specification provides for verifiable authenticity not verifiable veracity. Thus it would be more precise to call them authentic or authenticatible credence than verifiable credentials. Originally the standard was called the verifiable claims standard. The dictionary definition of *claim* is *an assertion of something as a fact, an assertion of a right or an alleged right*. In the most common usage claim is a more precise term for the data conveyed by a VC, but the term was changed from claims to credential because in common usage verifiable claims implies that the truth of the claims is verifiable not the origin of the claims [25]. But the root problems is using the term verifiable at all, because as discussed above, attaching verifiable to a data item does not specify what is being verified; the origin or the truth of the data item. Unfortunately the term credential is also problematic. The dictionary definition of *credential* is *evidence of authority, status, rights, entitlement to privileges, or the like*. A credential is therefore a type of authorization. This is a narrower definition than claim and much less precise to the normal use case. Indeed, there is a competing standard for verifiable authorizations for access control based on object capabilities called Authorization Capabilities for Linked Data (ZCAP-LD) [42]. Indeed an entirely ironic point of disagreement in the community is that verifiable credentials (synonymously verifiable authorizations) must not be used for authorizations. Instead the ZCAP-LD standard should be used. Object capabilities and ZCAP-LD are discussed in more detail later in this document.

The hypothesis is that the existing VC standard (despite its unfortunate use of terminology) may serve as a nascent ADC standard. The most glaring limitation of the current VC standard is that it does not provide standard syntax or semantics for chaining together credentials i.e. providing support for APCs in a standard way. The VC standard would have to be augmented with VPC semantics. There are use cases in the wild where VCs are being used for APCs but they are bespoke and hence not interoperable. This is analogous to the way shipping worked before standardized shipping containers. The transaction costs of using non-standard shipping containers were orders of magnitude higher than using standardized containers. We believe the same economics apply to data supply chains. Standardized ADCs and APCs will reduce data supply chain transactions costs by orders of magnitude and unleash previously unobtainable opportunities for value creation and capture. This is a cooperative network effect that reduces trust transaction costs (see [30; 33]).

On an historical note there is already an IETF standard for certificates as authorizations. It is IETF RFC-2693 SPKI Certificate Theory for Authorizations [4]. It includes many similar features at least in purpose if not implementation to the W3C VC specification including certificate revocations lists.

2.1 Chained Credentials

One effort to establish standard semantics for chaining together VCs is the Hyperledger Aries RFC-0104 Chained Credentials [8]. This RFC addresses both provenancing issuance of credentials as well as delegating authorizations via credentials. The RFC is detailed and is not reproduced here. A related effort is the Simple Grant Language (SGL) [5] specification and implementation. Simple grant language allows the expression or role based authorization rules. Aries RFC-0104 is a good starting point for developing a chaining semantic and provides some seminal use cases. One of those is to use chaining (delegation) for guardianship applications. This is related to Hyperledger Aries RFC 0103: Indirect Identity Control [6]. A discussion of why using a chained credential specification is preferable to using ZCAP-LD is provided here [7].

The following data structure provides a simplified notional illustration of a chaining semantic. It is not as detailed or complete as that proposed above but does illustrate the essential features:

```
Provenance Chains (Credence or Authorization)
{
     sources:
     [
       {id: val, rules: { …}, weight: value, …},
       {id: val, rules: { …}, weight: value, …},
        …
     ],
     destination: id,
     rules: { … },
     weight: value,
     sink: id,
     …
}
```

What this example does provide is support for weighted aggregation. The rules in the example are meant to express how the chaining works relative to expressed objectives and constraints. This is explained in more details below.

2.2 Entrained Authorizations

We believe the most compelling reason for a nativeAPC (authentic provenance chain) specification overlay to the VC specification is that the most important use cases benefit authorization s that are entrained with the associated data supply train. Our view is the the ultimate purpose of a data supply chain is to support decision making. Decision support is a type of automated reasoning where authorizations are merely a form of constraint satisfaction and all authentic data may be viewed a contributing to either objective or constraint satisfaction in order to take action. Consequently an authentication is merely another type of data that contributes to the decision. It is most conveniently and security provided using the same authenticity mechanism as any other data from the same source. When sharing data via VCs it makes the most sense that any associated authorizations/consents/restrictions i.e. constraints be conveyed in-band, self-contained, attached, embedded, or entrained with that data as it is shared. Not conveyed in a separate out-of-band mechanism. It is hard to find compelling reasons to do otherwise.

From an adoption on secure tooling perspective. It makes sense to have one authentic data container standard with multiple semantics on how to interpret the contents of that container. Thus to have two different authentic or verifiable container standards each with different security tooling and signature verification processes merely to convey two different attribute semantics means two security stacks which separably opens up vulnerabilities. The hard part, the part we must not get wrong is the signature verification which means establishment of control authority over the identifiers to which the signatures are ascertained. We therefore want one and only one way to do that. And lacking that we want one and only one tooling stack to do that. Given how hard it is to solve the signature verification/identifier problem, wanting yet another signature verification standard merely to convey an authorization attribute semantic versus a non-authorization attribute semantic is needless complication. Thinking of VCs as Verifiable Containers not Verifiable Credentials removes semantic dissonance. Recall from above that the definition of credential is a type of authorization. The existing VC standard is merely a standard for verifying signatures on some container of data. The semantics of the data are free with respect to the standard verification process. Anyone can define their own schema for the semantics of what goes inside the container.

The preferred approach is a single stack of thin layers not multiple stacks of parallel thick layers. Only the top layer splits into multiple "semantic" applications as shown by the vertical bar below. (note first is bottom last is top):

- Protocol to establish control authority over asymmetric key-pairs and identifiers for signatures
- Authentic Data Containers with verifiable signatures
- Authentic Provenance Chains
- Attributes | Authentications (semantics conveyed inside containers)

A discussion of some of these issues may be found here [40].

In light of this layers, a provenance chaining semantic may be referred to as a meta or super semantic that connects data together. A VC chaining semantic supports more than authorization/delegation it also supports provenance of data transformations, data custody source to sink, trust provenance (reputation), audibility, fine grained issuance semantics etc. In this regard authorization may be viewed as merely one of many sub-semantics of a chaining super semantic.

2.3 To ZCAP or Not

There is some disagreement within the community about adding authorization semantics to VCs instead of using the competing ZCAP-LD specification [42]. This disagreement may be misplaced. As described above we view authorization in its most general sense, as part and parcel of conveying information for decision making. Whereas ZCAP-LD is a narrowly defined specification that employs something called object capabilities. Object capabilities were designed as an access control mechanism for computer operating system and software services. Although object capabilities has applications beyond that narrow use case, its view of an authorization is very narrowly defined. Some background of object capabilities may be found here [22][14][21][17][15][7][18][16]. Therefore any type of authorization that is not well suited to object capabilities does not have a standardized semantic and the ZCAP-LD specification by definition is ill equipped to provide. Thus very simply it makes sense to add generalized authorization semantics to the VC specification. But given that, it poses the question then, could not object capability semantics also be added to the VC spec thus obviating the need to use anything but the VC spec. That may be the underlying concern but if there is a strong reason to use a non-VC specification that reason may continue in spite of a VC native object capability specification. But clearly any non-access control use case of authorization (such as licensing, consent, custodianship, guardianship, business process management ,supply chain etc.) is best provided by a VC native spec that some other yet to be defined non-access control based authorization specification. Object capabilities are a very useful model for access management and this work is not an anti-ZCAP-LD effort but an attempt at a general approach where an object capability semantic may be one of many. The important feature for these generalized authorization semantics is that the semantics are embedded into existing VCs, i.e. VC Native, and not part of a separate conveyance as is the stated objective of W3C ZCAP-LD [42].

3 Property Comparison

The literature around object capabilities for access control is extensive and it may be easy to discount other forms of authorization as a result. One way to understand the need for other forms of authorization is to do a property comparison. An authorization type can have several different properties or features that differentiate it from other authorization types. This section with briefly describe some of these distinguishing features or properties of different types of authorizations.

3.1 Verifiable Credential Models

Because the W3C VC specification is the standard under investigation for adoption as the basis for ADCs and PDCs we will use the associated terminology notwithstanding some of its definitional shortcomings. But for the sake of clarity, in the following diagrams everywhere one sees the work verifiable it means verifiable authenticity and could more precisely replaced with the term authentic or authenticatible.

The primary use cases for VCs employ a tripartite model of *issuer-holder-verifie*r [1; 23; 24; 40]. This is shown in the following diagram.

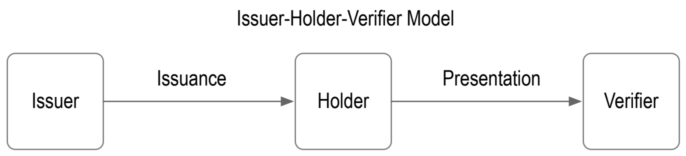

Figure 3.1. Basic Tripartite Model for VCs

This is a tripartite model. The three parties are the *issuer*, the *holder*, and the *verifier*. In this model the *issuer* issues a VC (verifiable credential). The *issuer* is the source of the data. In the issuance process, abstractly the *issuer* creates an authentic data container (ADC) that in this case is concretized in the form of a VC. The issuer fills the container with data and then provides other information to enable some other party to prove the authenticity of the the contents of that container. This other information allows another party to establish what public keys are authoritative for that issuance. This may be an identifier that may be resolved to provide proof of the associated controlling key-pairs. Attached to the VC are signatures that wrap the contents of the container. In this sense the container has authenticatible contents. But authentication is limited to authenticating that the VC was indeed issued by the controller of the key-pairs associated with the attached signatures. It is only authenticating attribution to the controller of the key-pairs, i.e. the *issuer*, nothing more.

The Holder receives a copy of the VC. In this case holding refers to holding the VC not something else. Often the Holder is the subject of the data in the VC but not necessarily so. The Holder may then at some time present the container to a Verifier. The Verifier does not verify the veracity of the contents of the data but merely verifies the authenticity of the contents with respect to the authoritative public keys for issuer as the source of the data. In this light one might consider using Authenticator instead of Verifier as the label for this role. But as mentioned above this is merely authenticating to the controller of the key-pairs for the issuing signature. It is important to reaffirm that verification is merely the verification of proofs of authenticity to the controller of the authoritative key-pairs, i.e. the controlling issuer, and not the actual person or organization associated with the controlling issuer. The process of verification of authenticity of the VC (ADC) with respect to the controlling issuer (not the associated entity) may be performed in different ways. This gives rise to different models for authenticity verification of the contents of the data.

The most common model employs a verifiable data registry to support authenticity verification. This is shown in the diagram below.

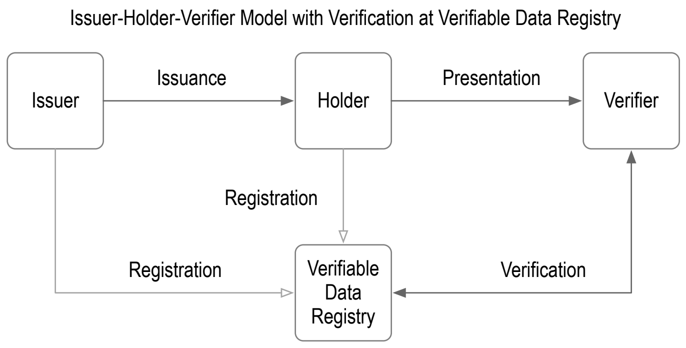

Figure 3.2. Tripartite Model with Verifiable Data Registry

The purpose of the verifiable data registry (VDR) is to enable verification of the VC by the verifier without communicating with the issuer. Once again, verifiable in verifiable data registry refers to data that is verifiably attributable to the source of that data. In this case the *issuer* and/ or *holder*. In this model, isolating the *verifier* from the *issuer*, may better protect the privacy of the *holder* with respect to its use of the VC from correlation by the *issuer*. The trade-off is that these enhanced privacy protections come at the cost of VDR infra-structure. Typically data items in the registry include identifiers, attributes, schema, and revocation information. A VDR is often provided by a public distributed consensus ledger such as Sovrin [36; 37]. However, a ledger is not required when using a public VDR based on KERI [20; 31; 32]. Because verification happens without needing communication between the verifier and issuer this may be called an open-loop verification model. Verification happens independently of the issuer. An open-loop model also enables different availability and privacy properties. For example, with a highly available VDR the issuer does not need to be highly available. The independent nature of the VDR enables persistent VCs whose validity is maintained across key rotations of the authoritative key-pairs. The makes VC validity independent of key management which in turn enables larger scale, even global scale VC infrastructure. This persistent model is explained in more depth here [34]. Finally an open loop model enables chaining in a decentralized eco-system. Either for provenance of source or for provenance of authorization, successive verifiers may become issuers in their own right of VCs derived from VCs presented to them. All the dependencies for verification flow back to the VDR not the issuers..

Alternatively the VDR infrastructure in this tripartite might be private. It might be or provided with private infrastructure amongst the parties or the VDR information might entrained with the VC itself either by reference or as an attachment. In this case the whole process may be made private to the three parties. This private mode of operation is enabled by KERI's direct mode or by Peer DID [9; 31].

A version of the tripartite model that does not employ a VDR is shown in the diagram below:

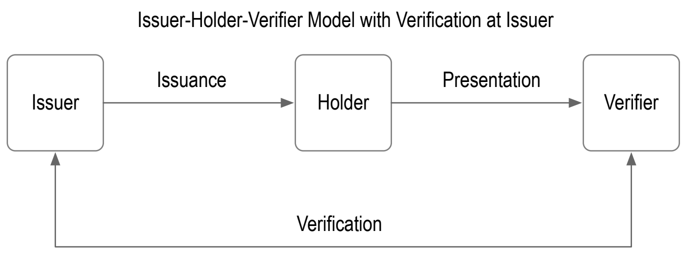

Figure 3.3. Tripartite Model without Verifiable Data Registry

In this model the *verifier* must contact the *issuer* directly to verify the VC. All the information that is normally maintained by a VDR to support verification is instead maintained by the *issuer* itself. This allows the *issuer* to track the behavior of the *holder*. Thus this model is fundamentally problematic for privacy protection with respect the *issuer* and *holder*. It also imposes an availability constraint on the *issuer*. The advantage is a simpler verification infrastructure that is closer to many more conventional models for authentication and may also allow leveraging tooling from these more conventional approaches. The disadvantage is that chaining VC where a verifier may become an issuer in the provenance chain requires maintaining availability of all the issuers in the chain independently. This may be highly problematic for a decentralized ecosystem of multiple disjoint entities.

The last model is a bipartite model. In this model the issuer also acts as the verifier. The holder presents the VC back to the issuer who then verifiers the presentation. This is shown in the figure below:

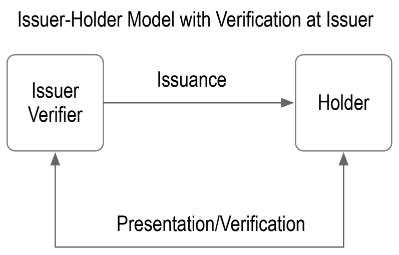
Issuer-Holder Model with Verification at Issuer
Issuer Verifier
Issuance
Holder
Presentation/Verification

Figure 3.4. Bipartite Model

This is the simplest model and also the most well known. Many access control mechanisms employ this bipartite model. The VC is analogous to a bearer token. Whoever presents the token gets access. This is also the model employed by many user managed federated identity systems. The user as holder is allowed to manage information on the issuer's infrastructure given the holder is able to present a previously issued VC for that purpose. This is also a closed loop model. This model is impractical, however, for many very important VC use cases like licensing. In addition, because the verifier role has been eliminated there is no mechanism for enabling decentralized propagation of consent constraints on verifiers as recipients of information, much less chained provenance of information.

3.2 Open Loop vs. Closed Loop Authorization

One of the primary use cases for authorization is to delegate. This requires a chaining semantic. A simple mode is to have both the service and delgator be the same entity. This we call a joint delgator-service model. In this model presentation of the authorization is back to the delegator. This is a common access control model.

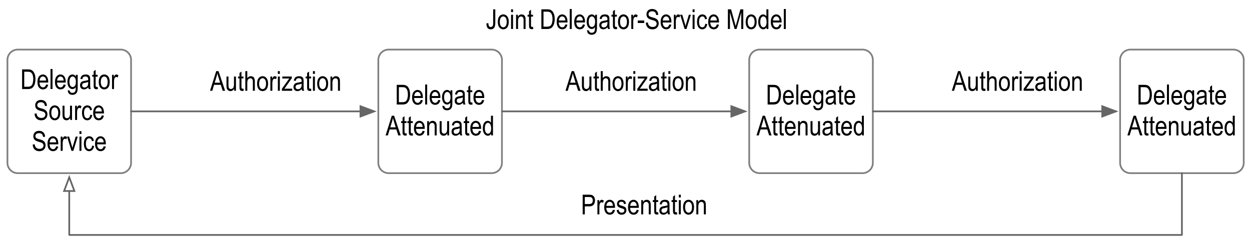

Figure 3.5. Joint Delegator-Service Model

In this joint model verification also happens back at the delegator source. This makes it closed loop as well. Joint models are closed loop by constructions. A chaining semantic of delegated authorizations overlaid on a closed loop VC verification model may be diagrammed as follows:

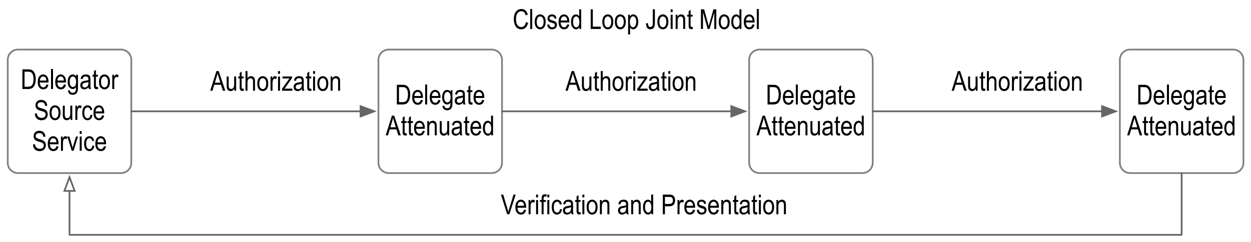

Figure 3.6. Joint Delegator-Service Model

In this joint closed loop model, the role of issuer is replaced with the role of delegator and the role of *holder* is replaced with the role of delegate. Each subsequent delegation goes to a new delegate (holder). Usually the delegated authorization may be attenuated. In other words the rights or privileges authorized thereby are reduced from the maximally permitted set in the original authorization. The final holder delegate of the authorization then presents the authorization back to the delegator who verifies the issuance and upon successful verification provides access to the service. This joint verification presentation is often assumed which makes a split model confusing. The joint model is essentially the bipartite closed loop model above but extended with multiple chained holder delegates. This is the typical model employed by object capabilities.

A variant of this model splits the service provider of the authorized service from the delegator.

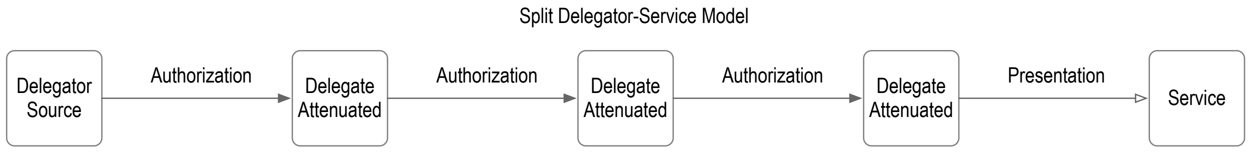

Figure 3.7. Split Delegator-Service Authorization Model

This split model may employ either closed loop or open loop verification. Closed loop verification is shown below:

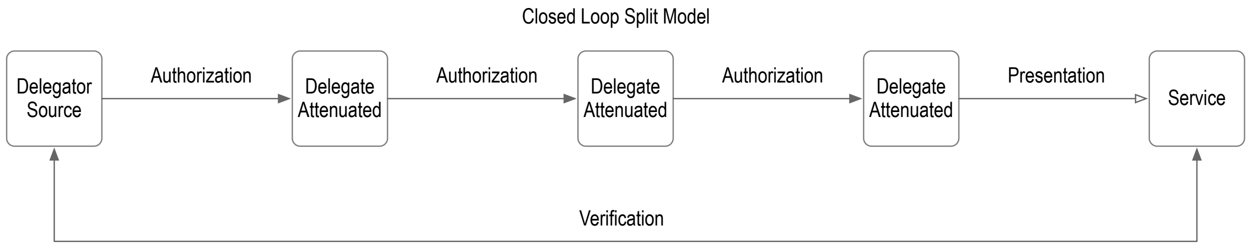

Figure 3.8. Closed Loop Split Delegator-Service Model

In this close loop split model, the service must communicate back with the source delegator to verify the authorization. This is the model typically employed by OpenIDConnect where the identity service provider is the delegator issuing tokens that the service provider must verify. Some object capability applications also employ this model. It is a common distributed access control model.

The advantage of both closed loop models is that one may employ a very simple authorization revocation rule with respect to rotations of the authoritative keys. If the keys rotate the authorizations are automatically revoked. This is enabled because the verification happens back at the source i.e. the issuer delegator of the authorization which is also the issuer controller of the keys pairs used to verify the signatures on the authorization. The issuer simply verifies against its current set of key-pairs and if they have been rotated from when the signatures were created the signatures will automatically fail. Unfortunately this model is inappropriate for many authorizations like licensing or consent where the authorization need to persist independently of key rotations. Imagine millions of business licenses all of which have to be reissued anytime the the issuer rotates the key pairs originally used to sign the licenses. The split model is also employed in forensic or compliance authorization models where the verification happens after the fact of access. A driver's license is a prime example of a split model used forensically. Access to the roadways does not require presentation of the license but if there is an incident then presentation of the license is required after the fact of the incident to enforce compliance or mete out punishment.

The open loop verification with a split model is shown below:

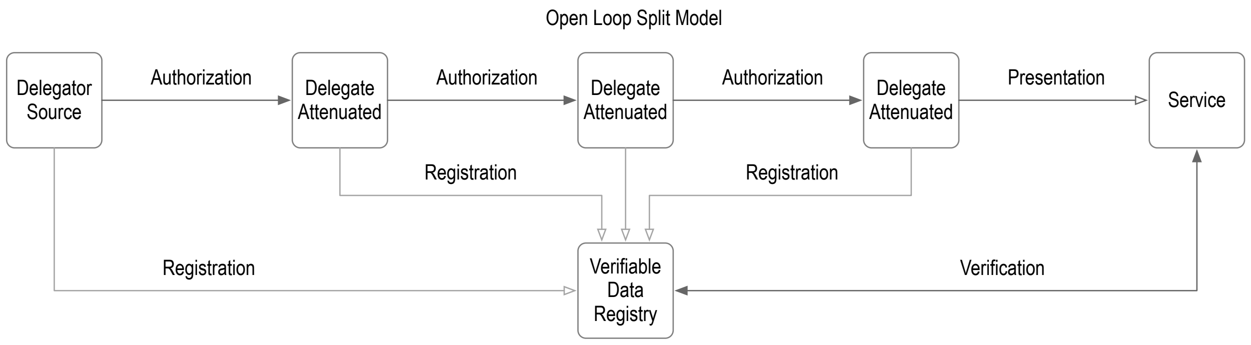

Figure 3.9. Open Loop Split Delegator-Service Model

The open loop model requires a verifiable data registry (VDR). This means that the service may verify any authorization against an independent registry. The delegator registers the original authorization in the registry. Registration by delegates with the VDR is optional. It depends on whether or not there are additional verifications of the authenticity of the delegates or if the authorization is a bearer authorization (any delegate will do). The major advantages of an open loop authorization system are as follows: Data privacy with respect to correlation by the delegator of the behavior of the delegates may be better protected. Authenticity of delegates may be independently verified. Authorization lifecycle may be independent of delegator key rotation. That is authorization may be persistent across rotations. This means that authorization do not have to be reissued in the event of a key rotation.

3.3 Persistent vs. Ephemeral

In this case persistent means that authorizations may remain valid in spite of a change or rotation of the authoritative keys used to sign the authorization. Ephemeral means that the authorizations are automatically revoked (i.e. do not persist) across a key rotation. This is an important feature in large scale operations where key management infrastructure may be multi-valent and authorizations may be hierarchical. As described in the section above one advantage of open-loop systems is that they enable the de-coupling of authorization life-cycle events from key management events such as key rotation while still preserving privacy and decentralization. While closed loop models could employ such de-coupling they still suffer from the fact that they enable correlation of the behavior of the delegates and tend to centralize key management and authorizations. Different authorization revocation rules are described in more details here [34].

3.4 Chained vs. Unchained

The diagrams above show delegation of authorization which is a type of chained authorization model. But some authorization models to not allow chaining. This is one of the limitations of the current VC specification which has no standing chaining semantic. Chaining is essential for any scalable provenancing. Besides the applications so far described in this paper for chaining semantics, an important one is for secure attestation semantics. The IETF and TCG (Trusted Computing Group) are promulgating several standards based on verifiable remote attestations of the configuration of trusted executions environments [26; 38; 43]. The semantics of these verifiable attestations could conveyed by ADCs (VCs) with appropriate chaining and provenancing semantics

3.5 Attenuable vs. Unattenuable

The diagrams above show attenuation of authorization which is dependent on chained authorization. Attenuation provides a practical trade-off between issuing many fine grained fit for purpose authorizations and a few but too permissive general authorizations. A delegate who in turn chooses to become a delegator may issue a more limited fit-for-purpose delegation that is an attenuation of a more general authorization it holds. This limits the risk to the delegator of the attenuated delegation while providing the flexibility to delegate. Most delegated authorization mechanisms benefit from at least limited attenuation capabilities.

3.6 Aggregative vs. Unaggregative

In complex or rather practical real world decision making tasks (as opposed to trivialized ones) aggregation of constraints is the usual situation rather than an exceptional case. Aggregation allows the joining of multiple authorizations for composite authorizations. When chained it allows for trees of authorizations. With a tree authorization the vast majority of hierarchical decision making structures may be implemented. It is notable that access control is not usually hierarchically aggregated this makes most access control mechanisms including many implementation of object capabilities ill suited to real work decision making under authority constrains (i.e. authorizations). The following diagram shows an aggregation tree.

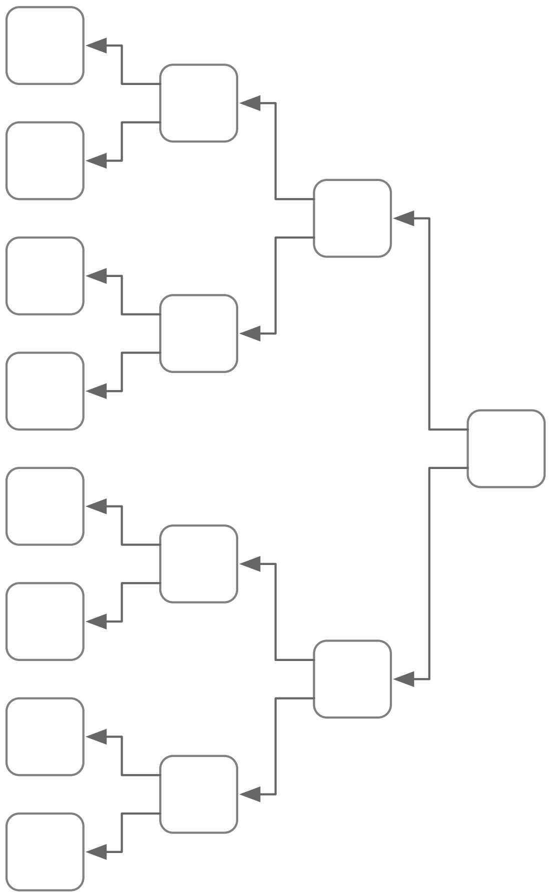

Figure 3.10. Aggregation Tree

The inverse of aggregation is expansion. In this case multiple delegates may be authorized by a single delegator. The following diagram shows an expansion tree.

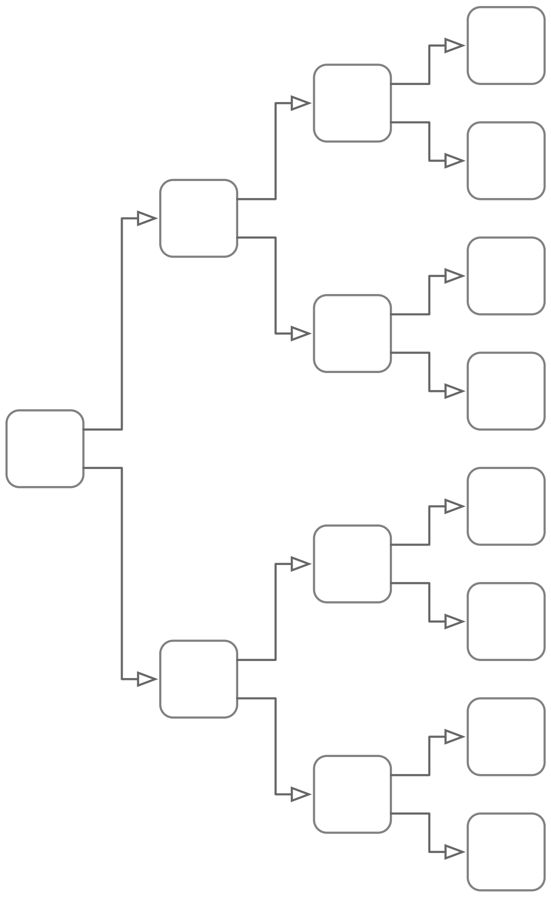

Figure 3.11. Expansion Tree

In systems where both aggregation and expansion are possible then complex relationships between delegators and delegates with respect to authorization are possible. This is shown in the following diagram.

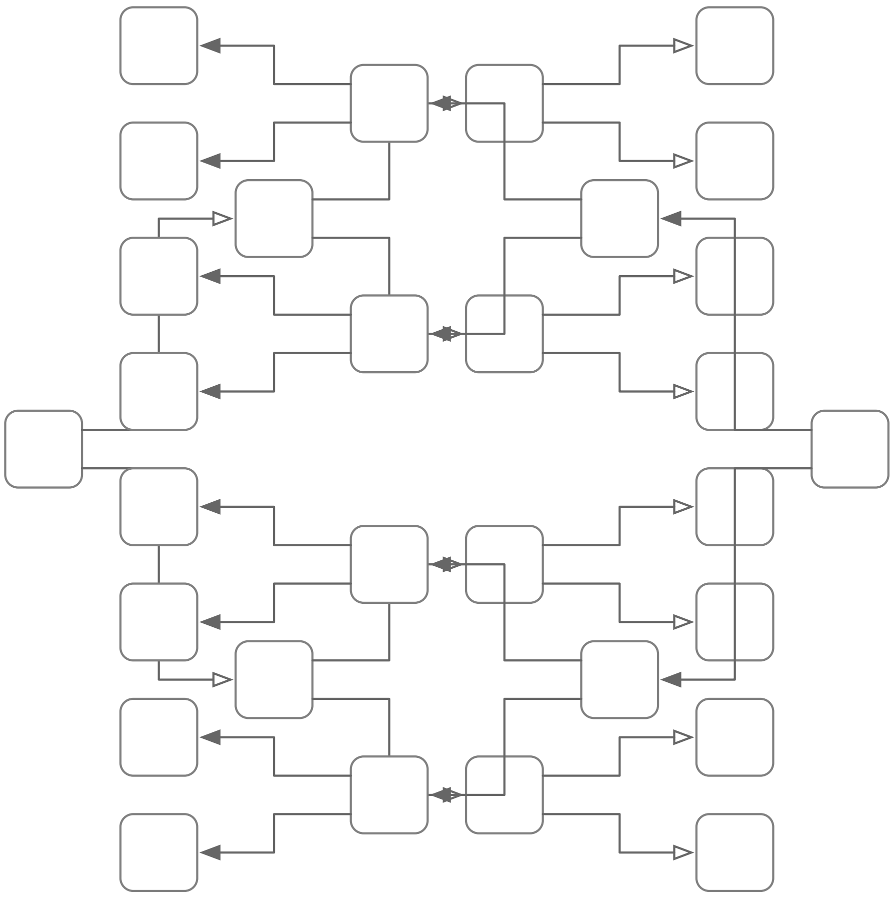

Figure 3.12. Complex Authorization Relationships

3.7 Weighted vs. Unweighted

When aggregating authorizations (constraint satisfaction) richer aggregations are provided by relative weighting of the different authorizations being aggregated. This makes little sense in an access control applications because access control control rules are Boolean. Either one gains access or one does not. But in more complex decision making tasks, especially chained hierarchical constraints the end authorization may be evaluated as satisfied when the aggregate reaches a threshold. In simple k of n threshold scheme all authorizations have the same weight. But more sophisticated schemes benefit from different weights being assigned to different aggregated authorizations. One example is the weighted signing thresholds in KERI [31].

3.8 Implicit vs. Explicit

Many credentials carry implicit authorizations. The actual actions that are authorized are not embedded or specified as rules or parameters or data embedded in the authorization itself but are implied by the type or identity of the authorization. A drivers license is a good example. A drivers license entitles the holder to driving privileges but those privileges are specified elsewhere and there is no direct way to evaluate them upon presentation of the authorization. Indeed the most common implementation of implicit authorizations are credentials and no doubt inspired the use of the term verifiable credential. In contrast, object capabilities, access tokens and the like are explicit authorizations with embedded are expressed rules for evaluating the associated privileges. Many applications besides access control such as custodianship or guardianship or consent benefit from explicit authorization semantics.

3.9 Cooperative vs. Uncooperative

A unique form of delegated authorization is provided by KERI for delegating identifiers [31; 34]. This enables delegated authorization from a more secure key management infrastructure but less performant to a less secure but more performant key management infrastructure while enabling recovery of key compromise to be protected by the more secure delegating infrastructure. This enables horizontally scalable performance without losing ultimate protection. In cooperative delegation both the delegator and delegate must participate in the creation of the authorization. It is not merely a one way authorization down from the delegator. The following diagram shows KERI's cooperative identifier delegation:

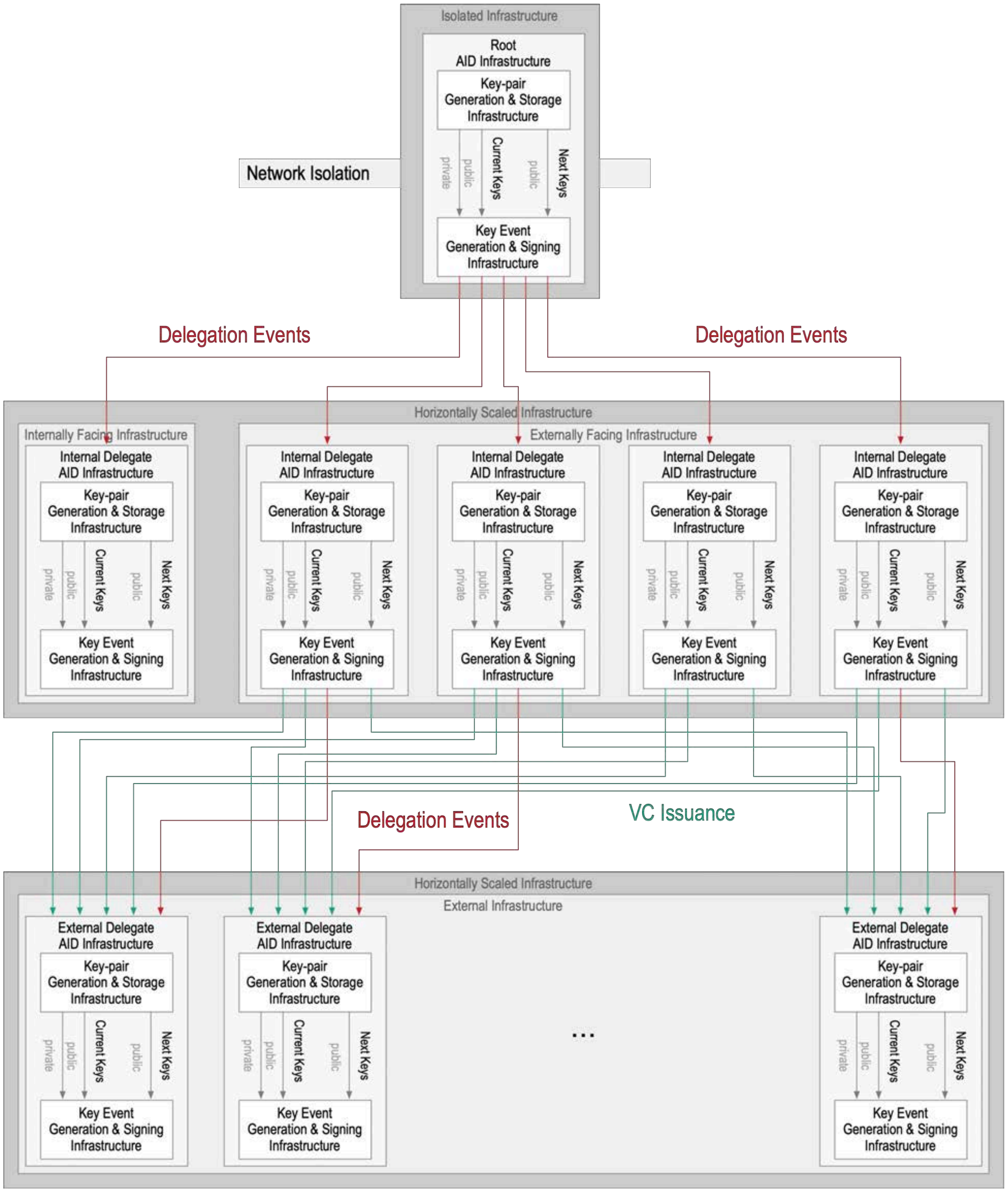
Isolated Infrastructure
Root AID Infrastructure
Key-pair Generation & Storage Infrastructure
private
public
Current Keys
public
Next Keys
Key Event Generation & Signing Infrastructure
Network Isolation
Delegation Events
Delegation Events
Horizontally Scaled Infrastructure
Internally Facing Infrastructure
Externally Facing Infrastructure
Internal Delegate AID Infrastructure
Delegation Events
VC Issuance
Horizontally Scaled Infrastructure
External Infrastructure
External Delegate AID Infrastructure
...

Figure 3.13. Cooperative Delegation in KERI

Another important use case for cooperative delegated authorization is consent [12; 12; 13; 19]. Several specifications have been developed around consentful authorization these include the Kantara Initiatives Consent Receipt specification which is a nascent ISO 27560 standard [13; 19]. Another consentful authorization is the consent to create binding specification which are included in the Kantara Distributed Assurance specifications [3; 12].

A consentful authorization imposes a liability on the delegate. The delegate must agree to the liability as a condition of creating the authorization. This makes it a cooperative authorization.

Oftent a consentful authorization may involve a disclosure of confidential information. Consent may thereby be used to protect privacy via the imposition of liability counter incentives to further disclosure. Chained consent enables delegated liability. This has been called chain link confidentiality [10].

There are two primary approaches to privacy protection in online exchanges of information.

- Manage the degree of disclosure
- Manage the degree of exploitation given disclosure

An open loop authorization model may employ both. Verification to the VDR not the issuer mitigate upstream correlation and consentful cooperative authorization mitigate downstream correlation.

The first approach is inherently leaky over time. Any disclosure over time becomes more and more correlatable. Limited disclosure without consent liability protection provides ephemeral privacy protection.

With cooperatively chained relationships, the chaining may be viewed as flowing both ways. Authorizations or constraints on actions flows from delegator to delegate. Whereas credence or trust flows from delegate to delegator. Likewise authority flows from delegator to delegate. Provenance is about tracing these relationships in either direction depending on what is being flowed. In each cooperative relationship we can think of the delegator as sending authority to delegate and the delegate sending credence to the delegator.

From a decision making perspective, when verifying aggregative cooperative provenance relationships we can express the verification of those relationships with rules. Thus verifications also becomes cooperative. For example the delegator imposes constraint rules for authorizations in order to restrict actions by the delegate and the delegate evaluates composition rules for authorizations in order to aggregate a set of authorizations to enable desired actions for meeting its objectives. A verifier may have its own rules of both types (objectives and constraints) for deciding whether or not to accept an authorization. A valid authorization therefore only occurs upon the is the conjoint satisfaction of three sets of rules, namely: delegator, delegate and service. In general authorization semantics include semantics for the expression and satisfaction of rule chains. The DIF Presentation Exchange specification is an attempt to standardize the ways that verifiers and holder presenting VC may better express and satisfy each other's rules or requirements. It does not include a chaining semantic but could be leveraged to provide a more expanded specification for generic APCs [1]. Likewise the W3C-CCG Traceability Vocabulary Specification provide some semantics for tracing the source of products in a supply chain [41]. It also does not have an explicit chaining semantic but could be leveraged to inform a more generic APC semantic.

3.10 Object Capabilities vs. Verifiable Credentials

Given the list of authorization properties we can classify and compare typical object capabilities and typical VCs as authorizations.

Typically object capabilities implementations such as ZCAP-LD provide closed-loop, ephemeral, attenuable, chained, unaggregative, unweighted, explicit, uncooperative authorizations.

Whereas the typical VC implementations provide open-loop, persistent, unattenuable, unchained, aggregative, unweighted, implicit, uncooperative authorizations.

As the foregoing dissuasion should clearly expose, other combinations of properties have merit and therefore would benefit from support for expressing those other combinations in an interoperable standard way. This is the goal of ADCs with APCs. What is also clear by this comparison is the ZCAP-LD is much too limiting as a general authorization specification and VCs are a better candidate.

4 Example Use Case

As a departure point a seminal use case is presented.

Steve is as a seller what like to sell an item to a buyer Bob would works for Edge Co. In order to buy Bob needs authorizations from his employer. Steve as the seller needs to verify that Bob has sufficient authorizations in order to trust that Steve will indeed by paid.

Bob is an employee of Edge Co who is authorized to make purchases. Edge Co is very large and contracts with several identity proofers who issue credentials of proof is identity. Edge Co issues to each of these contracted identity proofers an authorization to be an authorized identity proofer for Edge Co. Therefore Steve must validate several credentials (authorizations). These are one from Edge Co to Bob indicating that Bob is an Employee. One from Edge Co to any of the identity proofers to authorize them as valid proofers for Edge Co employees and one from one of the Identity Proofers to Bob indicating proof of Bob's identity (name etc).

Edge Co authorizes employees to issue purchase orders with an authorizations from either the director of procurement or by any two procurement managers. A procurement manger must have a credential from the director of procurement authorizing the manager to delegate procurement authorization.

Edge Co also constrains the size of any procurement from any given source that may be authorized by a procurement employee. This authorization must stipulate the maximum amount and the source. It must be issued by a procurement manager or by the procurement director. Authorizations above the maximum but be authorized by the CFO of Edge Co.

Each employee of Edge Co at any level of responsibility must have proof of identity from one of the contracted identity proofers and an employee authorization issued by Edge Co. Anything issued by Edge Co is actually issued by at least two of the managing directors of Edge Co. Managing Directors are authorized by the controllers of a special set of governing keys for Edge Co. These keys are known through a common knowledge well know associations. These include an credential from an industry legal entity vetting organization that certifies the root identifier associated with this special set of governing keys.

In order for Steve to verify that indeed the purchase order is valid, Steve must verify a hierarchical tree of authorizations. But because all authorizations in the tree may be verified computationally, this complex authorization may happen in the fraction of second. Steve need merely have established prior to the verification the root identifiers and controlling keys for Edge Co and each of its contracted identity proofers. Bob then may present the leaf authorization in the tree and then Steve can walk the tree to verify in turn each credential.

Edge Co has published and signed its schema and rules for authorizing purchases. Steve may verify against this set of rules and schema.

It would highly impractical if not prohibitively so for Steve to support verification of purchase orders from multiple companies like Edge Co without a standardized set of semantics for the schema and the rules for verifying these complex authorizations. But this example is indeed coherent with real world constraints. This means that the current VC spec may not actually reduce transactions costs over current manual methods of verification without augmenting the standard to support chaining and richer authorization semantics.

Whereas ZCAP-LD does not provide sufficient capability to support these sort of real world examples (non-access control).

5 Conclusion

The goal of this effort is to explicate the properties of a universal ACDC (authentic chained data container) ACDC specificaion. This may be decomposed into an ADC (authentic data container) standard with an APC (authentic provenance chain) semantic standard. And to specifically evaluate whether or not the W3C VC specification may be expanded to provide a concrete implementation of ADCS with APCs. The advantage of such an approach is that only one set of tooling and one infrastructure would be needed to support the authentic data economy.

Acknowledgments

The author wishes to thank all those that provided helpful feedback.

Author

Samuel M. Smith Ph.D. has a deep interest in decentralized identity and reputation systems. Samuel received a Ph.D. in Electrical and Computer Engineering from Brigham Young University in 1991. He then spent 10 years at Florida Atlantic University, eventually reaching full professor status. In addition to decentralized identity and reputation, he has performed pioneering research in automated reasoning, machine learning, and autonomous vehicle systems. He has over 100 refereed publications in these areas and was principal investigator on numerous federally funded research projects. Dr. Smith has been an active participant in open standards development for networking protocols, and decentralized identity. He is also a serial entrepreneur.

References

[1] Buchner, D., Zundel, B. and Riedel, M., "Presentation Exchange," Decentralized Identity Foundation (DIF), https://identity.foundation/presentation-exchange/

[2] Conway, S., Hughes, A., Ma, M. et al., "A DID for Everything," Rebooting the Web of Trust RWOT 7, 2018/09/26
https://github.com/SmithSamuelM/Papers/blob/master/whitepapers/A_DID_for_everything.pdf

[3] "Draft Recommendations Distributed Assurance Statement," Kantara Initiative,
https://kantarainitiative.org/confluence/display/WT/Draft+Recommendations

[4] Ellison, C., Frantz, B., Lampson, B. et al., "RFC 2693 SPKI Certificate Theory," IETF, 1999/09/01
https://www.rfc-editor.org/rfc/rfc2693.txt

[5] Evernym, "Simple Grant Language (SGL),"
https://github.com/evernym/sgl

[6] Hardman, D., "Aries RFC 0103: Indirect Identity Control," HyperLedger Aries, 2019-06-04
https://github.com/hyperledger/aries-rfcs/blob/master/concepts/0103-indirect-identity-control/README.md

[7] Hardman, D., "Why not ZCAP-LD?," HyperLedger Aries, 2019-11-04
https://github.com/hyperledger/aries-rfcs/blob/master/concepts/0104-chained-credentials/contrast-zcap-ld.md

[8] Hardman, D. and Harchandani, L., "Aries RFC 0104: Chained Credentials," 2019-11-04
https://github.com/hyperledger/aries-rfcs/blob/master/concepts/0104-chained-credentials/README.md

[9] Hardman, D., "Peer DID Method Specification," W3C, 2020/08/01

https://identity.foundation/peer-did-method-spec/

[10] Hartzog, W., “Chain Link Confidentiality,” Georgia Law Review, vol. 46 :657, pp. 48, 2012/04/24
https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2045818

[11] Huseby, D., “Authentic Data Economy,” Private Correspondence, 2020/11/24

[12] “IDESG Consent to Create Binding,” IDESG IDENTIRAMA,
https://wiki.idesg.org/wiki/index.php/Consent_to_Create_Binding

[13] “ISO 27560 Consent record information structure (Draft),” ISO,
https://www.iso27001security.com/html/27560.html

[14] Karp, A. H., “Access Control for IoT: A Position Paper,”
https://www.alanhkarp.com/publications/Access-Control-for-IoT.pdf

[15] Karp, A. H., “A Gentle Introduction to Capabilities,”
https://alanhkarp.com/Capabilities-101.html

[16] Karp, A. H. and Li, J., “Solving the Transitive Access Problem for the Services Oriented Architecture,” HP Laboratories HPL-2008-204R1, 2008-11-21
https://www.hpl.hp.com/techreports/2008/HPL-2008-204R1.pdf

[17] Karp, A. H., Haury, H. and Davis, M. H., “From ABAC to ZBAC, The evolution of access control models,” 2009-02-21
https://www.hpl.hp.com/techreports/2009/HPL-2009-30.pdf

[18] Karp, A. H. and Li, J., “Solving the Transitive Access Problem for the Services Oriented Architecture,” 2010 International Conference on Availability, Reliability and Security, 2019/02/15
https://ieeexplore.ieee.org/document/5438113

[19] Kathrein, A., Lizar, M. and Turner, D., “Consent Receipt Specification 1.1.0,” Kantara Initiative, 2019/12/30
https://kantarainitiative.org/download/7902/

[20] “KERI Project DIF,” Decentralized Identity Foundation,
https://github.com/decentralized-identity/keri

[21] Miller, M. S., “Robust Composition: Towards a Unified Approach to Access Control and Concurrency Control,” 2006-05-01
http://www.erights.org/talks/thesis/markm-thesis.pdf

[22] “Object-capbility model,” Wikipedia,
https://en.wikipedia.org/wiki/Object-capability_model

[23] Preukschat, A. and Reed, D., “Self-Sovereign Identity,” 2021.
https://www.manning.com/books/self-sovereign-identity

[24] Reed, D., “The Basic Buildiing Blocks of SSI,” SSI, 2020
https://freecontent.manning.com/the-basic-building-blocks-of-ssi/

[25] Reed, D., “Origin of Use of Verifiable Credentials,” Personal Correspondence, 2020/12/05

[26] “Remote ATtestation proceduresS (RATS) WG,” IETF,
https://datatracker.ietf.org/wg/rats/about/

[27] Ruff, T., “Verifiable Credentials Aren’t Credentials. They’re Containers.,” Medium, 2020-07-14
https://rufftimo.medium.com/verifiable-credentials-arent-credentials-they-re-containers-fab5b3ae5c0

[28] Ruff, T., “Like Shipping Containers, Verifiable Credentials Will Economically Transform the World,” Medium, 2020-07-14
https://rufftimo.medium.com/like-shipping-containers-verifiable-credentials-will-economically-transform-the-world-fece2b9da14a

[29] Ruff, T., “How Verifiable Credentials Bridge Trust Domains,” Medium, 2020-07-14

https://rufftimo.medium.com/how-verifiable-credentials-bridge-trust-domains-97155d0f3c17

[30] Shae, M., Smith, S. M. and Stocker, C., “Decentralized Identity as a Meta-platform: How Cooperation Beats Aggregation,” Rebooting the Web of Trust, vol. RWOT 9, 2019/11/19

https://github.com/SmithSamuelM/Papers/blob/master/whitepapers/CooperationBeatsAggregation.pdf

[31] Smith, S. M., “Kery Event Receipt Infrastructure (KERI) Design,” Github, 2020/04/22

https://github.com/SmithSamuelM/Papers/blob/master/whitepapers/KERI_WP_2.x.web.pdf

[32] Smith, S. M., “KERI.ONE,”

https://keri.one

[33] Smith, S. M., “Meta-Platforms and Cooperative Network-of-Networks Effects: Why Decentralized Platforms Will Eat Centralized Platforms,” SelfRule, 2018/04/25

https://medium.com/selfrule/meta-platforms-and-cooperative-network-of-networks-effects-6e61eb15c586

[34] Smith, S. M., “Universal Identifier Theory,” 2020-10-23

https://github.com/SmithSamuelM/Papers/blob/master/whitepapers/IdentifierTheory_web.pdf

[35] Smith, S. M., “Decentralized Autonomic Data (DAD) and the three R’s of Key Management,” Rebooting the Web of Trust RWOT 6, Spring 2018

https://github.com/SmithSamuelM/Papers/blob/master/whitepapers/DecentralizedAutonomicData.pdf

[36] Foundation, S., “Sovrin: Identity For All,”

https://sovrin.org

[37] “Sovrin: A Protocol and Token for Self- Sovereign Identity and Decentralized Trust,” Sovrin.org, 2018/01/01

https://sovrin.org/wp-content/uploads/2018/03/Sovrin-Protocol-and-Token-White-Paper.pdf

[38] TCG, “Implicit Identity Based Device Attestation,” Trusted Computing Group, vol. Version 1.0, 2018/03/05

https://trustedcomputinggroup.org/wp-content/uploads/TCG-DICE-Arch-Implicit-Identity-Based-Device-Attestation-v1-rev93.pdf

[39] “Verifiable Credentials Data Model,” W3C Candidate Recommendation,

https://w3c.github.io/vc-data-model/

[40] “w3c VC data model discussion Issue #756,”

https://github.com/w3c/vc-data-model/issues/756#issuecomment-735883431

[41] “w3c-ccg Traceability Vocabulary Specification,” w3c-ccg,

https://github.com/w3c-ccg/traceability-vocab

[42] Webber, C. L., Sporny, M. and Miller, M. S., “Authorization Capabilities for Linked Data v0.3 (ZCAP-LD):
An object capability framework for linked data systems,” W3C, 2020/October/22

https://w3c-ccg.github.io/zcap-ld/

[43] Working Group, R. A. T. S., “Network Device Attestation Workflow,” IETF Internet-Draft, 2019/12/03

https://tools.ietf.org/html/draft-fedorkow-rats-network-device-attestation-01

[44] “Zero-knowledge proof,” Wikipedia,

https://en.wikipedia.org/wiki/Zero-knowledge_proof