COVID-19 CREDENTIALS INITIATIVE

Verifiable Credentials Flavors Explained

Kaliya Young, “Identity Woman”

Ecosystems Director of the COVID-19 Credentials Initiative

LF PUBLIC HEALTH

About the COVID-19 Credentials Initiative

The COVID-19 Credentials Initiative (CCI) is an open global community looking to deploy and/or help deploy privacy-preserving verifiable credential projects in order to mitigate the spread of COVID-19 and strengthen our societies and economies.

The community builds on Verifiable Credentials, an open standard and emerging technology, which could offer additional benefits to paper/physical credentials, the most important being privacy-preserving and tamper-evident.

CCI joined Linux Foundation Public Health (LFPH) in December 2020 to work together on advancing the use of VCs, and data and technical interoperability of VCs in the public health realm, starting with vaccine records for COVID-19. We adopt an open-standard-based open-source development approach to public health, which has been proven very effective and efficient with LFPH's work in exposure notification apps.

Slack (#cci) | Email | Groups.io | Newsletter | Twitter | Linkedin | Medium

About Linux Foundation Public Health

Linux Foundation Public Health (LFPH)'s mission is to use open source software to help public health authorities (PHAs) around the world. Founded in summer of 2020, the initial focus of LFPH has been helping PHAs deploy an app implementing the Google Apple Exposure Notification (GAEN) system. LFPH just brought in CCI to take lead on creating interoperable standards for sharing pandemic-related health data. As the organization grows, LFPH will be moving into other areas of public health that can take advantage of open source innovation.

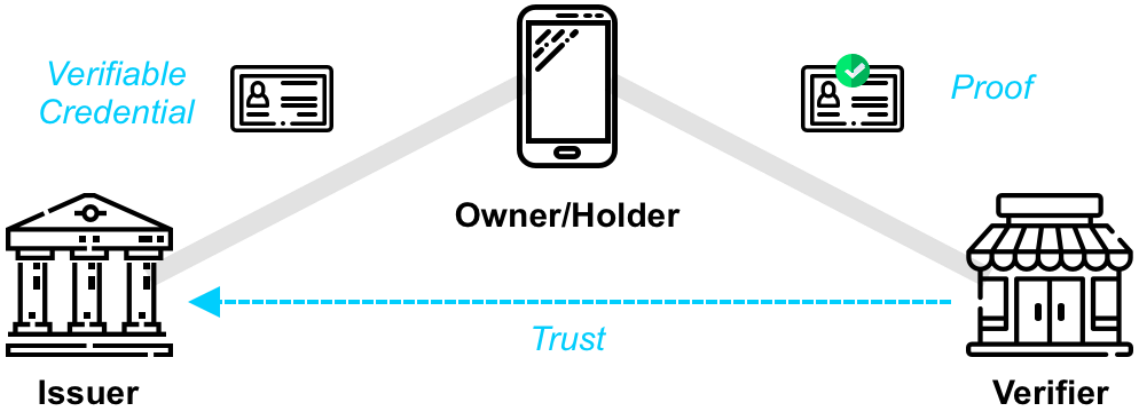
Slack | Email | Twitter | Linkedin

Acknowledgement: Thanks John Kelly, Markus Sabadello, Tobias Looker, Ivan Herman, Daniel Hardman, Kristina Yasuda, Manu Sporny, Suzana Maranhao, and Juan Caballero for reviewing this paper.

Table of Content

INTRODUCTION

Verifiable Credentials (VC or VCs) is one of the key standardized components of decentralized identity.



The VCs Data Model, defined at the W3C, is a universal data format that lets any entity express anything about another entity. It provides a common mechanism for the interoperable implementation of digital credentials that are cryptographically secure, tamper-evident, privacy-respecting, and machine-verifiable. A common standardized data model enables standardized credential packaging, cryptographic signing, and proof expression. This then creates a VC ecosystem with interoperable credentials, allowing credentials to be processed and understood across and between disparate systems.

This paper explains the differences between various VC flavors for the technically inclined readers. An important goal of the paper is to present a clear path forward that can end the interoperability trilemma that has emerged between three different VC flavor choices, and points to the adoption of the newest VC format to satisfy all stakeholders.

Why does this matter? **Without convergence to a standardized VC format, there won't be functional interoperability across the ecosystem.** If application builders process diverse cryptographic calculations using incompatible libraries and methods, and the underlying data has varying readability properties, then interoperability will not be achieved.

What different methods have in common is issuers use them to package up claims about an entity. The issuing entity then uses cryptography to seal the credential. This sealing provides a mechanism for other entities (the verifiers) to check the cryptographic signatures to see if the credential has integrity based on the issuer's public keys.

What is different is how issuers format claims inside the credentials and use cryptographic signatures suites to sign the information and seal the credential. These methods also have processes for sharing Verifiable Presentations (VP or VPs) to a verifier and conducting the cryptographic verification.

UNDERSTANDING FLAVORS OF VERIFYING MECHANISMS IN VCS

Below are the three primary flavors of VCs discussed in this paper. All have more than one critical implementation in various stages of production. There are advocated variations of these types, but they are less common.

- **JSON-LD** family with **LD Signatures** or with **BBS+ Signatures that enable Zero Knowledge Proofs (ZKP or ZKPs)**
- **JSON with JSON Web Signatures**, precisely in the form of a JSON Web Token (JWT)
- **ZKP with Camenisch-Lysyanskaya Signatures (ZKP-CL)**

There are essentially two different “types” of VCs that are unique from one another because of how they elect to express the VC data model, including the associated cryptographic material (e.g., digital signatures).

- VCs expressed in JSON-LD using Linked Data Proofs (**JSON-LD**)
- VCs expressed in JSON using JSON Web Signatures (JWS**)**, precisely in the form of a JWT (**JSON-JWT**)

These two formats support a range of different digital signature algorithms - a vital property to ensure VCs are cryptographically agile, enabling them to move with the constant evolution of cryptography. With both of these formats, users can implement a whole host of different digital signature schemes.

It is important to note that not all digital signature algorithms are the same. Security and capabilities can differ. Some support novel properties such as selective disclosure and anti-correlation techniques via ZKP technology. BBS+ Signatures is one such algorithm that developers leverage with the JSON-LD data format.

The third credential format and signature scheme mentioned in the VC Specification is “Camenisch-Lysyanskaya ZKP [CL-SIGNATURES],” however this format is not explained in the specification. This format is a bespoke data expression format coupled with a particular digital signature algorithm. The company Evernym and then the Sovrin Foundation originally developed this format as part of the Indy codebase which they contributed to Hyperledger, a Linux Foundation project hosting many different open source enterprise-focused blockchain projects. This codebase was forked into the Hyperledger Indy project for ledger code, Hyperledger Ursa for cryptographic libraries, and the ledger-agnostic Hyperledger Aries codebase for agent, wallet, and credentialing code.

Two of these credential formats leverage JSON-LD. In recent years this format has garnered a lot of interest for good reason. It supports semantic clarity and discernment by verifiers about the issuer’s intended meaning for particular attributes in the credential. This format also has a handy feature supporting translation between languages on the fly. It is also new, and because

of this, fewer developer tools support it than JSON. Last year, a new format **JSON-LD ZKP with BBS+ Signatures** came forward to bridge the gap between JSON-LD and ZKP-CL. MATTR, a New Zealand-based company seeking to bridge the differences, introduced this method of packaging and expressing VCs. It uses the simplest, widely available cryptography to support selective disclosure. MATTR shared the new format with the VC community as an open standard. Both the ZKP-committed groups and the JSON-LD community have expressed interest in adopting it. According to its designers, JSON-LD ZKP BBS+ should not be seen as a different format of VC. It shares much of the processing logic for JSON-LD, even though it supports a new set of fundamental types that enable new functions like derived proofs.

A Metaphor for the difference between JSON and JSON-LD

The second main format JWT is a part of the JOSE framework. It provides no means to support semantic disambiguation but has the benefit of being simpler for assertion format implementations.

"JOSE is the rice and beans of cryptography standards. It's got everything you need to survive and is easy to make, but its extensibility model guarantees you will be eating rice (base64url) and beans (JSON) forever. That might make you "fat" because base64url inflates JSON.

On the other hand, Linked Data Proofs, as seen in Verifiable Credentials and Verifiable Presentations, are like a pharmaceutical drug - really hard to build, but capable of solving all kinds of problems and formally described by an information-theoretic model (molecular formula ~= RDF). Linked Data Proofs are capable of working with other bases, other structured data formats (base58, CBOR), and the extensibility model is anything that you can model in RDF.

Context determines the relevance of either model. Most people don't go to a pharmaceutical lab to make lunch, but most people who make drugs in their kitchen also eventually end up sick."

- Orie Steele, Transmute Industries

The first half of this paper explains a scenario of the lifecycle of a VC and the differences in some stages at a high level. The second half of the paper expands on the differences between JSON & JSON-LD and the two different strategies to achieve ZKP functionality, ZKP-CL and JSON-LD ZKP with BBS+ Signatures.

Public-Private Key Cryptography (Simplified)

Explaining public-private key encryption, the “fancy math”, is not in scope for this paper. For those trying to understand why it matters and how it works, here are some simplified explanations that will help.

There is a special relationship between public keys and private keys. An entity uses algorithms to generate a new key pair. They can publish the public key and keep the private key - well ‘private’, shared with no-one. Because of this special relationship - a person can prove they control a private key associated with their public key without ever sharing it. They prove they control their private key by responding to a cryptographic challenge sent to them that uses their public key. When they use their private key to respond to the challenge (by calculating a number and sending it back), the entity challenging them knows they actually control the private key - based on the response - but they still don’t learn the private key information. It is like a super-secret one can prove possession repeatedly without giving it away. As a way to prove control of an identifier, this is much better than a password.

These properties of public-private key pairs are used throughout the exchange of VCs. Verifiers use these properties to know that an issuer signed a particular document with their private key - because the verifier has access to the associated public key and can calculate the verification. These two numbers, the public key and private key, have a special relationship that is leveraged when doing the verification, so it is provable that the issuer signed the credential - and that the contents of the credential have not been altered since signing. Cryptographic signatures are one of the key enabling technologies of VCs.

Another key enabling technology is ZKP, which uses more advanced math to prove things are true without revealing all of the underlying information. In the context of VCs, ZKPs can provide verification that an entire credential has not been tampered with, even if only a subset of its contents is shared.

The Journey of a VC

To help understand critical non-trivial technical differences between these types of VC, this section will walk through the various scenes of a credential’s lifecycle. This will help clarify where these credential types cause different processing to happen in their lifecycle.

Scene 1

In this scenario, the issuer is an educational institution. It will issue to its students a VC representing the completion of their degree requirements, post graduation. The institution (the issuer) decides what should be in this particular type of credential, and attributes will have values expressed. Here are some examples that might be in such a credential:

“First_name”
“Last_name”

"Degree Issued"
"Date Earned"

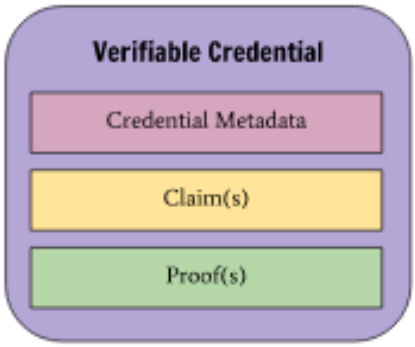
The issuer defines the credential and then expresses schema structure and attribute values in the different formats.

JSON-LD family creates an @context file that references RDF-defined schemata published via web-based open-data registries and "ontologies".

JSON-JWT defines each credential's semantics according to terms in the IANA-registered global schemata for JWT (which is very small). For name values that are not in the IANA registry, they can assign it a public name (i.e., a URI), [or] give it a local name (i.e., any string)." The relying party (verifier) is left guessing about the meaning, and if they don't want to guess, they must find an out-of-band way to connect back to the issuer to figure out the meaning.

JSON-ZKP-CL defines each credential's schema using a JSON array of field names. In Indy-based systems, field names are registered to the same DLT which anchors identities (Adding an "@context" at the top could make them interpretable as JSON-LD, but there's really no added value since the @context definition helps find schemata on the open web, while Indy credentials are defined on-chain). Each issuer of a credential must also post a credential definition that says to the world, "I intend to publish credentials from schema X, signed using keys Y[0]...Y[N], and with revocation strategy Z"

All VCs contain Credential Metadata, Claims and Cryptographic Proof(s).



Source: The Verifiable Credentials Data Model

Scene 2

The issuer is going to sign the credentials it issues cryptographically. This means the issuer needs to generate a public-private key pair and use the private key to sign the credentials. Issuers need to share their public keys widely. One common way to do this is to leverage Decentralized Identifiers (DID or DIDs), being defined at W3C, and their associated DID documents. The issuer creates a DID and associates a public-private key pair with it. They then create a DID document, sign it, and post it to a distributed ledger.

There are currently over 80 DID methods to create, read, update and delete DIDs & DID documents on ledgers and in other forms. *(This paper is not about the wide range of choices; suffice it to say, they exist).* It is possible to anchor issuance identities of VCs to conventional or centralized cryptographic trust mechanisms, leveraging X.509 certificates, or by reference to web-based DIDs that get resolved via the open web using the well-Known DID method (so one doesn't have to use a blockchain).

Scene 3

Who the heck is the Issuer? *This is where things can get philosophical really fast, but it doesn't have to.* There are already systems in place in our contemporary world to evaluate what things are true or not. For example, in higher education, accreditation systems evaluate existing institutions and accredit them. These systems come into play in our emerging digital world too. An accreditation body can issue a publicly viewable VC to an educational institution. The institution would also have incorporation paperwork from the jurisdiction where it operates. Accreditation could be issued to the educational institution as a publicly available VC. An educational institution is anchored in relationship to other institutions who have knowledge of it and authority to assert information about it. There might also be broader agreements about issuing educational credentials called "governance or trust frameworks" that this institution has signed. New networks now anchor VCs to existing governance networks and the authority of identifiers. Examples include the work of Global Legal Entity Identity Foundation (GLEIF) and GS1, the entity that manages product codes internationally. GS1 is the driving force behind much of the Linked-Data semantic ontology harmonization that JSON-LD credentials build on.

So, in this scene the issuer is contextualized in the broader framework of meaning, truth, and authority that typically already exists. Technical procedures express this authority digitally so that verifiers can figure out if they should have confidence in the statements by the issuer of a particular credential.

Scene 4

The issuer and the holder are the first two actors. It is assumed that in this case the issuer has a trusted relationship with the holder that authorizes them to issue a credential, e.g., a school issuing a degree to a graduate.

The holder who will receive the credential has to share an identifier with the issuer that it can use to anchor the credential. Different VC formats will implement this differently.

> **JSON-LD** anchors credentials to DIDs resolvable by a published DID method and may be anchored in a DLT.
>
> **JSON-LD ZKP with BBS+** anchors credentials to DIDs and includes a mechanism for binding to a public key (connected to a private key) which is not shared with a verifier and does not reveal the DID (similar to the feature currently used in the JSON-ZKP-CL system).
>
> **JSON-JWT** anchors credentials to DIDs, which may be anchored in a DLT.

JSON-ZKP-CL anchors credentials to long numbers that contain “link secrets,”. It is possible to link ZKP-CL credentials to DID by including a DID for the holder in the credential’s schema. It is also possible to anchor ZKP-CL credentials to a biometric.

Scene 5

Once the issuer has the holder’s cryptographic anchor for the credential, they use it as the basis for issuing the credential. The issuer assembles all the information it wants to enclose in the credential, including the information for each attribute articulated in the schema defined in Scene 1. The holder with a wallet will leverage these bundles of claims in a credential to prove to verifiers that the claims apply to them; the claims were made about them (or others) by the issuers. The issuer prepares the credential using different methods before sending it to the holder.

JSON-LD takes the attributes, dereferences (i.e., “fetches”) and expands them, then canonicalizes them (puts them in alphanumeric order), and finally signs the whole credential with LD Signatures.

JSON-LD ZKP with BBS+ takes the attributes, canonicalizes them, and then signs each of the individual attributes with BBS+ Signatures.

JSON-JWT creates a JSON object containing the attribute-value pairs in any order, and signs the entire object using JWT.

JSON-ZKP-CL generates a numeric representation of each field (useful for predicates) and then signs both the numeric and the text representation of each of the statements using a CL Signature.

Scene 6

The issuer and the holder have to connect in some way to pass the credential. There are several options currently in use. This paper is not about these options and it isn’t a settled question - options including DIDComm, pairwise connection using JWT, SIOP CHAPI and QR codes or other form factors like smart cards. Some systems will create a “pairwise” (single-use) DID for each new communication channel - so in the following scenes, some of these DIDs might be contextual, and others might be static.

Scene 7

The holder now has the VC in their digital wallet (or another form factor). The wallet needs to have the capability to receive and do calculations with the VC to create a VP.

Scene 8

The holder is interacting with a verifier and wants to prove that it has possession of a certain VC or, in the case of ZKP approaches, prove a set of assertions made about a subject by a particular issuer. The holder or their wallet must get a request from the verifier about what type

of credential it is looking for. This is done via a protocol under development called Presentation Exchange. One group composed of people from all the different VC format types is working on this and aligning on one way to do this.

Scene 9

The holder wallet/agent must perform specific calculations on the VC to get it ready to share in a VP. All of these VC flavors need a nonce from the verifier to prevent replay attacks.

JSON-LD When sharing, the VC is not modified, but instead placed in an encapsulating VP (which also contains the nonce) and then digitally signed by the holder. This encapsulation effectively states: "I, the holder, am sharing a VC, and the nonce you gave me... and I'm bundling those up into a VP, digitally signing that with the keys declared in my public DID document (so you know it's me) and am sending it over to you."

JSON-LD ZKP with BBS+ The holder's wallet can select some of the elements of the credential they are presenting. Holder's wallet/agent takes the signature for a given attribute and generates a presentation that they give to the verifier.

JSON-JWT The VC in the holder's wallet is not modified, but encapsulated in another JWT, which is signed by the holder in preparation for sharing it with the verifier. This encapsulation has a similar effect as with JSON-LD (see above).

JSON-ZKP-CL The holder generates a new, never-before-seen credential (wrapped inside a VP) that combines and reveals whatever attributes from issued credentials are requested, plus any predicates (assertions like "date_of_birth is more than 18 years ago), and hides everything else. The "proof" block of this new VC is not a traditional digital signature, buta mathematical demonstration that the holder indeed possesses VCs signed by the requisite issuer(s), containing the disclosed attributes, and having the stipulated schema(s). The proof also demonstrates that the issuer has not revoked the credentials and that they are bound to the holder because the holder knows the link secret(s) that were used at issuance.

Scene 10

The holder must get the VP to the verifier. Like the one in Scene 6, it involves the holder having to connect to the verifier. This is an unsettled area - current options include DIDComm, pairwise connection using JWT, SIOP CHAPI and QR codes or other form factors like smart cards. Some systems will create a "pairwise" (single-use) DID for each new communication channel - so in the following scenes, some of these DIDs might be contextual, and others might be static.

Scene 11

The verifier then takes the information it gets from the holder in the form of a VP and does calculations with this information. It has to verify (cryptographically) that the VP is valid. In all of these formats, the verifier looks up ("resolves") the issuer's DID and finds its public key. It does a validation check on the presented attributes using the issuer's public key.

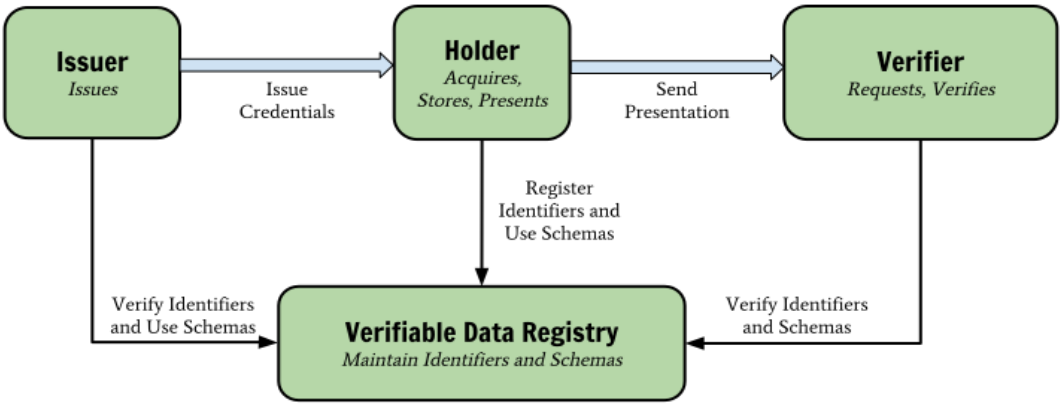
JSON-LD takes the holder's identifier (from the VP) and the issuer's identifier (from the VC embedded in the presentation), as well as the public keys associated with those identifiers. The verifier now verifies that the signature contained in the VC was created using the issuer's private key, and the signature contained in the VP was created using the holder's private key. The verifier can now be certain that the VC came from the issuer, and it was presented by the holder. Several additional checks are performed, such as checking if keys are expired or have been revoked.

JSON-LD ZKP with BBS+ The verifier finds the public keys related to the signatures and verifies that the proofs of signatures match along with the information presented: it 1) was not tampered with, 2) came from the issuer, 3) passes a validation check on the presented attributes, and 4) proves that it has knowledge of the holder by providing “a piece of crypto” that they can decrypt without exposing an identifier related to it.

JSON-JWT Same as JSON-LD (see above)

JSON-ZKP-CL The verifier has received the VP. This presentation can contain attributes from more than one credential. For each attribute shared, the verifier looks up the credential schema it came from along with the DID/DID document of the Issuer. It uses these two pieces of information to verify the attribute presentation. Each attribute statement in the VP needs to go through this process. The verifier can be assured that all of the attributes were issued to the holder of the same link secret.

The above 11 scenes walk through all the steps that happen in the below diagram while articulating the differences in how the various VC formats approach it. In two of these VC methods (JSON-LD and JWT) the holder anchors a DID to a ledger and it is used by the issuer as the “id” property in the VC (in the diagram this is represented by the arrow going from the holder box to the Verifiable Data Registry) for the VC issuance. The other two (ZKP-CL and JSON-LD ZKP with BBS+ Signatures) generally do not.



Source: [The Verifiable Credentials Data Model]

Exploring the difference between JSON and JSON-LD

Here is what it says in the VC data model specification:

> *"Digital proof mechanisms, a subset of which are digital signatures, are required to ensure the protection of a Verifiable Credential. Having and validating proofs, which may be dependent on the syntax of the proof (for example, using the JSON Web Signature of a JSON Web Token for proofing a key holder), are an essential part of processing a Verifiable Credential. At the time of publication, Working Group members had implemented Verifiable Credentials using at least three proof mechanisms:*
>
> - *JSON Web Tokens [RFC7519] secured using JSON Web Signatures [RFC7515]*
> - *Linked Data Signatures [LD-SIGNATURES]*
> - *Camenisch-Lysyanskaya Zero-Knowledge Proofs [CL-SIGNATURES].*
>
> *One of the goals of this specification is to provide a data model that can be protected by various current and future digital proof mechanisms. Conformance to this specification does not depend on a particular proof mechanism; it requires clearly identifying the mechanism a Verifiable Credential uses."*

What this format does is supporting understanding between different systems as it says in the specification:

> *"When two software systems need to exchange data, they need to use terminology that both systems understand. As an analogy, consider how two people communicate. Both people must use the same language and the words they use must mean the same thing to each other. This might be referred to as the context of a conversation."*

The authors of the first iteration of this specification were experienced with JSON-LD and leveraged this format when developing VCs. It has some friendly properties that make it understandable in a wide range of contexts because it assumes an open world.

> *"A context in JSON-LD works in the same way. It allows two applications to use shortcut terms to communicate more efficiently but without losing accuracy."*

Both of these formats, JSON and JSON-LD, create objects that are processed and then signed.

What is JSON?

So, let's start with JSON (JavaScript Object Notation).

JSON is a widely used open standard object and document format used for data interchange and is human readable. It consists of attribute-value pairs.

Here is an example of JSON (from the JSON Wikipedia page)

```
{
 "firstName": "John",
 "lastName": "Smith",
 "isAlive": true,
 "age": 27,
 "address": {
  "streetAddress": "21 2nd Street",
  "city": "New York",
  "state": "NY",
  "postalCode": "10021-3100"
 },
 "phoneNumbers": [
  {
   "type": "home",
   "number": "212 555-1234"
  },
  {
   "type": "office",
   "number": "646 555-4567"
  }
 ],
 "children": [],
 "spouse": null
}
```


There are several different data types and structures supported in JSON. From the example above:

- Number
  - Age 27
- String: a sequence of zero or more Unicode characters.
  - firstName - “John”
- Boolean: either of the values true or false
  - Is alive = true
- null: an empty value, represented by the string null
  - No spouse
- Array: an ordered list of zero or more values, each of which may be of any type.
  - phoneNumbers - represents an array
  - There is more than one phone number listed, so it is represented as an array
  - children - represents an empty array
- Object: a collection of name-value pairs where the names (also called keys) are strings. Objects are intended to represent associative arrays, where each key is unique within an object. Objects are delimited with curly brackets and use commas to

separate each name-value pair, while within each pair the colon ':' character separates the key or name from its value.
- Address - an object consisting of four different name-value pairs.

What is JSON-LD?

One of the main goals of JSON-LD was to make it very easy for developers to interpret JSON objects according to immutable and open data registries rather than local schemata, which can be separated from the data they are essential to interpreting. It was created for web developers working with data that must interoperate semantically across the Web.

What distinguishes JSON-LD?

JSON-LD supports an additional layer of context to map the name part of the name-value pair to an RDF ontology.

What is an RDF ontology? It is a knowledge graph that structures metadata in shared vocabularies so that when machines read it they can understand more about what it means.

Here is JSON-LD being used to describe the movie Avatar.

At the top in the @context field it references the place where one can look up more about what the terms below mean so in the schema.org context there is a type called “Movie” and they can then look up what are all the different terms for this type.

```
{
  "@context": "http://schema.org/",
  "@type": "Movie",
  "name": "Avatar",
  "director":
    {
      "@type": "Person",
      "name": "James Cameron",
      "birthDate": "1954-08-16"
    },
  "genre": "Science fiction",
  "trailer": "../movies/avatar-theatrical-trailer.html"
}
```

Why does this help us with VC?

This method allows credential issuers to define a type of credential they are issuing and then use the @context field for navigating a tree of schemata that explain the structure and content of a credential. The @context field is an address (URL) which also helps place the credential in

its original context. This helps the verifier confirm the meaning of all the “attributes” of the “attributes-value pairs” that are listed on the credential.

JSON-LD supports disambiguation between properties found in different credentials issued by different issuers and so verifiers understand what they are looking at.

One of the features that is valuable for a multilingual world is the ability to map property values to another language via a language map. So, when a credential is published in one language, English, and is presented to a verifier who is familiar with another language, Thai, the program can go look up the values of the claims in the document and have them accurately translated so they can be read by the verifier.

This means that definitions can be expressed in different languages and translated.

In addition to all the features JSON provides, JSON-LD introduces:

- a universal identifier mechanism for JSON objects via the use of IRIs,
- a way to disambiguate keys shared among different JSON documents by mapping them to IRIs via a context,
- a mechanism in which a value in a JSON object may refer to a resource on a different site on the Web giving the ability to annotate strings with their language,
- a way to associate data types with values such as dates and times,
- and a facility to express one or more directed graphs, such as a social network, in a single document.

What is JWT Claim?

JWT takes a different approach to determining the meaning of claim terms in credentials. There is an IANA registry for JWT claims as a first place to look for JWT claim definitions. If the claim name isn’t in the IANA register, then the claim can be given a “give it a public name (i.e., a URI), [or] a local name (i.e., any string)”. The meaning of the terms is decided between the issuers and verifiers.

In the VC Implementation Guidelines, there is a long list of the different characterizations of methods: JSON with JWT’s support vs JSON-LD with LD Signatures, Benefits of JWT, Benefits of JSON-LD and LD-Proofs.

To summarize the most salient points:

JSON is an older standard, officially recognized as a standard in 2013. JSON-LD 1.0 was formally standardized in 2014. The version and standard was updated to JSON-LD 1.1 and ratified in 2020.

That being said, one can use JSON libraries to process JSON-LD objects/documents, and conversely, interpret JSON documents as JSON-LD by providing a context.

Both of these formats have serialization and deserialization processes. These involve taking a document format that is human readable and turning it into a string (eliminating the spaces and indents for example) so that it is transportable over the internet. This serialized form can also be signed, although some formats (such as JSON-LD), must first be processed by a canonicalization algorithm before the signature can be calculated.

JSON-LD canonicalizes the attribute-value pairs in a predictable way based on the information model being processed.

For example, if someone sends a VC with the fields "cat, dog, immunization," and someone else sends "immunization, cat, dog"... when both of them canonicalize, they'll be in the same order, for example: "dog, cat, immunization." This is the sorting of the "information model."

Because of the way JSON-LD canonicalization works, one can express the same information in a variety of syntaxes (JSON-LD, CBOR-LD, TURTLE, RDF/XML) and a digital signature made in one of those syntaxes can be reused across them all. If this were not true, they would have to create a digital signature for each format, forcing entire ecosystems to pick one syntax (which is rare across multiple industries, as many adopt other formats such as JSON, CBOR, or XML).

Neither of these formats easily supports selective disclosure and ZKPs, but both claim they can. If an issuer puts each attribute in its own VC, then this is possible. It isn't possible for a holder to separate out VC claims when they are bundled together within one VC.

Both JSON-JWT credentials and JSON-LD LD-Signature credentials require that the holder share the whole credential with verifiers - there is no partial share/show. This makes it harder to support selective disclosure.

Issuers can choose to package up credentials with claims such as "over18" or "over21" and package up atomistic claims, - a credential with just one claim and no other.

Another issue is that both JSON-JWT and JSON-LD LD Signatures rely on the credential having a DID registered in a public registry as the anchor for the credential. Every time a holder shares the credential this identifier is shared and thus, each presentation of the credential means that it is possible, if verifiers are colluding, to see where the holder presented their credential. One way around this is to have issuers re-issue credentials or to have a trusted third party vouch for the contents of credentials without having to reveal them directly to verifiers.

Understanding Choices for Selective Disclosure

Selective Disclosure

Today in real life, one presents documents that contain their actual birth date and humans check that date to determine if they are over 18 or 21 or not.

Selective disclosure means one can hide some of the attributes of a credential, selectively revealing them for presentation as required.

There are everyday things that people want to prove, like an assertion of age “over 18” or “over 21” for age restricted products (cigarettes) or places (bars).

Some forms of selective disclosure involve calculating the values based on predicates (ZKP-CL). Current implementations of JSON-LD ZKP with BBS+ Signatures do not support predicates (such as “greater than”, “less than”, “equal to). Instead, the issuer can enclose values in the credential e.g., “over 18” that the holder can disclose without sharing the attribute containing their birthdate. BBS+ Signatures can support predicates from a cryptographic standpoint, though current implementations have opted for the simpler approach of per-attribute disclosure.

One may also want to withhold fields like physical addresses. When they present physical credentials in real life situations, they can’t do that withholding; all the data available on the document is revealed. In the digital world, they can selectively share attribute fields satisfying the verifier’s specific requirements.

This feature of digitized credentials makes them better for privacy protection by providing individuals with granular control over the information shared with various institutions. When implemented properly, selective disclosure provides robust data minimization.

ZKP

ZKP are a type of cryptographic math that lets one prove things are true without actually revealing the information. So, it is technically how one generates a selective disclosure proof for example, “age above 21”, from an actual birth date.

There are several different metaphors put forward by the cryptographers to explain how the math works. Below are a few examples helpful in understanding the technology.

- Where is Waldo
- The Ali Baba Cave
- Two Balls and the color blind friend

ZKP-CL

This credential format was created specifically to leverage the CL Signatures.

JSON-JWT and JSON-LD Signatures each have their own way of representing the meaning of the attributes within a VC. JSON-JWT references an IANA registry and assumes a “closed world” authority model based on that authoritative registry. JSON-LD Signatures use an @context field to reference existing RDF mapping to known dictionaries and assumes an open world model where new terms and definitions can easily be introduced.

Where are the schemas for ZKP-CL? The schemas are defined in a document and they are written to an Indy ledger where they can't be changed. To update a schema, a new version must be created and written to the ledger for future use.

This schema is fetched from the ledger by verifiers when calculating the verification.

The underlying cryptography allows for calculations on what is signed sharing key aspects of some but not all of the information. One can use a birth date in the credential to calculate that a person is over 18 or 21 or 65.

So how are all the elements in a ZKP-CL credential defined?

The ZKP-CL processing model requires that the credential's schema is defined and posted to a ledger. The credential issuer follows the pre-defined schema. They leverage the CL Signature suite to sign each statement in the credential.

The credential is anchored to a link secret that is known only to the holder (stored in the holder's software), and when the holder is issued a credential, it packages up a cryptographic commitment to the link secret within another long number that the issuer uses for the credential ID.

A metaphor for this is that of a digital watermark; the link secret is like a stamp that is used to create a watermark. The stamp used to create the watermark is NOT packaged up or put in the credential in any form; all that is in the credential is very-hard-to-fake evidence that the holder possesses the stamp and is capable of generating such watermarks.

Unlike non-ZKP methods, zero-knowledge methods generally do not share a correlatable identifier (such as a persistent or public DID) and also do not reveal actual signatures. Instead, they only reveal a cryptographic proof of a valid signature. This means the holder of the credential cannot be correlated just by presenting a proof of the credential (this is the primary privacy problem with public/private key cryptography that ZKP-CL Signatures were invented to solve over a decade ago). All other non-ZKP Signature formats require a resolvable DID—which is a correlatable globally unique identifier—and also produce the same signature on all equivalent presentations, which is a second fully correlatable globally unique identifier.

JSON-LD ZKP with BBS+ Signatures

This approach to VCs is innovative because it brings together different elements from several of the existing approaches to VCs. It is based on the usage of BBS+ JSON-LD Signatures, which is a subclass of LD Signatures, in combination with a JSON-LD credential schema. The cryptography behind this mechanism is BBS+ Signatures, which require what's called a pairing-friendly curve. Existing implementations in the VC ecosystem use the elliptic curve BLS12–381.

By leveraging the technology of JSON-LD with a specific set of cryptographic key types and algorithms, this mechanism is able to produce a VC that can be used to generate proof presentations and ZKPs that selectively disclose attributes of the credential.

Attributes from multiple different BBS+ credentials can be combined into a single privacy preserving credential presentation in a way that's similar to other kinds of VCs, with the added ability of selective disclosure. JSON-LD BBS+ credentials share the same format provided by JSON-LD for structure and processing the information inside of the credential. The @context is used to clearly articulate the meaning of the "name" parts of the "name-value" pairs. This allows the credentials to retain their semantic expressiveness while inheriting the security properties of Linked Data Proofs, which include the ability to represent proof chains and build integrity and trust on top of open data vocabularies such as schema.org.

BBS+ Signatures are a kind of multi-message digital signature. Traditional digital signatures allow one to use a public + private keypair to produce a digital signature over a message (virtually any kind of data for which one wants to establish integrity). The drawback of this method is that they can typically only produce a signature over the entire message, or in this case the entire credential at once. This is a serious limitation in that any proof of the digital signature will have to prove that the entire credential was signed, without the capability of selectively disclosure applied to the credential.

Multi-message digital signature schemes (like BBS+ and CL-Signatures) are able to sign an array of messages, rather than simply a single message. The same mechanism is used when a private key produces a digital signature over a message one wishes to sign, but now they have the flexibility to break a message up into its fundamental attributes. Since each message corresponds to a claim in the credential, creating credentials in this manner gives them the ability to derive and verify a proof of the digital signature over a subset of credential attributes. This process allows the credential holder to choose which messages they wish to disclose in the proof and which messages they want to keep hidden. The derived proof indicates to the verifier that they know all of the messages that have been signed, but that they are only electing to disclose a subset of these messages.

While BBS+ Signatures handle the cryptographic side of the interaction, JSON-LD manages the semantics and schemas associated with the data. In order to select a subset of credential attributes from the original credential to create a ZKP, users apply an aspect of the JSON-LD standard known as JSON-LD framing. JSON-LD frames allow the user to describe how they would like to transform or modify an input document, in this case a VC. It provides a syntax for specifying which attributes of a credential the user wants to include in a proof, and BBS+ Signatures uses this array of signed attributes to handle the rest. The benefit of using this approach is first and foremost that it is interoperable with existing schema technologies and credentials using JSON-LD, and by extension, is fully compliant with the VC specification as it exists today. In addition, because its signatures and proofs are self-describing and self-contained, they don't require any extra setup or external dependencies such as a credential definition, which is stored externally, e.g., on-ledger. This approach allows for the simple and standardized usage of JSON-LD to leverage open data vocabularies while retaining the privacy-preserving features such as selective disclosure and ZKPs, which have traditionally come with their own set of limitations and trade-offs. The choice is no longer standards-based interoperability *or* privacy-preserving cryptography -- one can have both.

CONCLUSION

This paper explained the differences between the different flavors of VCs for technically inclined readers. It elaborated on the differences between JSON and JSON-LD and articulated differences between the two different implementations of ZKP style credentials. The ‘Journey of a VC’ section articulated all steps where VCs are active and highlighted the differences in how different VC flavors ’behave’. The newest VC flavor JSON-LD ZKP with BBS+ is being well received because it provides a way to use JSON-LD and provide ZKPs. Until this innovation it was not possible to do both. Hopefully, this paper has made clear the substantive technical differences that exist between VC formats and can lead decision makers towards convergence. Without convergence, there won’t be functional interoperability across the ecosystem.

If you have any feedback or questions about the paper, please contact:

Kaliya Young
Ecosystems Director
COVID-19 Credentials Initiative (CCI)
kaliyay.cci@lfph.io

If you have general inquiries about CCI, please contact:

Lucy Yang
Community Director
COVID-19 Credentials Initiative (CCI)
lucyy.cci@lfph.io