VC Data Integrity Tutorial

Dr. Greg M. Bernstein and others?

Table of contents

1 Introduction

This document provides a tutorial introduction to W3C verifiable credentials to allow non-profit school or community clubs to cooperate with each in a timely and privacy enhancing way via the issuance and verification of *verifiable credentials*. We will illustrating the use of the following W3C **specifications**:

- Verifiable Credential Data Model 2.0 [1]
- Verifiable Credential Data Integrity 1.0 [2]
- JSON-LD-1.1 [3]

In addition we will go through the decision process in selecting the appropriate **cryptosuite** from among:

- Data Integrity EdDSA Cryptosuites v1.0 [4]
- Data Integrity ECDSA Cryptosuites v1.0 [5]

1.1 Cooperating School/Community Clubs Use Case

Suppose that we have two community hiking clubs (purely fictitious): *The Arcata Tall Trees Hiking Club* and the *White Mountain Old Trees Hiking Club* as shown below. Both clubs exists to promote the "great outdoors", hiking safety, and comraderie.


When teaching begining and intermediate website development to computer science majors one of the authors of this tutorial would have students create a website for a real or fictional school or community club of their choosing, such as a hiking club, a guitar club, a cooking club etc… They would develop this website from a simple static HTML page to a full front-end/back-end server based application with logins for club members and club officers with different views and club management capabilities based on membership/officer status. A partially complete example club is the Bay Area Windsurf Foiling Club. This tutorial goes one step further and adds the ability to issue verifiable credentials to its members so that they can participate in someway with a different club that has some kind of reciprocity agreement.

To make the club interesting from a credential perspective the club should certify knowledge or skills of a member. We assume the club has a website and member data base of some sort. Clubs can get quite elaborate with loaner equipment for members, events (open to all, open only to members, members paid, etc.), rating system, required volunteer work, etc. An example of such a club is the Cal Sailing Club a university club that morphed into a community club which has an extensive training program, collection of boats and windsurfers, and runs a number of events every year.

1.1.1 Why Bother with Verifiable Credentials?

Hiking club example. Member status. Knowledge certification. Skills certification. Each club has separate member databases which for privacy reasons do not share with the other club. If only a single club then a "hike leader" could check the club database. Other mechanisms could include one club sending email to another or calling the other. These are not as timely as a current credential and sacrifice privacy. "Now we know that he/she is participating with that other club…"

2 Club Credential Specification

In this section we Start with domain specific knowledge of the club, review the popular JSON data exchange "language" and tools, bring in basic structure from the VC Data Model 2.0 [1], and then add a custom "context" to define our club's credential.

2.1 Club Specific Domain Knowledge

As we will see in the following sections that the verifiable credentials data model 2.0 [1] provides a well thought out structure for credentials of any type. However, domain knowledge about the club is needed to actually define the details of the credentials.

In this section I'm going to assume a wind sport club that includes multiple disciplines (sailing, windsurfing, etc.) since at one time I actually was a member of such a club.

One of the first questions that comes up is how best to organize the credentials for the club? Given that this wind sport club has many disciplines: (a) dinghy sailing, (b) keel boat sailing, (c) windsurfing, (d) windsurf foiling, (e) wing foiling; should we have one credential with lots of properties, or lots of separate credentials, or something in between, e.g., membership credential and sport specific credentials? Since verifiable credentials are kept in digital wallets rather than physical wallets, more credentials from the same issuer isn't necessarily a big problem. For example the Open Badges Specification Spec Version 3.0 based on VC DM 2.0 emphasizes a single *achievement* (see Open Badges Overview). While the Comprehensive Learner Record Standard Spec Version 2.0 features a list of *achievements*, e.g. all the courses completed on the way to a academic degree.

My inclination is to have credentials specific to the club's distinct disciplines allowing for some redundancy if someone earns credentials in multiple disciplines. Here are other notions:

1. Membership: are their dues currently paid. Natural expiration date, covered by dues period. This would tend to be a much shorter period of time than for knowledge or skill expiration.
2. Knowledge competency (written):

1. General Knowledge: Dock etiquet, Tides and Currents, Wind conditions, Boundaries for skill levels. Note: dock etiquet (shared use of docks) can change if some docks are out of service. Youth classes are being held, etc… The others would tend not to change much over time.
2. Sport and Equipment Specific Knowlege (Beginner, Intermediate, and Advanced)
    1. Dinghy Sailing: Hoist usage, rigging of specific boat types.
    2. Windsurfing: equipment familiarity (terminology), board set up. Rigging
    3. Windsurf foiling: equipment familiarity, foil setup, rigging
    4. Wing foiling: equipment familiarity, foil setup, wing setup and care.
3. Sport Specific Skills
    1. Dinghy sailing: Junior Skipper (Docking, tacking, jibing, man-over-board, capsize and recovery)
    2. Windsurfing: Beginner (board paddling to return to dock), Junior: Sail in 10mph or less wind around buoy X in cove and return to dock; Intermediate: sail in 10-15mph wind around buoy Y and return; Advanced: Sail in 15-25mph wind around buoy Z and return.
    3. Windsurf foiling: must be *advanced* windsurfer. Intermediate sustained foiling for 30 seconds or more. Advanced: demonstrate successful foiling jibe. Distance cruising from Berkeley: (a) Treasure Island, (b) R2 Buoy (above TI), (c) R4 Buoy (off Angel Island), (d) Blossom Rock Buoy, (e) Alcatraz rounding, (f) Harding Rock Buoy, (f) Golden Gate Bridge (and back!).
    4. Wing foiling: etc…

Now there are also a number of things a club might need to know about you that do **not** need to be in a credential such as (a) phone number, (b) address, (c) emergency contact information, (d) student status, etc…

2.2 JSON for Everything!

2.2.1 Data Modeling and Exchange: JSON

JSON:

> JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language Standard ECMA-262 3rd Edition - December 1999. JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language.

JSON is the text based data "lingua franca" for the web. I would teach this early (week 4) in a first web programming class for CS majors. See JSON Slides.

Example of an JSON based credential from [1]:

```
{
  "@context": [
    "https://www.w3.org/ns/credentials/v2",
    "https://www.w3.org/ns/credentials/examples/v2"
  ],
  "id": "http://university.example/credentials/58473",
  "type": ["VerifiableCredential", "ExampleAlumniCredential"],
  "issuer": "did:example:2g55q912ec3476eba2l9812ecbfe",
  "validFrom": "2010-01-01T00:00:00Z",
  "credentialSubject": {
    "id": "did:example:ebfeb1f712ebc6f1c276e12ec21",
    "alumniOf": {
      "id": "did:example:c276e12ec21ebfeb1f712ebc6f1",
      "name": "Example University"
    }
  }
}
```

2.2.2 Data Validation: JSON-Schema

JSON is great for modeling almost any data structure imaginable. However most applications only wants certain types of data in a particular format, i.e., a small subset of possible JSON structures and content. Typically the structure and type of data is specified by a ***schema*** of some kind.

From JSON-Schema:

> While JSON is probably the most popular format for exchanging data, JSON Schema is the vocabulary that enables JSON data consistency, validity, and interoperability at scale. JSON Schema is a declarative language for annotating and validating JSON documents' structure, constraints, and data types

In addition a JSON-Schema is specified via JSON! I Would teach this in a second web development course for use in data validation. JSON-Schema Slides.

I use JSON-Scheme in multiple places in my open source VC interoperability test server. Below is an example that does some basic checks on a credential:

```
{
     "$id": "https://grotto-networking.com/simple-credential.schema.json",
     "title": "UnsignedCredential",
     "description": "A basic credential validator, data model v1.1 or v2.0",
     "type": "object",
     "properties": {
          "@context": {
               "type": "array",
               "items": {
                    "type": ["string", "object"]
               }
          },
          "id": {
               "type": "string",
               "format": "uri"
          },
          "type": {
               "type": ["string", "array"]
          },
          "credentialSubject": {
               "type": ["object", "array"]
          },
          "issuer": {
               "anyOf": [
                    {
                         "type": "string",
                         "format": "uri"
                    },
                    {
                         "type": "object",
                         "properties": {
                              "id": {
                                   "type": "string",
                                   "format": "uri"
                              }
                         },
                         "required": ["id"]
                    }
               ]
          },
          "credentialStatus": {
               "type": "object",
```

```
                "properties": {
                    "id": {
                        "type": "string",
                        "format": "uri"
                    },
                    "type": {
                        "type": ["string", "array"]
                    }
                },
                "required": ["type"]
            }
        },
        "required": ["@context", "type", "credentialSubject", "issuer"]
}
```

To actually perform checks of JSON document against a JSON-Schema, optimized code libraries are used. I use and would have my students use npm: AJV. In the figure below we show the NPM page for AJV, as you can see from the weekly download numbers (100M+) JSON-Schema for input validation is extremely popular.

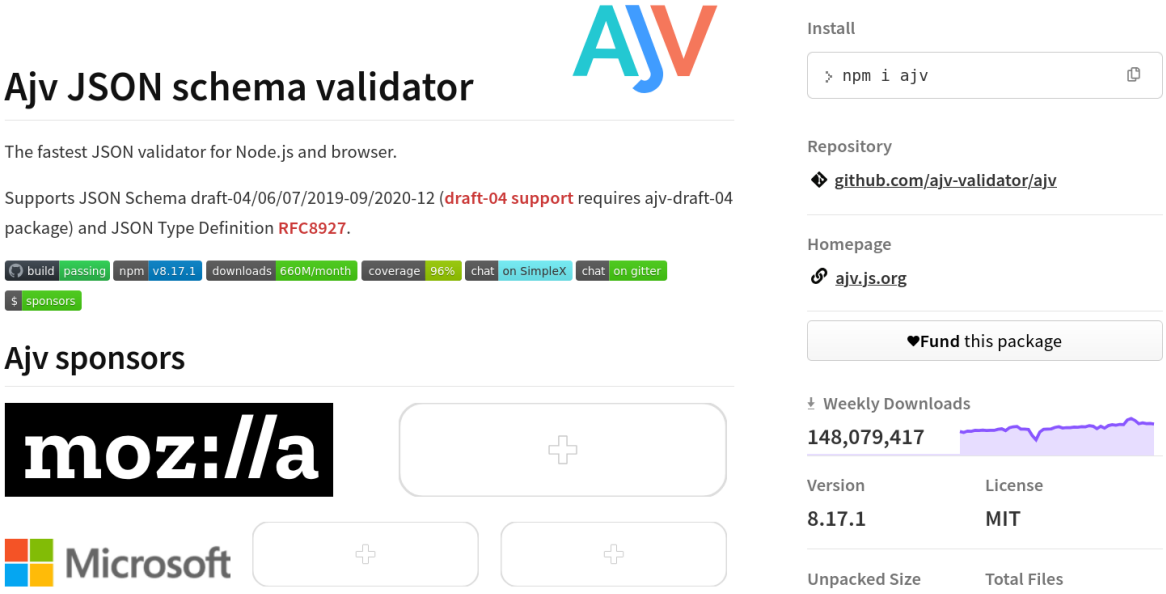

Figure 1: AJV NPM page

2.2.3 But What Does it mean: JSON-LD

From [3]: 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,
- the ability to annotate strings with their language,
- a way to associate datatypes 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.

Let's focus on the second bullet "a way to disambiguate keys". Consider the following (invalid) credential for a "mystery club":

```
{
  "@context": [
    "https://www.w3.org/ns/credentials/v2",
    "https://www.MysteryClub.org/ns/credentials"
  ],
  "id": "http://university.example/credentials/58473",
  "type": ["VerifiableCredential", "MysteryClubEquipmentCredential"],
  "issuer": "did:example:2g55q912ec3476eba2l9812ecbfe",
  "validFrom": "2010-01-01T00:00:00Z",
  "credentialSubject": {
    "id": "did:example:ebfeb1f712ebc6f1c276e12ec21",
    "equipment": {
      "id": "did:example:c276e12ec21ebfeb1f712ebc6f1",
      "poles": "Some kind of pole"
    }
  }
}
```

From the above we see that "mystery club" has issued a credential about *equipment* to a *credential subject*. We see that subject has a key denoted *poles*. But what does this mean? (a) hiking poles, (b) treking poles, (c) spinaker pole, (d) poles for pole-vaulting, (e) alpine ski poles, (f) cross country ski poles, (g) gondolier pole, etc… What we are missing is *context*! Figuratively and literally JSON-LD enables us to clearly define what our *terms* mean via a `@context` which is a JSON document that maps *keys* to long form, unique terms that can be documented. In the following we'll look at keys/terms that we get from the VC Data Model

2.0 *context* and the Data Integrity 1.0 *context*, then we'll define a custome context for our club.

2.3 Structure and Fields from the VC Data Model 2.0

The VC Data Model 2.0 [1] provides us with the following required and optional structure as shown below. Note that the *comments* `//` are not legal JSON syntax but are included here to explain the contents in a compact manner.

```
{
  "@context": [
    "https://www.w3.org/ns/credentials/v2",
    "https://www.mysailclub.org/ns/credentials" // Our context will go here
  ],
  "id": "http://university.example/credentials/58473", // optional, id of the VC
  "name": "The name of this credential", // optional
  "description": "A description of the credential", // optional
  "type": ["VerifiableCredential", "WaterClubCredential"], // required
  "issuer": { // required, Specify the issuer nicely
    "id": "https://mysailclub.org", // club IRI (but also URL)
    "name": "My Wind Sports Club", // issuer name
    "description": "A public or school based wind sports club." // optional issuer desc
  },
  "validFrom": "2010-01-01T00:00:00Z", // optional
  "validUntil": "2020-01-01T19:23:24Z", // optional
  "credentialSubject": { // required
    "id": "did:example:ebfeb1f712ebc6f1c276e12ec21", // optional, id of the subject
    // Here is where all the custom stuff will go
  }
}
```

2.4 Example Sport Specific Credential

I'm going to work somewhat backwards and will start with a fairly rich example for the wind sport discipline of "windsurf foiling" and then use that to guide the development of the corresponding `@context`.

```
{
  "@context": [
    "https://www.w3.org/ns/credentials/v2",
    "https://bawfc.grotto-networking.com/credentials" // Where I should be a copy of the con
```

```
  ],
  "name": "Bay Area Windsurf Foiling Credential",
  "description": "Denotes membership, knowledge, and skills in windsurf foiling",
  "type": ["VerifiableCredential", "WindFoilCredential"],
  "issuer": { // Specify the issuer nicely
    "id": "https://bawfc.grotto-networking.com", // club IRI (but also URL)
    "name": "Bay Area Windsurf Foiling Club", // issuer name
    "description": "Not a real club, but real information" // issuer desc
  },
  // "validFrom": "2010-01-01T00:00:00Z",  // Not using TMI
  // "validUntil": "2020-01-01T19:23:24Z", // Not using different parts expire differently
  "credentialSubject": {
    // "id": "did:example:ebfeb1f712ebc6f1c276e12ec21", // not using
    // Here is where all the custom stuff will go
    "membership": // Related to dues
        {"start": "2025-03-01T00:00:00Z"
        "end": "2025-10-31T00:00:00Z"},
    "knowledge": [ {"topic": "DockEtiquette", "date": "1984-01-01T00:00:00Z"}
        {"topic": "TidesAndCurrents", "date": "2010-01-01T00:00:00Z"},
        {"topic": "WindPatterns", "date": "2010-01-01T00:00:00Z"},
        {"topic": "LandMarksBuoys", "date": "2010-01-01T00:00:00Z"},
        {"topic": "BigBoatsFerries", "date": "2010-01-01T00:00:00Z"}
    ],
    "skills": [{"name": "Rigging", "date":"2010-01-01T00:00:00Z"},
        {"name": "FoilSetup", "date": "2017-01-01T00:00:00Z"}
    ],
    "distanceAchiements": [{"name": "TI", "date": "2010-01-01T00:00:00Z"},
        {"name": "R2", "date": "2011-01-01T00:00:00Z"}
    ]
  }
}
```

2.5 Creating the Club Specific `@context`

The above example credential may need a bit of clean up. The `name` key may not be able to be redefined. In general every term (key) used in our example and not part of the VC DM2 needs to be defined in our club specific context. These include: *membership*, *start*, *end*, *knowledge*, *topic*, *date*, *skills*, *name*, *distanceAchiements*. Note that I've reused *name* and *date* in multiple places. Is this bad?

I was getting errors on redefining *name* so revised the example credential to look like:

```
{
  "@context": [
    "https://www.w3.org/ns/credentials/v2",
    "https://bawfc.grotto-networking.com/credentials"
  ],
 "name": "Bay Area Windsurf Foiling Credential",
  "description": "Denotes membership, knowledge, and skills in windsurf foiling",
  "type": ["VerifiableCredential", "WindFoilCredential"],
  "issuer": {
    "id": "https://bawfc.grotto-networking.com",
    "name": "Bay Area Windsurf Foiling Club",
    "description": "Not a real club, but real information!"
  },
  "credentialSubject": {
    "type": ["Member", "WindsurfFoiler"],
    "membership":
        {"start": "2025-03-01T00:00:00Z", "end": "2025-10-31T00:00:00Z"},
    "knowledge": [ {"topic": "DockEtiquette", "date": "1984-01-01T00:00:00Z"},
        {"topic": "TidesAndCurrents", "date": "2010-01-01T00:00:00Z"},
        {"topic": "WindPatterns", "date": "2010-01-01T00:00:00Z"},
        {"topic": "LandMarksBuoys", "date": "2010-01-01T00:00:00Z"},
        {"topic": "BigBoatsFerries", "date": "2010-01-01T00:00:00Z"}
    ],
    "skills": [{"skill": "Rigging", "date":"2010-01-01T00:00:00Z"},
        {"skill": "FoilSetup", "date": "2017-01-01T00:00:00Z"}
    ],
    "distanceAchiements": [{"landmark": "TI", "date": "2010-01-01T00:00:00Z"},
        {"landmark": "R2", "date": "2011-01-01T00:00:00Z"}
    ]
  }
}
```


Iterating via the VC Playground with an embedded context first led finally to the following club specific context:


```
{
    "@context": {
        "Member": "https://bawfc.grotto-networking.com/cred/#Member",
        "WindsurfFoiler": "https://bawfc.grotto-networking.com/cred/#Member",
        "membership": "https://bawfc.grotto-networking.com/cred/#membership",
        "knowledge": {
            "@id": "https://bawfc.grotto-networking.com/cred/#knowledge",
```

```
            "@container": "@set"
        },
        "skills": {
            "@id": "https://bawfc.grotto-networking.com/cred/#skills",
            "@container": "@set"
        },
        "distanceAchiements": {
            "@id": "https://bawfc.grotto-networking.com/cred/#distanceAchiements",
            "@container": "@set"
        },
        "start": {
            "@id": "https://bawfc.grotto-networking.com/cred/#skills",
            "@type": "http://www.w3.org/2001/XMLSchema#dateTime"
        },
        "end": {
            "@id": "https://bawfc.grotto-networking.com/cred/#skills",
            "@type": "http://www.w3.org/2001/XMLSchema#dateTime"
        },
        "topic": "https://bawfc.grotto-networking.com/cred/#topic",
        "date": {
            "@id": "https://bawfc.grotto-networking.com/cred/#skills",
            "@type": "http://www.w3.org/2001/XMLSchema#dateTime"
        },
        "skill": "https://bawfc.grotto-networking.com/cred/#skill",
        "landmark": "https://bawfc.grotto-networking.com/cred/#landmark"
    }
}
```

TODO: clean up based on feedback and discuss.

3 Signatures, Keys, and Cryptosuites, oh my!

3.1 Choosing a Cryptosuite

Basic review of digital signatures, public key cryptography. Concerned with forgeries!

3.2 Dealing with Cryptographic Keys

3.2.1 Private keys

These need to be **protected**! There is a wide range of approaches...

3.2.2 Public keys (DIDs)

This needs to be made available in a "secure maner", don't want a malicious party to substitute in their public key and *impersonate* the real issuer.

4 Credential Issuance

4.1 Proof Options: Additional Structure and Fields from Data Integrity 1.0

Important this also sets up the "proof options" that are used by the DI cryptosuites.

5 Credential Verification

References

[1] M. Sporny, T. T. Jr, M. Jones, G. Cohen, and I. Herman, "Verifiable credentials data model v2.0," W3C, Candidate Recommendation, Feb. 2025. Available: https://www.w3.org/TR/vc-data-model-2.0/

[2] G. Bernstein, M. Sporny, D. Longley, I. Herman, and T. T. Jr, "Verifiable credential data integrity 1.0," W3C, Candidate Recommendation, Feb. 2025.

[3] G. Kellogg, D. Longley, and P.-A. Champin, "JSON-LD 1.1," W3C, {W3C} Recommendation, Jul. 2020.

[4] T. T. Jr, M. Sporny, and G. Bernstein, "Data integrity EdDSA cryptosuites v1.0," W3C, Candidate Recommendation, Feb. 2025.

[5] G. Bernstein, D. Longley, and M. Sporny, "Data integrity ECDSA cryptosuites v1.0," W3C, Candidate Recommendation, Feb. 2025.