Discovery of Self Sovereign Identity (SSI) from a security perspective

June 29, 2022

Share

This post is based on my understanding and feedback after studying the **Self Sovereign Identity** concepts (**SSI**) via all the documents and videos provided by Damien Bod in his blog post about SSI.

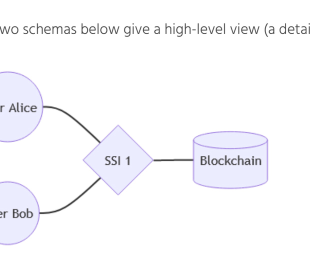
In addition, the following version of W3C specifications was used:

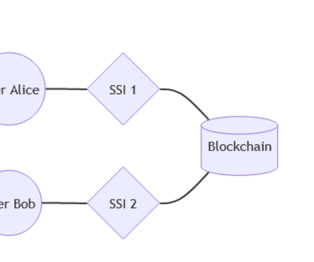
Specification name	Version	URL
Decentralized Identifiers (DIDs)	1.0	https://www.w3.org/TR/2021/PR-did-core-20210803/
Verifiable Credentials Data Model	1.1	https://www.w3.org/TR/2022/REC-vc-data-model-20220303/
Linked Data Cryptographic Suite Registry	Draft Community Group Report 29 December 2020	https://w3c-ccg.github.io/ld-cryptosuite-registry/
RSA Signature Suite 2018	Draft Community Group Report 26 May 2020	https://w3c-ccg.github.io/lds-rsa2018/

What is SSI?

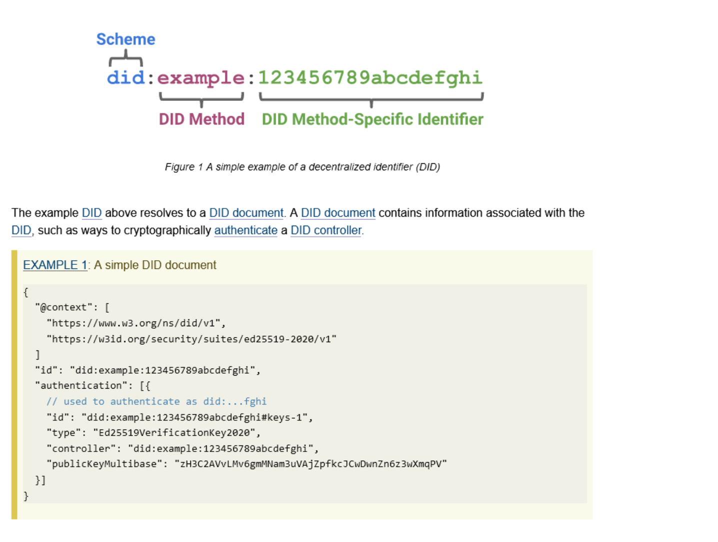
To quote Damien: "*Self-sovereign identity is an emerging solution built on blockchain technology for solving digital identities which gives the management of identities to the users and not organizations.*"

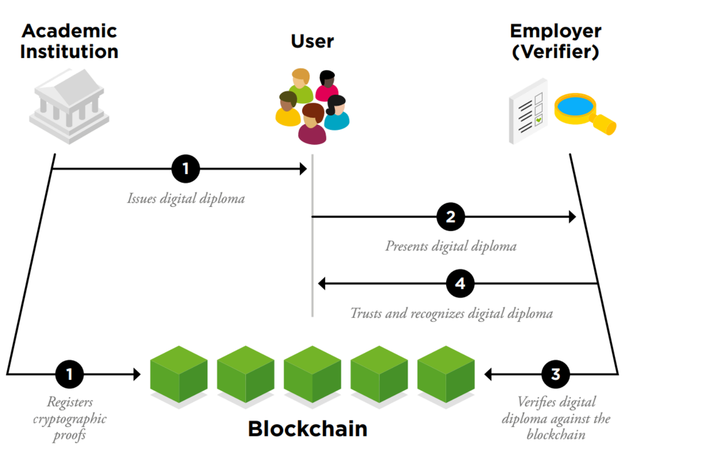
The two schemas below give a high-level view (a detailed view is provided in the next section):

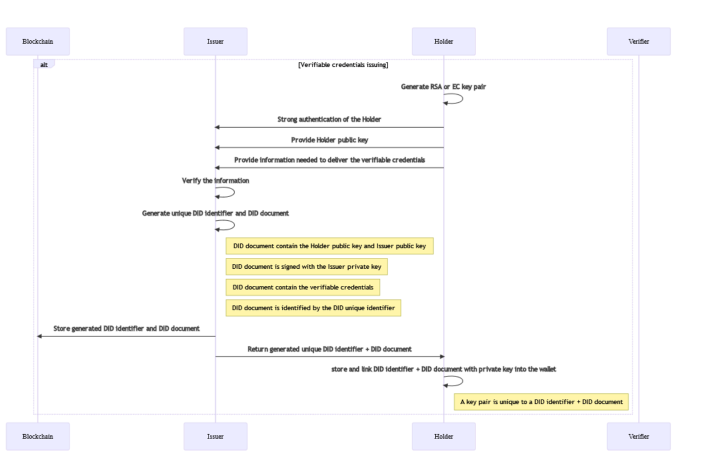


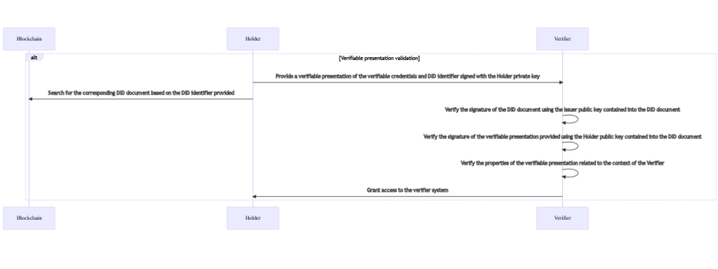


Some terminologies









- Holder fully controls the level of information provided to a Verifier within the verifiable credentials provided to it.
- Digital signature, using asymmetric RSA/EC key pair, is a critical pillar of SSI and trust/integrity foundation.
- Availability of the blockchain used by the Issuer/Verifier is a critical pillar of SSI.

Remarks about SSI providers

- For the moment; it is not possible to use any wallet with any SSI solution. Each solution is locked to its own wallet (October 2021 – source).
- An SSI provider (or solution) provides the Issuer and Verifier posture of a verifiable credential and it seems not to be possible to currently mix 2 different SSI providers (one for Issuer and one for Verifier) even if it is theoretically possible (source):

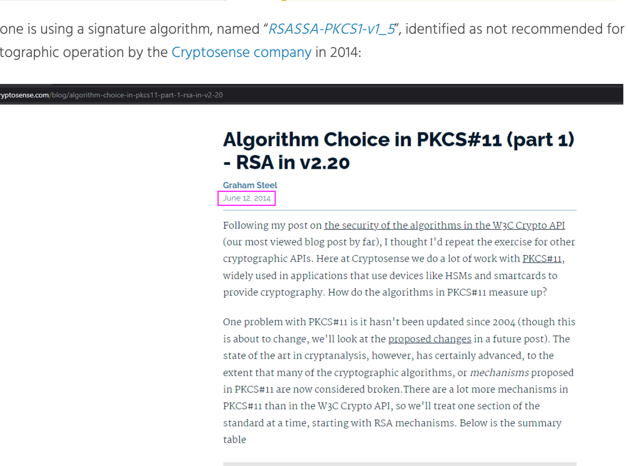
> righettod · March 21, 2022 - 16:59 · Reply→
>
> Thank you very much for such great infos and blog posts!!!!
>
> I have a small question to ensure my correct understanding:
>
> Currently it is not possible to use a SSI provider A or language verification API to verify the Verifiable Credentials issued by a SSI provider B?
>
> Thank you very much in advance for your feedback
>
> > damienbod · March 21, 2022 - 19:57 · Reply→
> >
> > Hi rightod, If the SSI A can understand the VC produced by the SSI B and both can use the same ledger where the DID is persisted and communicate using an agent using a common protocol then these can work together when the verifier knows and trusts the issuers DID
> >
> > Greetings Damien
>
> righettod · March 22, 2022 - 08:32 · Reply→
>
> So based on your feedback and the information from your blog posts, it is theoretically possible but currently SSI providers your evaluated do not support. It is correct?
>
> > damienbod · March 22, 2022 - 11:05 · Reply→
> >
> > Yes correct, if the providers use the same standards, then most can work together.

- An SSI provider (or solution) provides a REST API providing an abstraction over the Issuing and Verification operation of verifiable credentials, the wallet management as well as the blockchain to store DID identifier + DID document.

Cryptographic algorithms and SSI-related W3C specifications

Algorithms supported by the "*Verifiable Credentials Data Model*" are defined in the specification named "*Linked Data Cryptographic Suite Registry*". It is important to note that this one was in the "Unofficial Draft" status when this post was written (March 2022). Therefore, there is not currently W3C official standard regarding supported and recommended signature algorithms.

Among the list of mentioned signature algorithms in the document, there is one, named "*RsaSignature2018*" :


w3c-ccg.github.io/ld-cryptosuite-registry/#rsa

W3C Community Group Draft Report

TABLE OF CONTENTS

1. Introduction
2. The Registration Process
3. Signature Suites
   - 3.1 Ed25519
     - 3.1.1 Ed25519Signature2018
   - 3.2 RSA
     - 3.2.1 RsaSignature2018
   - 3.3 Secp256k1
     - 3.3.1 EcdsaSecp256k1Signature2019
     - 3.3.2 EcdsaSecp256k1RecoverySignature2020
   - 3.4 JWS
     - 3.4.1 JsonWebSignature2020
   - 3.5 GPG
     - 3.5.1 GpgSignature2020
   - 3.6 JCS Ed25519 Signature 2020
     - 3.6.1 JcsEd25519Signature2020
   - 3.7 BBS+ Signature 2020
     - 3.7.1 BbsBlsSignature2020

§ 3.2 RSA

§ 3.2.1 RsaSignature2018

Summary	
Identifiers	RsaSignature2018
Status	PROVISIONAL
Authors	Dave Longley, Manu Sporny
Specification	RSA Signature Suite 2018

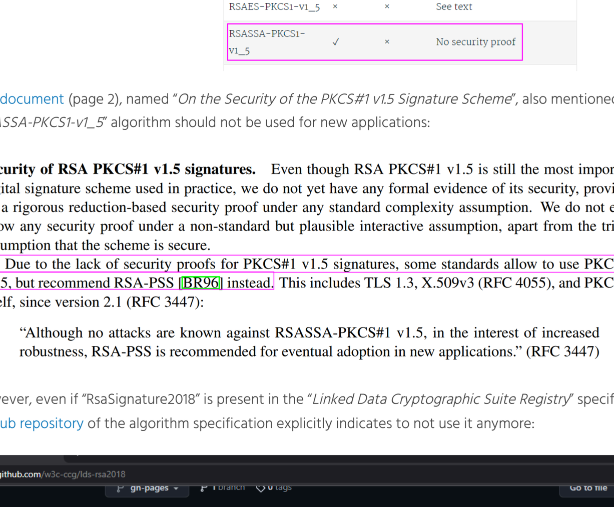
§ RsaVerificationKey2018

Summary	
Identifiers	RsaVerificationKey2018
Status	PROVISIONAL
Authors	Dave Longley, Manu Sporny
Specification	RSA Signature Suite 2018

EXAMPLE 2: Example of RsaVerificationKey2018

```
{
  "id": "did:example:123456789abcdefghi#keys-1",
  "type": "RsaVerificationKey2018",
  "controller": "did:example:123456789abcdefghi",
  "expires": "2017-02-08T16:02:20Z",
  "publicKeyPem": "-----BEGIN PUBLIC KEY...END PUBLIC KEY-----\r\n"
}
```

This one is using a signature algorithm, named "*RSASSA-PKCS1-v1_5*", identified as not recommended for future cryptographic operation by the Cryptosense company in 2014:


cryptosense.com/blog/algorithm-choice-in-pkcs11-part-1-rsa-in-v2-20

Algorithm Choice in PKCS#11 (part 1) - RSA in v2.20

Graham Steel
June 12, 2014

Following my post on the security of the algorithms in the W3C Crypto API (our most viewed blog post by far), I thought I'd repeat the exercise for other cryptographic APIs. Here at Cryptosense we do a lot of work with PKCS#11, widely used in applications that use devices like HSMs and smartcards to provide cryptography. How do the algorithms in PKCS#11 measure up?

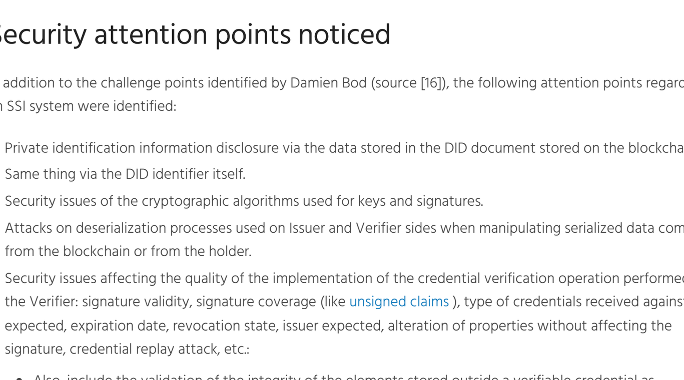
One problem with PKCS#11 is it hasn't been updated since 2004 (though this is about to change, we'll look at the proposed changes in a future post). The state of the art in cryptanalysis, however, has certainly advanced, to the extent that many of the cryptographic algorithms, or *mechanisms* proposed in PKCS#11 are now considered broken.There are a lot more mechanisms in PKCS#11 than in the W3C Crypto API, so we'll treat one section of the standard at a time, starting with RSA mechanisms. Below is the summary table

Algorithm/mode	Ok legacy	Ok future	Note
RSAES-PKCS1-v1_5	×	×	See text
RSASSA-PKCS1-v1_5	✓	×	No security proof

This document (page 2), named "*On the Security of the PKCS#1 v1.5 Signature Scheme*", also mentioned that "*RSASSA-PKCS1-v1_5*" algorithm should not be used for new applications:

> **Security of RSA PKCS#1 v1.5 signatures.** Even though RSA PKCS#1 v1.5 is still the most important digital signature scheme used in practice, we do not yet have any formal evidence of its security, provided by a rigorous reduction-based security proof under any standard complexity assumption. We do not even know any security proof under a non-standard but plausible interactive assumption, apart from the trivial assumption that the scheme is secure.
>
> Due to the lack of security proofs for PKCS#1 v1.5 signatures, some standards allow to use PKCS#1 v1.5, but recommend RSA-PSS [BR96] instead. This includes TLS 1.3, X.509v3 (RFC 4055), and PKCS#1 itself, since version 2.1 (RFC 3447):
>
> "Although no attacks are known against RSASSA-PKCS#1 v1.5, in the interest of increased robustness, RSA-PSS is recommended for eventual adoption in new applications." (RFC 3447)

However, even if "RsaSignature2018" is present in the "*Linked Data Cryptographic Suite Registry*" specification, the GitHub repository of the algorithm specification explicitly indicates to not use it anymore:


github.com/w3c-ccg/lds-rsa2018

gh-pages · 1 branch · 0 tags · Go to file · Code

OR13 Create README.md ✓ c1b711c on May 26, 2020 · 8 commits

LICENSE	Initial commit	5 years ago
README.md	Create README.md	2 years ago
index.html	Correct section number in reference	3 years ago

README.md

RsaSignatute2018

Although support for this suite exists to varying degrees in software implementations, the specification is out of date, and the suite is not recommended for further use.

If you need to support RSA or NIST curves, please consider supporting:

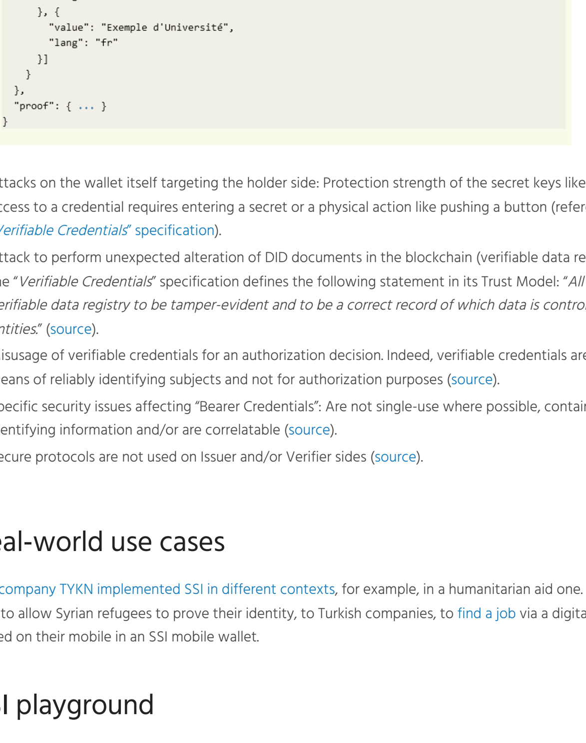
- https://github.com/w3c-ccg/lds-jws2020

It shows that the "*Linked Data Cryptographic Suite Registry*" specification needs to be finalized to remove any deprecated signature algorithms to prevent any issue by the implementer of SSI systems.

Security attention points noticed

In addition to the challenge points identified by Damien Bod (source [16]), the following attention points regarding an SSI system were identified:

- Private identification information disclosure via the data stored in the DID document stored on the blockchain.
- Same thing via the DID identifier itself.
- Security issues of the cryptographic algorithms used for keys and signatures.
- Attacks on deserialization processes used on Issuer and Verifier sides when manipulating serialized data coming from the blockchain or from the holder.
- Security issues affecting the quality of the implementation of the credential verification operation performed by the Verifier: signature validity, signature coverage (like unsigned claims), type of credentials received against expected, expiration date, revocation state, issuer expected, alteration of properties without affecting the signature, credential replay attack, etc.:
  - Also, include the validation of the integrity of the elements stored outside a verifiable credential as mentioned in the "*Verifiable Credentials*" specification (source):


§ 8.2 Content Integrity Protection

This section is non-normative.

Verifiable credentials often contain URLs to data that resides outside of the verifiable credential itself. Linked content that exists outside a verifiable credential, such as images, JSON-LD Contexts, and other machine-readable data, are often not protected against tampering because the data resides outside of the protection of the proof on the verifiable credential. For example, the following highlighted links are not content-integrity protected but probably should be:

EXAMPLE 34: Non-content-integrity protected links

```
{
  "@context": [
    "https://www.w3.org/2018/credentials/v1",
    "https://www.w3.org/2018/credentials/examples/v1"
  ],
  "id": "http://example.edu/credentials/58473",
  "type": ["VerifiableCredential", "AlumniCredential"],
  "credentialSubject": {
    "id": "did:example:ebfeb1f712ebc6f1c276e12ec21",
    "image": "https://example.edu/images/58473",
    "alumniOf": {
      "id": "did:example:c276e12ec21ebfeb1f712ebc6f1",
      "name": [{
        "value": "Example University",
        "lang": "en"
      }, {
        "value": "Exemple d'Université",
        "lang": "fr"
      }]
    }
  },
  "proof": { ... }
}
```

- Attacks on the wallet itself targeting the holder side: Protection strength of the secret keys like ensuring that access to a credential requires entering a secret or a physical action like pushing a button (reference in the "*Verifiable Credentials*" specification).
- Attack to perform unexpected alteration of DID documents in the blockchain (verifiable data registry) because the "*Verifiable Credentials*" specification defines the following statement in its Trust Model: "*All entities trust the verifiable data registry to be tamper-evident and to be a correct record of which data is controlled by which entities.*" (source).
- Misusage of verifiable credentials for an authorization decision. Indeed, verifiable credentials are intended as a means of reliably identifying subjects and not for authorization purposes (source).
- Specific security issues affecting "Bearer Credentials": Are not single-use where possible, contain personally identifying information and/or are correlatable (source).
- Secure protocols are not used on Issuer and/or Verifier sides (source).

Real-world use cases

The company TYKN implemented SSI in different contexts, for example, in a humanitarian aid one. The objective was to allow Syrian refugees to prove their identity, to Turkish companies, to find a job via a digital work permit stored on their mobile in an SSI mobile wallet.

SSI playground

In addition to the free tiers offered by some SSI providers (source section "Companies"), the "Hyperledger Indy" open-source project offers a complete SSI playground in the form of a collection of docker images.

Using this playground, it is possible to deep dive into the different elements involved in SSI flows such as the reception by a Holder of a verifiable credential from an Issuer.

A demonstration video was performed to show an example of the usage of this playground.

Conclusion

Self-Sovereign Identity (SSI) is a very interesting approach to significantly enhance the resiliency of any delivered credentials, as well as a boost for the dematerialization of official identification documents like passports, driver's licenses, national identity cards and so on.

However, it misses nowadays a point from an SSI provider perspective because of the need to strictly follow the W3C specification to get interoperability between providers. In addition, the W3C specification for "*Linked Data Cryptographic Suite Registry*" must be finalized and released. It is needed to provide a clear guidance about cryptographic algorithms for secure handling of the keys and signature aspects of SSI flows.

Did you like this article? Find some more articles in our blog section.

Dominique Righetto

General

Recent Posts
The Necessity of Cyber Crisis Exercises
The Art of Password Spraying: A Comprehensive Analysis
Common client-side vulnerabilities of web applications

Categories
Advice
Consulting
Cyber Blog Post
General
Newsletter
PR
The Cyber Blog Times
Uncategorized

Search
Search ...