VCWG Data Integrity

August 2022 - A proposal for streamlining crypto suites

Agenda

01

The Problems

Agility and proliferation

02

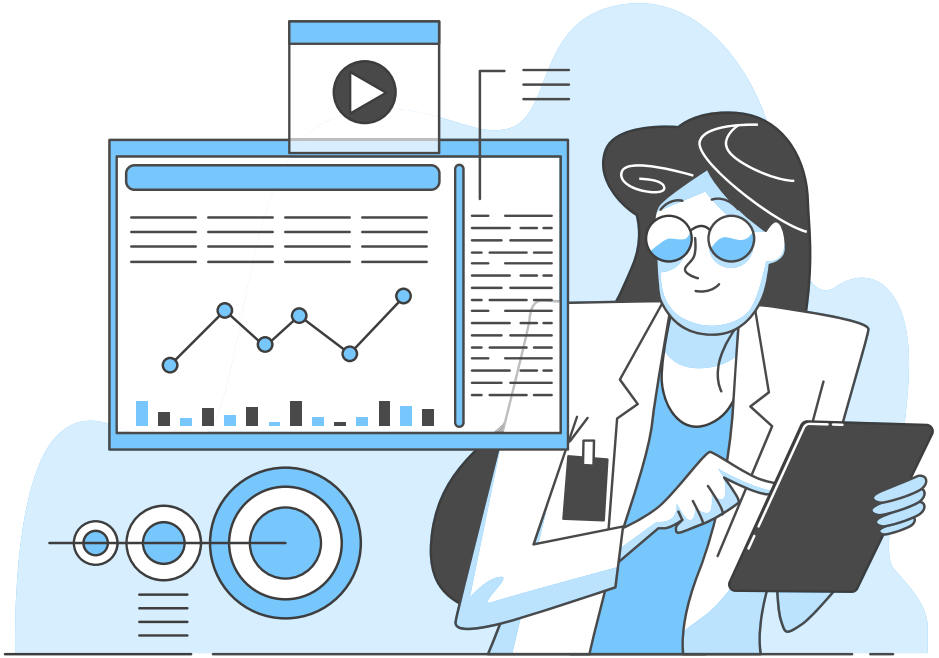
A Solution

Simplification plan

03

Roadmap

Execution timeline



01

The Problems

The Problem (2018): Default Crypto Suites

"Let's pick a handful of default crypto suites for every version of the specification (e.g., 2K RSA, P-256 ECDSA, EdDSA)."

```
{
  "@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"],
  ...
  "proof": {
    "type": "Ed25519Signature2018",
    "created": "2018-02-25T14:58:42Z",
    "verificationMethod": "https://example.edu/issuers/a#key-1",
    "proofPurpose": "assertionMethod",
    "jws": "z3FXQjecWufY46…UAUL5n2Brbx"
  }
```

New concern: "You're excluding certain communities, like those that use PGP and Koblitz curves! You need more cryptographic agility!"

The Problem (2020): Crypto Suite Proliferation

"Ok, let's be less coupled to the VC data model context and more agile.
Let's move crypto suite definitions into their own JSON-LD Contexts!"

```
{
  "@context": [
    "https://www.w3.org/2018/credentials/v1",
    "https://www.w3.org/2018/credentials/examples/v1",
    "https://w3id.org/security/suites/ed25519-2020/v1"
  ],
  "id": "http://example.edu/credentials/58473",
  "type": ["VerifiableCredential", "AlumniCredential"],
  ...
  "proof": {
    "type": "Ed25519Signature2020",
    "created": "2022-02-25T14:58:42Z",
    "verificationMethod": "https://example.edu/issuers/a#key-1",
    "proofPurpose": "assertionMethod",
    "proofValue": "z3FXQjecWufY46…UAUL5n2Brbx"
  }
```

New concern: "There are going to be soooo many crypto suites,
and they all have more or less the same properties!"

The Problem (2020): Crypto Suite Proliferation

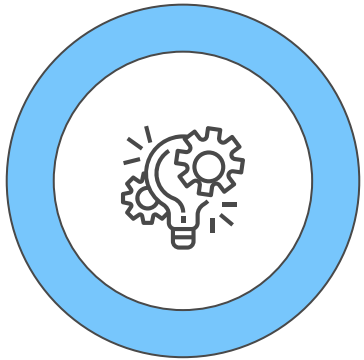
How many crypto suites could there be? Well, there are at least this many today:

- `https://w3id.org/security/suites/ed25519-2020/v1`
- `https://w3id.org/security/suites/x25519-2019/v1`
- `https://w3id.org/security/suites/merkle-disclosure-2021/v1`
- `https://w3id.org/security/suites/secp256k1recovery-2020/v1`
- `https://w3id.org/security/suites/pgp-2021/v1`
- `https://w3id.org/security/suites/blockchain-2021/v1`
- `https://w3id.org/security/suites/jws-2020/v1`
- `https://w3id.org/security/suites/bls12381-2020/v1`
- `https://w3id.org/security/suites/eip712sig-2021/v1`
- `https://w3id.org/security/suites/secp256k1-2020/v1`
- `https://w3id.org/security/suites/secp256k1-2019/v1`
- `https://w3id.org/security/suites/merkle-2019/v1`
- `https://w3id.org/security/suites/chained-2021/v1`

It's not terrible, and some of those are necessary, but most of them only differ by the crypto suite `type` that they define, such as `Ed25519Signature2020` or `JsonWebSignature2020`.

02

The Solution



The Solution

What if we define a base Data Integrity Signature type in the Verifiable Credentials v2 context that works for 80% of the crypto suites that we already have?

Since we only seem to be changing the crypto suite type in most crypto suites, if we shift that value to be a string, we can greatly reduce crypto suite proliferation.

This solution is backwards-compatible and does not preclude other more advanced crypto suites.

The Solution: A Backwards-Compatible Example

```
{
  "@context": [
    "https://www.w3.org/2022/credentials/v2",
    "https://www.w3.org/2022/credentials/examples/v2"
  ],
  "id": "http://example.edu/credentials/58473",
  "type": ["VerifiableCredential", "AlumniCredential"],
  ...
"proof": {
    "type": "DataIntegritySignature",
    "cryptosuite": "eddsa-2022", <-- this is now a string value
    "created": "2022-02-25T14:58:42Z",
    "verificationMethod": "https://example.edu/issuers/a#key-1",
    "proofPurpose": "assertionMethod",
    "proofValue": "z3FXQjecWufY46…UAUL5n2Brbx"
  }
```

Other potential crypto suites: nist-ecdsa-2022, koblitz-ecdsa-2022, rsa-2022, pgp-2022, bbs-2022, eascdsa-2022, ibsa-2022, jws-2022, recommended-2022, selective-disclosure-2022, postquantum-2022, etc.

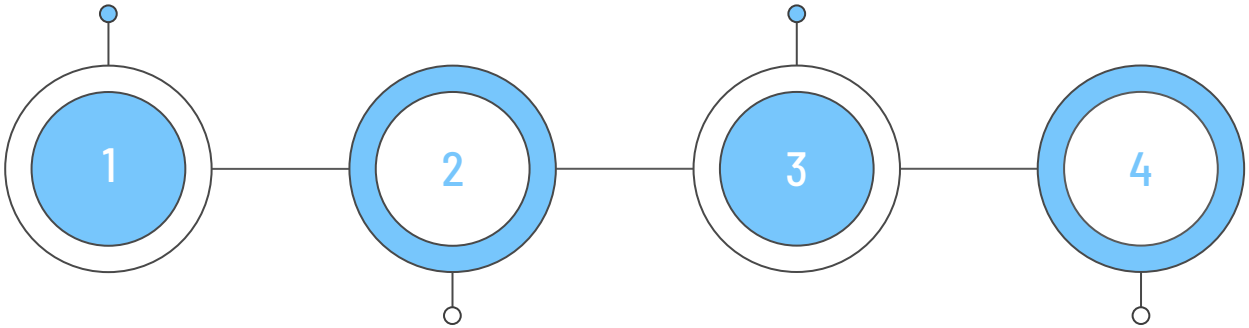
Downsides?

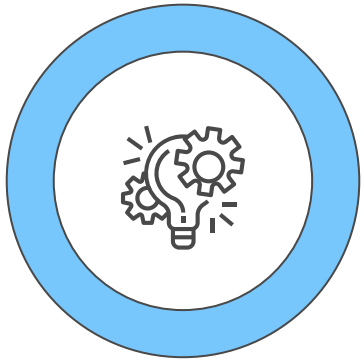
Semantic compression with CBOR-LD can't easily compress short, unique strings, so we become ~10-15 bytes less efficient per encoded signature.

...any other downsides?

03
The Roadmap

The Roadmap


Define v2 DataIntegritySignature (1 month)
1
Demonstrate Implementation and Test Suite (3 months)
2
Call for Multiple Implementations (6 months)
3
Candidate Rec by June 2023
4



Future Data Integrity Work (for later discussion)

- The Multikey format
- Cryptographic Hardening vs. Cryptographic Agility
- Recommended, agile crypto suites

Discussion?

Credits

Do you have any questions?

msporny@digitalbazaar.com

https://www.w3.org/2017/vc/WG/