Decentralized Identifier WG FF2F Sessions

Day 1: June 30, 2020
Chairs: Brent Zundel, Dan Burnett
Location: Virtual

Welcome!

- Logistics
- W3C WG IPR Policy
- Agenda
- IRC and Scribes
- Status
- Timeline Reminder

Logistics

- **Zoom call:**
  - See https://lists.w3.org/Archives/Member/member-did-wg/2020Jun/0000.html for dial in information (member only link)
- **Meeting times**:
  - Tuesday June 30: 10:00 - 13:30 EDT (16:00 - 19:30 CET, 07:00 - 10:30 PDT, 23:00 - 02:30 JST)
  - Wednesday July 1: 12:00 - 15:30 EDT (18:00 - 21:30 CET, 09:00 - 12:30 PDT, 01:00 - 04:30 JST)
- **DID WG Agenda**: https://tinyurl.com/ycyhrv8w
- **Live slides**: https://tinyurl.com/y7dopdom (Google Slides)
- **Breakout Rooms**:
  - **Hallway**: https://zoom.us/j/91515310373?pwd=WVF0T1Jrc0NwazI3TUZhYU85dUEyZz09
  - **Other**: https://mit.zoom.us/j/91971361936?pwd=MIo3UWlqUDAvYVg2VUtQZlcxMUJCUT09

W3C WG IPR Policy

- This group abides by the W3C patent policy https://www.w3.org/Consortium/Patent-Policy-20040205
- Only people and companies listed at https://www.w3.org/2004/01/pp-impl/117488/status are allowed to make substantive contributions to the specs
- Code of Conduct https://www.w3.org/Consortium/cepc/

Today’s agenda

10:00		
10:00	Welcome, Introductions, Status, and Logistics	Chairs
10:30	Use Cases	Phil
11:30	Break	
12:00	Rubric	Chairs
12:30	Implementation Guide	Markus/Drummond
13:00	Features at Risk: CBOR & JSON Representations	Manu

IRC and Scribes

- Meeting discussions will be documented
  - Text Chat: http://irc.w3.org/?channels=did
  - IRC://irc.w3.org:6665/#did
- Telecon info
  - https://lists.w3.org/Archives/Member/member-did-wg/2020Jun/0000.html

	Tuesday	Wednesday
1	Manu	PhilA
2	Amy	Drummond

<JoeAndrieu> q+ to comment on biometrics
<brent> ack JoeAndrieu
<Zakim> JoeAndrieu, you wanted to comment on biometrics

DID WG Mission and Goals

- “... standardize the DID URI scheme, the data model and syntax of DID Documents, which contain information related to DIDs that enable the aforementioned initial use cases, and the requirements for DID Method specifications.”

Charter Deliverables and Status

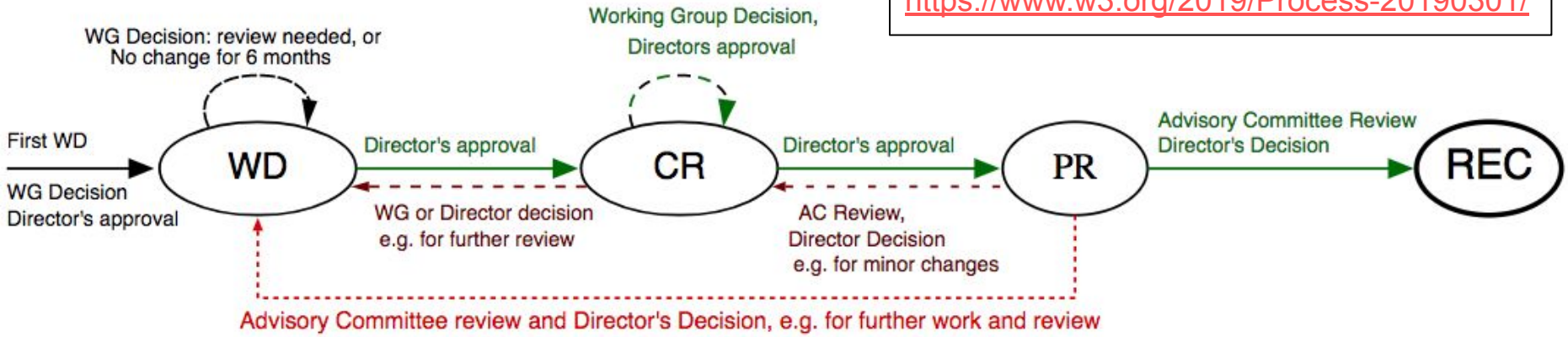
- Recommendation-track Specification
  - Decentralized Identifiers v1.0
    - Almost all major issues resolved, lots of little stuff
- W3C Notes
  - Decentralized Identifier Use Cases v1.0
    - Close to done. Maybe today?
  - Decentralized Characteristics Rubric v1.0
    - Nothing yet. We will discuss today.
- Other Deliverables
  - Test Suite and Implementation Report
    - There are some tests, but we have not yet SERIOUSLY begun.

W3C Technical Report Process

- Working Draft (WD) - does not imply consensus
- Candidate Recommendation (CR)
  - Entry - to publish as CR, the document is expected to be feature complete, have had wide review, and must specify the implementation requirements needed to exit
  - Exit - to exit CR (and move to PR), the document must satisfy the stated implementation requirements; it must also not have made any substantive change not warned about upon entry
- Proposed Recommendation (PR)
  - Basically a one-month sanity check during which the AC is encouraged to have any final review and discussion, but if anything major happens it’s a fail (requiring a move back to CR or earlier)
- Recommendation - Done
  - But errata are possible

Timing

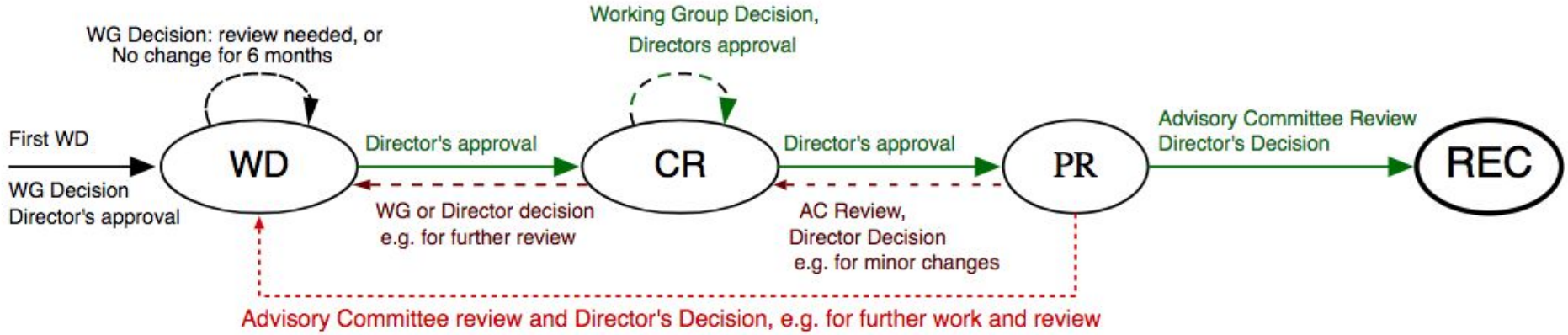
https://www.w3.org/2019/Process-20190301/

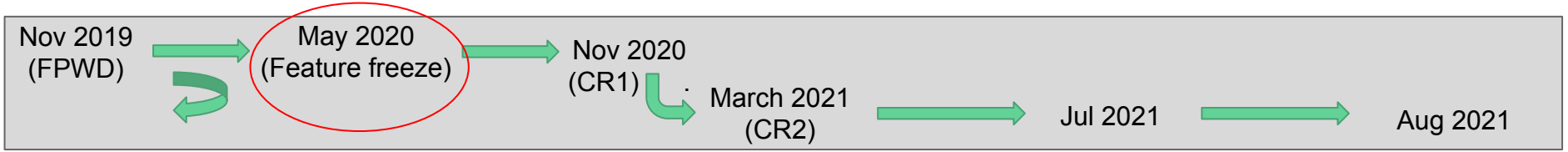


2.3 Timeline

Specification	FPWD	CR	PR	Rec
Decentralized Identifier Use Cases & Requirements (NOTE)	November 2019			August 2021
Decentralized Characteristics Rubric (NOTE)	December 2019			September 2021
Decentralized Identifiers Data Model and Syntax(es)	November 2019	November 2020	July 2021	August 2021
Note: The group will document significant changes from this initial schedule on the group home page.				

Timing of our primary spec





Decentralized Identifiers Data Model and Syntax(es)	November 2019	November 2020	July 2021	August 2021

https://www.w3.org/2019/Process-20190301/

Goals for this meeting

- Get to feature freeze
  - Path to CR

Use Cases (Phil & Joe, 60 min)

SitRep

- UCR doc adapted from CCG original (Kim, Ryan and Adrian listed as authors)
- Has section saying that we're not starting from scratch - so we can talk about DIDs, DID Docs etc.
- BUT... still keeping away from 'solutioneering'
- Includes all the original 'focal' use cases (extended use cases)
- Now includes 8 short use cases (1-3 paragraphs)
- Has been tricky to ensure use cases are for DIDs and not VCs
- Most issues now resolved - but we have a few to highlight and, ideally, resolve today.

Open Issues

https://github.com/w3c/did-use-cases/issues

Issue 2 Need to add coverage of portability/substitutability
Left over from CCG
Propose: close (overtaken by events)

Issue 14 What does it mean for a DID to be "recorded in a registry"?
Mike raised this, Joe and Manu have discussed, Dave supports
Propose: resolved as per Joe's comment

Open Issues

UCR issue 39 is a duplicate of core issue 248. Need term for relying party

From what I can see, the options are:

- Relying party (current phrase, @OR13 supports, several against)
- (DID) client (used in DID Resolution, @peacekeeper supports)
- Requesting Party (@agropper , @dmitrizagidulin , @msporny, @brentzundel, @OR13 latterly)
- Verifier (@talltree, @msporny latterly)
- DID user (@agropper latterly, @msporny strongly opposes)

Some people have changed their minds over time (which is allowed!)

But from the above, if there's anything like a consensus here, it's around Requesting Party. Are there any strong -1s for 'Requesting Party' ?

(thumbs up from Manu, Orie and Kyle. PR ready to go)

Propose: agree on ‘Requesting Party’ and close both issues

Open Issues

Issue 75 Auto-inclusion of glossary from DID-Core

We auto-include the glossary, to avoid different definitions for the same thing
Problems:

1. It's not (quite) self contained. It includes, for example, references to #did-resolution although, thanks Amy, it's a lot more self contained than it was.
2. It includes more solutioneering than is comfortable.

Discussion: do we fix those errors - which we hope will make the filter mechanism work - or take a snapshot and allow a delta with "the core spec is normative" warning?

Joe’s to do list

Issue 13 Describe relationship between DID Methods and DID Authentication in Use Cases document (raised by Mike)

Issue 52 Consider use cases from key representation discussion in did-core

Issue 81 Add DHS use case(s)

Phil’s to do list

Issue 58 Suggestion: Better partitioning of "DID" actions
Text already updated. Need to create better diagram, likely to be a set of SVG diagrams.

Issue 62 Jargon from VCDM and crypto
Need to read through and generally make it more readable. Lots done in this regard but more to do.

WG members’ to do list

If your use case isn’t there.
If your requirement is not covered by a use case

Please supply a PR

We have a PR outstanding but the contributor, YueJing95, has not responded to request for clarification.

Propose: close

Final steps

Re-do collation of requirements (including issue 87 domain map).

We won’t rename the master branch

Break (30 min)

Rubric (Chairs.Dan, 30 min)

Starting

This deliverable is optional.

Do we want to do it?

If so, we need a starting point.

Discuss.

Finishing

(Assuming we just started)

Next steps?

Implementation Guide (Markus/Drummond, 30 min)

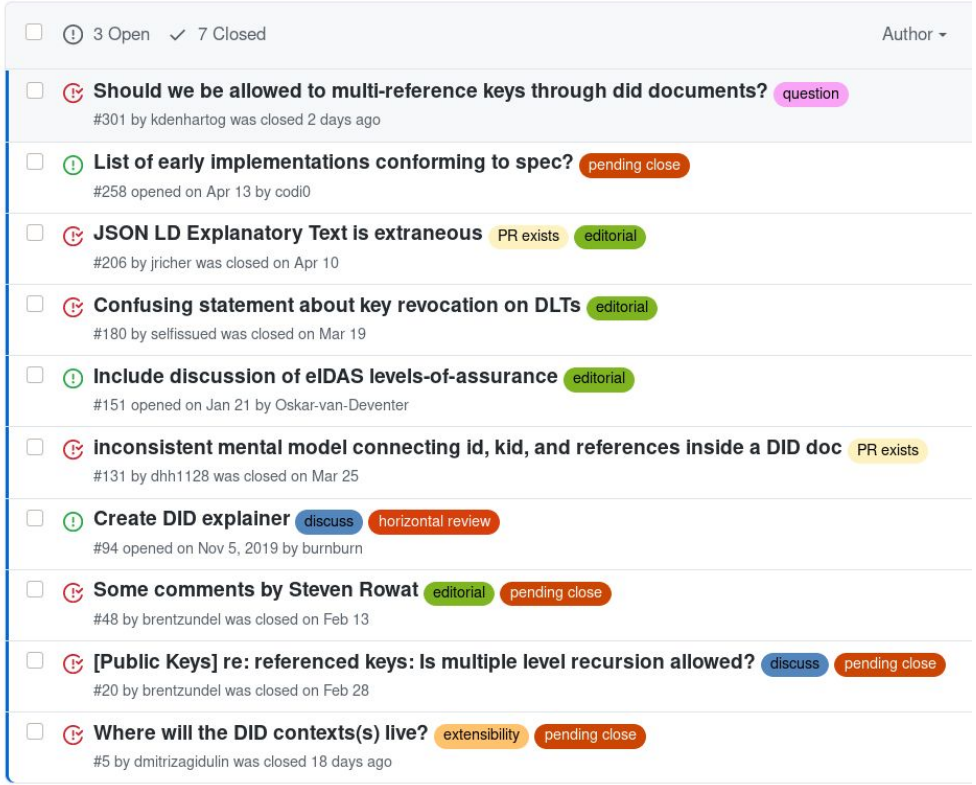
Purpose

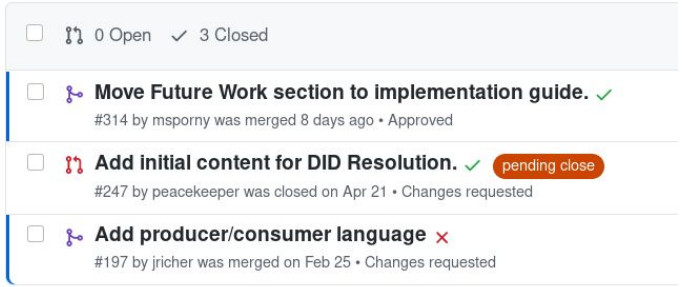
- The Implementation Guide provides guidance to implementers that are developing software for the Decentralized Identifier ecosystem
- https://w3c.github.io/did-imp-guide/
- This is where we put any non-normative content that we feel does not need to be in DID Core or the DID Specification Registries, e.g., Future Work
- It's also where we should recommend best practices for:
  - DID method specification authors
  - Authors of DIDs and DID documents
  - DID resolver and dereferencer implementers
  - Verifiable data registries
  - Application developers who want to use DIDs

Current Content

- JSON-LD and DID Documents:
  - Details on how to use JSON-LD
- Future Work:
  - Upper Limits on DID Character Length, Equivalence, Verifiable Timestamps, Time Locks and DID Document Recovery, Smart Signatures, Verifiable Credentials, Alternate Serializations and Graph Models

Issues and PRs mentioning "Implementation Guide"





Example

Should we include a discussion of eIDAS levels-of-assurance in did-core and/or did-use-cases?

DID Core Spec Section 9.2.3. Authentication and Verifiable Claims ➜ Add a short note that it's possible to establish a binding between DIDs and a legal identity, but don't go into details.
DID Core Spec Section 9. Security Considerations ➜ Add a subsection with technical considerations about the security of keys.
Use Cases and Requirements ➜ What can you do once you support the use of DIDs with legally enabled identity systems such as eIDAS? Why is this important? What are the risks?
DID Implementation Guide ➜ How is the DID-eIDAS bridge implemented, how are DIDs and VCs used to achieve different LoAs?

Other ideas for content?

- Discuss relationship of DID-based public key infrastructure to conventional PKI, e.g., x.509 Certs
- How to establish proof of control authority (e.g. KERI)
- Use of "id" properties in DID documents (immutable or not)
- FAQ-style content
- DIDs and National ID programs (e.g., eIDAS)
- DIDs and decentralized social networks (e.g., ActivityPub, federated social web)

Next Steps / Call to Action

- Additional editors for this doc?
- Start submitting PRs
  - Ideas for additional content
  - Implementation experience / best practices
  - Open questions

Features at Risk: CBOR and JSON (Manu, 30 min)

What is a Feature At Risk?

To publish a Candidate Recommendation, in addition to meeting the general requirements for advancement a Working Group ... the W3C: ... may identify features in the document as "at risk". These features may be removed before advancement to Proposed Recommendation without a requirement to publish a new Candidate Recommendation.

-- W3C Process 2020

Basically, if you think a feature might be removed or changed, mark it as "at risk"

Why would something be "at risk"?

- Possibility of less than two independent implementations of a specific feature.
- At least one normative change to a feature is expected.
- Specification feature is not thoroughly documented.
- Enough independent implementations, but shaky deployment experience.
- Looming formal objections.
- General feelings of inadequacy and malaise.

Areas of Concern

- JSON-only DID Documents (JSON Representation)
- CBOR-only DID Documents (CBOR Representation)
- Services in a DID Document (Privacy Concerns)
- publicKey (can we do this before CR?)
- Also: We need to vet every normative statement in the spec

Possible Mitigations

- JSON-only DID Documents
  - Two independent JSON-only implementations
  - Test suite contributions for JSON-only implementations
- CBOR-only DID Documents
  - Two independent CBOR-only implementations
  - Test suite contributions for CBOR-only implementations
- Services in a DID Document (Privacy Concerns)
  - Alternative "seeAlso" proposal (preferably before CR)
  - Worst case: Mark as "at risk"
- publicKey
  - Replace with "verificationMethod" (preferably before CR)
  - Worst case: Mark as "at risk"

Other At Risk Features?

- TBD
  -

End of Day 1

Decentralized Identifier WG Virtual Face-to-Face meeting

Day 2: July 1, 2020
Chairs: Brent Zundel, Dan Burnett
Location: the interwebs

Today’s agenda

12:00		
12:00	Review and Agenda	Chairs
12:15	Test Suite Design	Dmitri
13:30	Break	
14:00	Key Representations and Crypto Algorithms/Specs	Orie
15:00	Working Session	Chairs

Test Suite Design (Dmitri, 75 min)

DID-Core Test Suite Challenges

1. Cross-Language Library Testing
2. Data Model Testing
3. ***Abstract*** Data Model Testing

Challenge 1: Cross-Language Testing

Q: How do we test libraries implemented in multiple languages?

A: CLI / Unix pipes / `stdin` to the rescue.

Challenge 2: Data Model Testing

Q: What are we actually testing?

A:

- DID (the URI) ABNF
- DID Document Data Model features
- What about the `resolve()` / `resolveStream()` contract?

Let's step back...

Q: What's the goal of the test suite?

- Ensure the spec is *implementable*
- Get an idea of which features are At Risk
- A rough census poll of state of DID method implementations?

Data Model Testing - DID URIs

How do we test DID URIs?

- Pass in an example DID for the method
- Apply ABNF/regex

Data Model Testing - DID Documents

Q: How do we test/validate the (abstract) data model?

A:

1. Pass in a DID Document
2. Validate it (against spec MUSTs / schema)
3. Tally the features used (e.g. Is anyone using service endpoints? Are there *very common* features not in the spec?)

Data Model Testing - DID Documents

Where are we getting these DID Documents?

1. ~~Static example DID Documents for a method.~~
    a. Pro: Simple.
    Con: Is static. Not actually testing the implementations / libs.

2. Lib *generates* the DID & DID Document.
    a. Pro: Actually tests the implementation over time.
    Con: Unlike `resolve()`, generating is not specified. (It's not even the C from CRUD).

3. Lib `resolve()`s a DID to a DID Document.
    a. Pro: Also tests the resolve() contract.
    Con: Requires net access. Possible over-reliance on Universal Resolver?
    Also, this *is* a Data Model spec...

Abstract and Concrete Representations

What about the Abstract/Concrete part of the Data Model?

We want to test following DID Doc representations:

- JSON-LD
- JSON only
- CBOR (CBOR / CBOR-LD / Dag-CBOR / ?)

Proposal: We pass in `content-type / accept` in the test harness CLI.

Other Test Suite Challenges

Report Generation

1. We want to produce (a mostly automated) Implementation Report, like https://w3c.github.io/vc-test-suite/implementations/

2. For public/open source libs, make updates easy and automatic (from repos)

3. But also still support proprietary implementation reports

Report Generation

Ideal Workflow

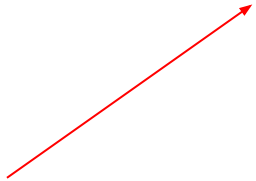
1. For free/open-source implementation libs:
   PRs to a YAML config file (e.g. universal-resolver/docker-compose.yml)

   a. Github repo + Travis-CI like 'install' / 'start' script sufficient?
   b. Use Dockerhub? (or is that overkill)
   c. Run test suite -> downloads and installs all libs -> generates report

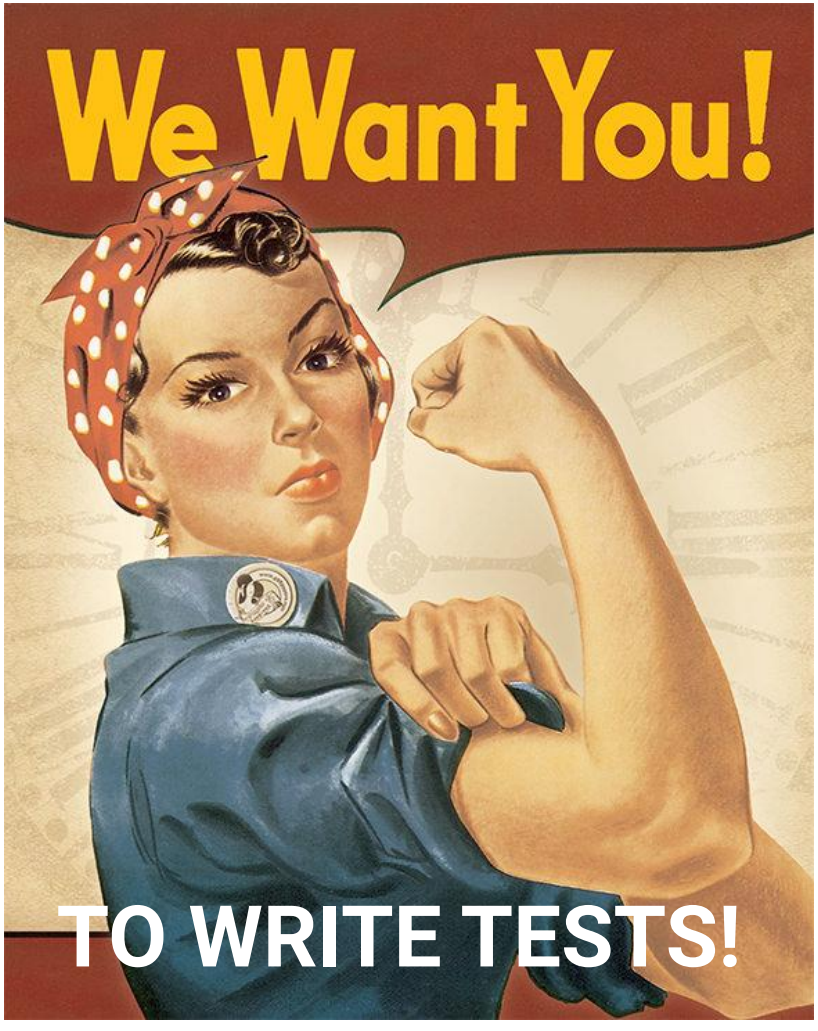
2. For proprietary implementation libs:
   PRs to submit Implementation Reports? (e.g. vc-test-suite/implementations)

   a. YAML instead of JSON (allows for comments, etc)

https://github.com/w3c-ccg/did-test-suite



(We should probably move it to the DID WG org.)


We Want You!
TO WRITE TESTS!

Break (30 min)

Key Representations and Crypto Algorithms (Orie, 60 min)

What is security made of?

- Excellent documentation
- Transparent / simple implementations (not obscurity)
- Academic analysis
- Observed ability to defend value
- Blessings from authorities (?)

- Formal verification (Coq, Agda, TLA+, Proverif)
- Frequent rotations
- Short expirations
- Performance / lifetime trade offs
- Extreme Paranoia

Questions for the Group

Should modern cryptographic tooling / data models or standards support legacy crypto?

Who decides when it’s not a good idea to use something anymore?

How do(?) we communicate risks associated with specific key types and algorithms?

How do we encode “key purpose” or “key use”?... (again not everyone uses JOSE).

How will I get the features I want if I can’t implement them myself?

Who watches the watchmen?

Why would we trust that this standards governance process won’t be compromised to the advantage of interested parties? (is this happening right now)

Background

https://github.com/panva/jose https://github.com/square/go-jose https://github.com/mpdavis/python-jose https://github.com/digitalbazaar/forge	Popular Signing and Encryption libraries exist! 82,712 Weekly Downloads (for panva/jose) 8,814,510 Weekly Downloads (for forge)
https://www.iana.org/assignments/jose/jose.xhtml	Nice formal registry...
web-signature-encryption-algorithms	Lots supported Signing and Encryption Algorithms...
NIST FIPS 186-5-draft	Some support for modern crypto...
https://safecurves.cr.yp.to/rigid.html	Lots of “IANA Recommended”, “Manipulatable” crypto….

“American security is better served with unbreakable end-to-end encryption than it would be served with one or another front door, backdoor, side door, however you want to describe it.”
- Gen. Michael Hayden

“I no longer trust the constants. I believe the NSA has manipulated them through their relationships with industry.”
- Bruce Schneier

Background (2) - popularity leads to exploitation…

“No Way, JOSE! Javascript Object Signing and Encryption is a Bad Standard That Everyone Should Avoid”

- https://paragonie.com/blog/2017/03/jwt-json-web-tokens-is-bad-standard-that-everyone-should-avoid

“The most blatant way to make your app vulnerable is to get the **alg** header, and then immediately proceed to verify the JWT's HMAC or signature, without first checking if that JWT **alg** is permitted. What will happen to your app if it gets an unsecured JWT with `alg = none`?”

- https://connect2id.com/products/nimbus-jose-jwt/vulnerabilities

If you are using go-jose, node-jose, jose2go, Nimbus JOSE+JWT or jose4 with ECDH-ES please update to the latest version. RFC 7516 aka JSON Web Encryption (JWE) Invalid Curve Attack. This can allow an attacker to recover the secret key of a party using JWE with Key Agreement with Elliptic Curve Diffie-Hellman Ephemeral Static (ECDH-ES), where the sender could extract receiver’s private key.

- https://blogs.adobe.com/security/2017/03/critical-vulnerability-uncovered-in-json-encryption.html

What is crypto-agility?

"**Crypto-agility** (*cryptographic agility*) is a practice paradigm in designing information security systems that encourages support of rapid adaptations of new cryptographic primitives and algorithms without making significant changes to the system's infrastructure. Crypto-agility acts as a safety measure or an incident response mechanism when a cryptographic primitive of a system is discovered to be vulnerable.[1] A security system is considered crypto agile if its cryptographic algorithms or parameters can be replaced with ease and is at least partly automated.[2][3]„

"The names of the algorithms used should be communicated and not assumed or defaulted."

- https://en.wikipedia.org/wiki/Crypto-agility

"as written, this library greatly expands the attack surface on anyone using it and goes against the whole reason that cryptosuites exist (that is, people that know better should be picking an extremely limited set of options, not enabling a kitchen sink approach, which this library does)."

- https://github.com/w3c-ccg/lds-jws2020/issues/4

Why is IANA recommending NIST Curves, why are node and go libraries supporting a "kitchen sink"… are we dancing around just saying "JOSE considered unsafe?"

Are DID WG members the "people that know better"?

How is that stuff in a did document used?

- Verify Digital Signatures
  - https://w3c-ccg.github.io/security-vocab/#assertionMethod
  - https://www.iana.org/assignments/jose/jose.xhtml#web-signature-encryption-algorithms
- Derive Secrets for use with Encryption Algorithms
  - https://w3c-ccg.github.io/security-vocab/#keyAgreement
  - https://www.iana.org/assignments/jose/jose.xhtml#web-signature-encryption-algorithms

- How much of JOSE will we actually support?
- Where will we document how to use all/any of JOSE features with DIDs?
- Who will do this work?
- Will anyone do this work?
- How do people use things that are not documented or supported anywhere?

Reminder that PGP exists...

```
(async () => {
    const publicKeyArmored = `-----BEGIN PGP PUBLIC KEY BLOCK-----...`// Public key
    const [privateKeyArmored] = `-----BEGIN PGP PRIVATE KEY BLOCK-----...`// Encrypted private key
    const passphrase = `yourPassphrase`; // Password that private key is encrypted with
    const privateKey = (await openpgp.key.readArmored([privateKeyArmored])).keys[0];
    await privateKey.decrypt(passphrase);
    const readableStream = new ReadableStream({
        start(controller) {
            controller.enqueue('Hello, world!');
            controller.close();
        }
    });
    const encrypted = await openpgp. encrypt({
        message: openpgp.message.fromText(readableStream),                  // input as Message object
        publicKeys: (await openpgp.key.readArmored(publicKeyArmored)).keys, // for encryption
        privateKeys: [privateKey]                                           // for signing (optional)
    });
    const ciphertext = encrypted.data; // ReadableStream containing '-----BEGIN PGP MESSAGE ... END PGP MESSAGE-----'
    const decrypted = await openpgp. decrypt({
        message: await openpgp.message.readArmored(ciphertext),             // parse armored message
        publicKeys: (await openpgp.key.readArmored(publicKeyArmored)).keys, // for verification (optional)
        privateKeys: [privateKey]                                           // for decryption
    });
    const plaintext = await openpgp.stream.readToEnd(decrypted.data); // 'Hello, World!'
})();
```

Agility / Performance

Reminder that Minimal Cipher exists...

```
// Retrieve them from config, a ledger, registry or back channel
const keyAgreementKey = await fetchFromSomewhere();
// or derive them from an existing Ed25519 signing key
const X25519KeyPair = require('x25519-key-pair');
const {Ed25519KeyPair} = require('crypto-ld');
const edKeyPair = await Ed25519KeyPair.generate();
const keyAgreementKey = X25519KeyPair.fromEdKeyPair(edKeyPair);
// Don't forget to set your key's id. For example, DID + fingerprint
keyAgreementKey.id = `${did}#${keyAgreementKey.fingerprint()}`;
// or derive them from an authentication key extracted from DID Document
const didDoc = await veresDriver.get({did});
const authnKey = didDoc.getVerificationMethod({proofPurpose: 'authentication'});
const edKeyPair = await Ed25519KeyPair.from(authnKey);
const keyAgreementKey = X25519KeyPair.fromEdKeyPair(edKeyPair); keyAgreementKey.id = authnKey.id;
const recipient = {
  header: {
    kid: keyAgreementKey.id,
    alg: 'ECDH-ES+A256KW'
  }
}
const recipients = [recipient];
```

Agility / Performance

Reminder that DID Comm is being built...

```
const plaintext = {
    id : "urn:uuid:ef5a7369-f0b9-4143-a49d-2b9c7ee51117",
    type : "hello-world-message-type",
    from : "urn:uuid:8abdf5fb-621e-4cf5-a595-071bc2c91d82",
    expiry : 1516239022,
    time_stamp : 1516269022,
    body : { message: "Hello world!" }
};
const recipientJWK = JSON.parse(await fs.readFile("example-keys/example-recipient.json"));
const key = jose.JWK.asKey(recipientJWK);
const jwe = new jose.JWE.Encrypt( JSON.stringify(plaintext),{typ : 'JWM',enc : "A256GCM"});
jwe.recipient(key, { kid : key.kid, alg : 'ECDH-ES+A256KW' });
jwe.recipient(key, { kid : key.kid, alg : 'ECDH-ES+A256KW' });
const jwmGeneral = jwe.encrypt('general');
```

Agility / Performance

Encoding a key type and purpose in the name and verification relationship.


```
{
  "@context": [
    "https://w3id.org/did/v0.11"
  ],
  "id": "did:key:z6MkgMKSnFDE8kDjHKS9sUDnnPMF1MrYDKNBx2PW8g8PDia3",
  "publicKey": [
    {
      "id": "did:key:z6MkgMKSnFDE8kDjHKS9sUDnnPMF1MrYDKNBx2PW8g8PDia3#z6MkgMKSnFDE8kDjHKS9sUDnnPMF1MrYDKNBx2PW8g8PDia3",
      "type": "Ed25519VerificationKey2018",
      "controller": "did:key:z6MkgMKSnFDE8kDjHKS9sUDnnPMF1MrYDKNBx2PW8g8PDia3",
      "publicKeyBase58": "2u4QBzxnoCjGApbTBuFwwHoFBnagoS7qG1UaJQANJVnf"
    }
  ],
  "assertionMethod": ["did:key:z6MkgMKSnFDE8kDjHKS9sUDnnPMF1MrYDKNBx2PW8g8PDia3#z6MkgMKSnFDE8kDjHKS9sUDnnPMF1MrYDKNBx2PW8g8PDia3"],
  "keyAgreement": [
    {
      "id": "did:key:z6MkgMKSnFDE8kDjHKS9sUDnnPMF1MrYDKNBx2PW8g8PDia3#zC8uEokkH3X6dZAi8yJgD5EgN2ZWvA3i3wsA597tDn96qf",
      "type": "X25519KeyAgreementKey2019",
      "controller": "did:key:z6MkgMKSnFDE8kDjHKS9sUDnnPMF1MrYDKNBx2PW8g8PDia3",
      "publicKeyBase58": "9QX6PZTr9MM84fAWuBe75WgjKF7CV1SSyGn7GoRP2rjT"
    }
  ]
}
```


“The names of the algorithms used should be communicated and not assumed or defaulted.”

Json Web Key 2020 Proposal

W3C CCG will maintain a JSON-LD Signature Suite which documents how to use JOSE with DIDs... the key representation will support JWA / JWS / JWT / JWE.

Keep the JSON-LD side of working with JOSE as simple as possible.... While actually supporting interoperability on both fronts.

- The suite will provide better guidance than IANA does, on what “recommended” means.
- The suite will support the needs of “Pure JSON” / “JOSE only” folks.
- JOSE features that are not documented / contributed to will NOT be included.
- The suite will be year timestamped, and updated as the security landscape changes.
- The DID WG will reference this suite, as it does for others, via the spec registries.

Encoding a key purpose in verification relationship

```
{
  "@context": [
    "https://w3id.org/did/v0.11",
    {
      "@base": "did:example:123"
    }
  ],
  "id": "did:example:123",
  "publicKey": [
    {
      "type": "JsonWebKey2020",
      "id": "#j3k7usPJq1C73Dt4tMaedVjbfV4Dd9n9EpRCvHVA3as",
      "controller": "did:example:123",
      "publicKeyJwk": {
        "kid": "j3k7usPJq1C73Dt4tMaedVjbfV4Dd9n9EpRCvHVA3as"
        "kty": "EC",
        "crv": "P-384",
        "x": "x1_e618lJHBf8EcjWxfEyZgPwUs-2SIZnnArinev_CAtgvFZchBubsUAeqtkkxH9",
        "y": "FI6bMaEQVDttPqfY05wPZOfL18S6_MFPxLTozKJ10UPdK2OCkGoSbYHLxFPeewBN",
      }
    }
  ],
  "assertionMethod": ["#j3k7usPJq1C73Dt4tMaedVjbfV4Dd9n9EpRCvHVA3as"],
  "keyAgreement": ["#j3k7usPJq1C73Dt4tMaedVjbfV4Dd9n9EpRCvHVA3as"]
}
```

“The names of the algorithms used should be communicated and not assumed or defaulted.”

Next steps for JOSE

- https://docs.microsoft.com/en-us/microsoft-edge/dev-guide/windows-integration/web-authentication#authenticate-your-user
  - WebAuthN uses `publicKeyJwk` ... but we've seen almost 0 contribution on this front from DID WG members.... It's still not even supported in CCG vocabulary....
- https://github.com/microsoft/VerifiableCredentials-Crypto-SDK-Typescript/issues/12
  - Does this make sense given the near complete lack of support for JOSE today?

Who is going to do the work to make JOSE and DIDs work together?

Is the future Multicodec / Multibase?

- Compact double clickable string and binary representations.
- Friendlier towards other programing languages (not everyone owns a browser)
- Growing adoption within the blockchain ecosystem
- Registries are easier to update quickly and safely
  - Already used to describe bls12_381-g1 and g2 for JSON-LD ZKPs
- Key representations less of a foot gun...what does “jwk.d” do?
- Fingerprint algorithms less of a foot gun… JWK has no canonical representation... Sidetree forced to rely on JCS as well.
- NIST Curves aren’t even registered… (is this a good thing?)

Is base58 the “defacto” standard key representation for DLT keys today?

Recommendations

- Recommend we allow base58 key representations to be valid for all key types, in addition to JWK

- Recommend we include disclaimers about JOSE and algorithmic agility in the Security Considerations section of the DID Core Spec.

- Recommend we work together to ensure that Json Web Key 2020 is usable.
  - Meets the needs of “Pure Json”, legacy crypto, NIST / FIPS crypto... that *some* backward facing support for DID interoperability exists.

- Recommend the did spec registries provide some indication of security implications / independent vendor implementations for registered crypto. Ideally a green, yellow, red scale.

Issues

- https://github.com/w3c/did-spec-registries/issues/66
- https://github.com/w3c/did-spec-registries/issues/46
- https://github.com/w3c/did-core/issues?q=is%3Aissue+is%3Aopen+jose

Working Session (Chairs.Brent, 30 min)

Registries

- Should we require principled requirements for inclusion of properties?
- What registry instructions are needed?

ethereumAddress

https://github.com/w3c/did-spec-registries/pull/73

Can we declare feature freeze now?

For example:

- No additional core properties
- No additional representations
- No additional DID Method operations

Basically, if it’s not proposed already in an issue or PR it won’t go into V1.0 of the spec (but could perhaps go into a registry anytime, depending on what it is)

TPAC 2020

Virtual this year

October 2020, format still being worked out

End of Day 2

Documents and Background

- Home: https://www.w3.org/2019/did-wg/
- Charter: https://www.w3.org/2019/09/did-wg-charter.html
- Primer: https://w3c-ccg.github.io/did-primer/

DID WG Scope

- Define the DID URI scheme.
- Recommend a data model and syntax(es) for the expression of Decentralized Identifier Documents, including one or more core vocabularies.
- Recommend a set of requirements for DID Method specifications that are conformant with the data model and syntax(es).
- Provide a rubric of decentralized characteristics for DID Method specifications.
- Concentrate their efforts on the initial use cases with a particular focus on enabling future specification and implementation of Identity and Access Management.
- Define extension points enabling authentication, signing and cryptography mechanisms.
- With the initial use cases document as input, the WG will produce a NOTE at the end of the process that is a refined Use Cases document.
- Establish a deterministic mapping between DID method identifiers and the resolution process used to resolve that DID method.

DID WG Out of Scope

- Authentication or Authorization Protocols
- Browser APIs
- Specific DID Method specifications or Protocol specifications
- "Solving Identity" on the Web
- Defining specific authentication, signing, or cryptography mechanisms. Scope is limited to defining extension points for these mechanisms.