25th July 2024

Supply Chain Integrity, Transparency, and Trust (SCITT)

This session is being recorded

Note Well

This is a reminder of IETF policies in effect on various topics such as patents or code of conduct. It is only meant to point you in the right direction. Exceptions may apply. The IETF's patent policy and the definition of an IETF "contribution" and "participation" are set forth in BCP 79; please read it carefully.

As a reminder:

- By participating in the IETF, you agree to follow IETF processes and policies.
- If you are aware that any IETF contribution is covered by patents or patent applications that are owned or controlled by you or your sponsor, you must disclose that fact, or not participate in the discussion.
- As a participant in or attendee to any IETF activity you acknowledge that written, audio, video, and photographic records of meetings may be made public.
- Personal information that you provide to IETF will be handled in accordance with the IETF Privacy Statement.
- As a participant or attendee, you agree to work respectfully with other participants; please contact the ombudsteam (https://www.ietf.org/contact/ombudsteam/) if you have questions or concerns about this.

Definitive information is in the documents listed below and other IETF BCPs. For advice, please talk to WG chairs or ADs:

- BCP 9 (Internet Standards Process)
- BCP 25 (Working Group processes)
- BCP 25 (Anti-Harassment Procedures)
- BCP 54 (Code of Conduct)
- BCP 78 (Copyright)
- BCP 79 (Patents, Participation)
- https://www.ietf.org/privacy-policy/(Privacy Policy)

Note Really Well

- IETF meetings, virtual meetings, and mailing lists are intended for professional collaboration and networking, as defined in the IETF Guidelines for Conduct (RFC 7154), the IETF Anti-Harassment Policy, and the IETF Anti-Harassment Procedures (RFC 7776). If you have any concerns about observed behavior, please talk to the Ombudsteam, who are available if you need confidentiality to raise concerns confident about harassment or other conduct in the IETF.
- The IETF strives to create and maintain an environment in which people of many different backgrounds and identities are treated with dignity, decency, and respect. Those who participate in the IETF are expected to behave according to professional standards and demonstrate appropriate workplace behavior.
- IETF participants must not engage in harassment while at IETF meetings, virtual meetings, social events, or on mailing lists. Harassment is unwelcome hostile or intimidating behavior—in particular, speech or behavior that is aggressive or intimidates.
- If you believe you have been harassed, notice that someone else is being harassed, or have any other concerns, you are encouraged to raise your concern in confidence with one of the Ombudspersons.

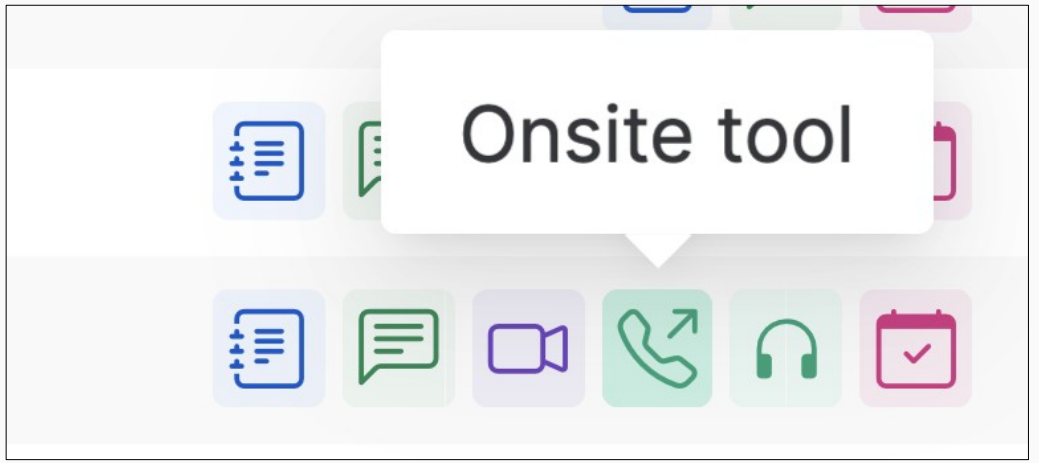
IETF 120 Meeting Tips

In-person participants

- Make sure to sign into the session via Datatracker or the QR Code in this session.
- Use Meetecho (usually the “Meetecho lite”) client to:
  - join the mic queue
  - participate in shows of hands
- *Keep audio and video off if not using the onsite version.*

Remote participants

- Make sure your audio and video are off unless you are chairing or presenting during a session.
- Use of a headset is strongly recommended.



Note taker?

Please assist in taking notes

HedgeDoc

IETF

Agenda

- Welcome and Introduction (5 min): Chairs
- SCITT Overview (10 mins): Steve Lasker
- Recap since 119 (20 mins): Henk Birkholtz/Roy Williams
  - SCITT Architecture
- SCRAPI (15 mins): Jon Geater (chair hat aside)
- Hackathon Report (10 min): Steve Lasker
- Next Steps (10 min): Chairs
- AOB Open Mic (15 min): All
- Wrap-up and Conclusion (5 min): Chairs

SCITT Overview

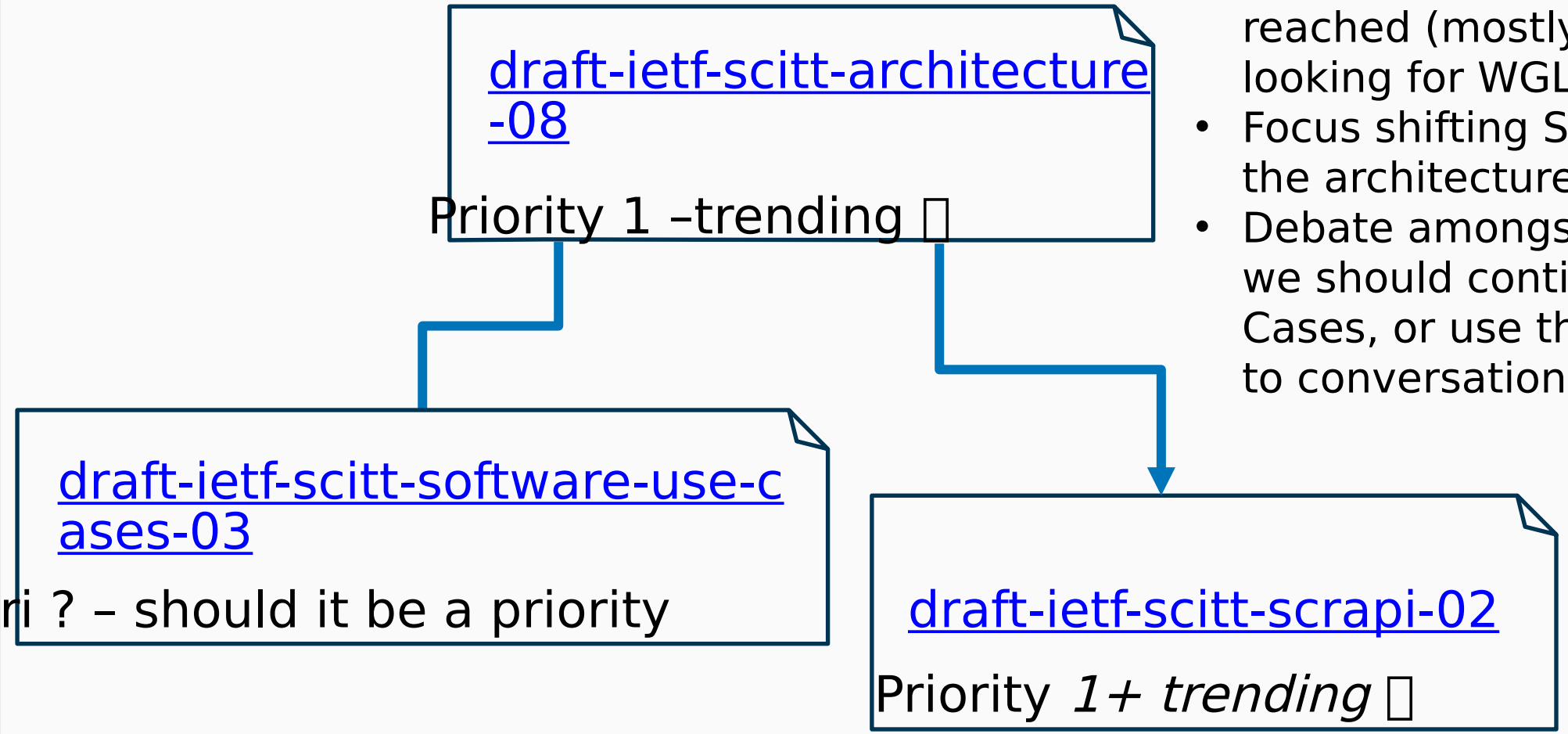
Steve Lasker

7

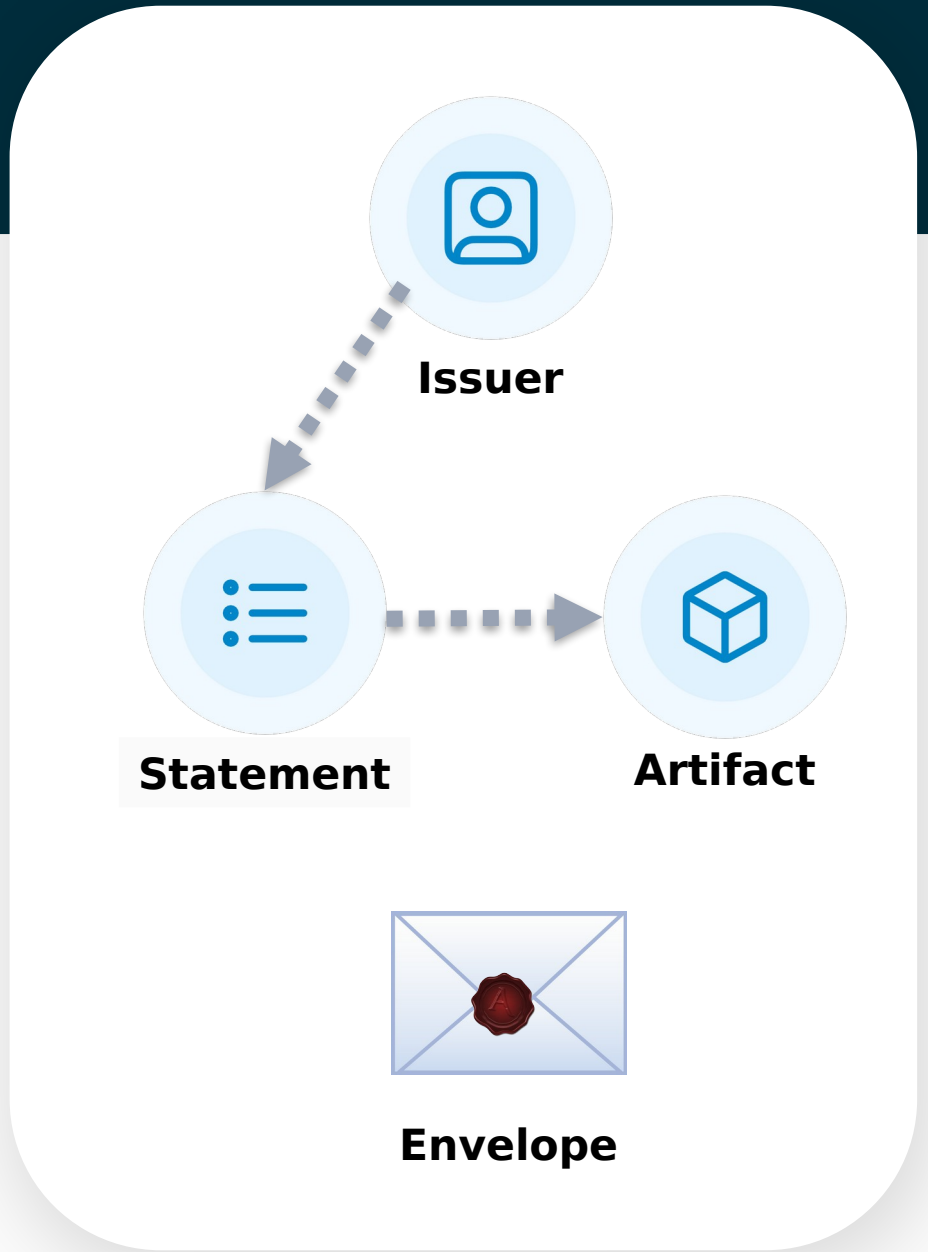
What is SCITT

- **Working Group Charter**
  The Supply Chain Integrity, Transparency, and Trust (SCITT) WG will define a set of interoperable building blocks that will allow implementers to build integrity and accountability into software supply chain systems to help assure trustworthy operation.

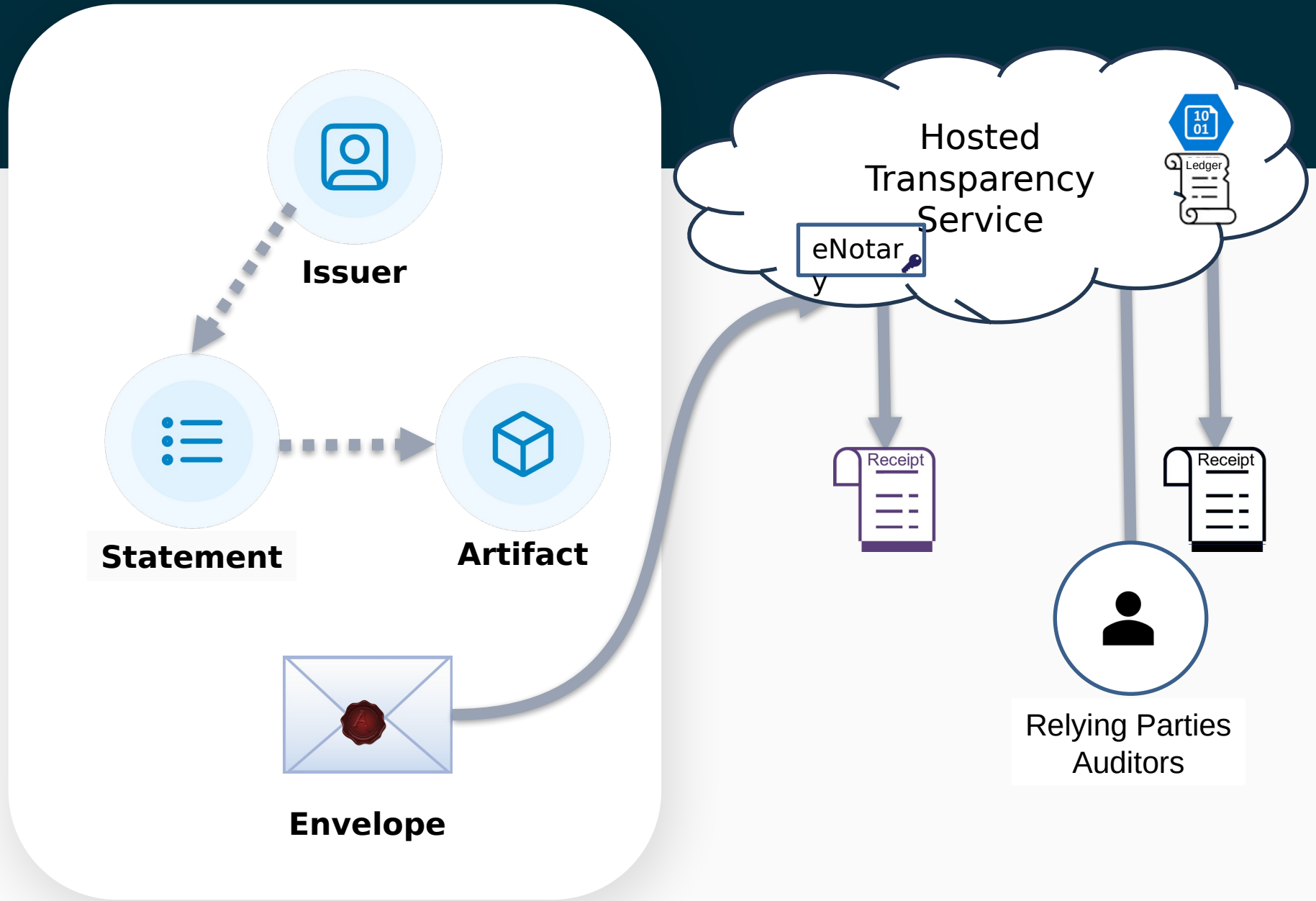
Internet Drafts



- The Architecture draft has reached (mostly) stabile and looking for WGLC
- Focus shifting SCR**API**, validating the architecture is complete
- Debate amongst the authors if we should continue the Use Cases, or use them as reference to conversations



Issuers make Statements about Artifacts



Signed Statements Registered on a SCITT Transparency Service

```
Signed_Statement =

#6.18(COSE_Sign1)
```

Envelope

Issuer

Statement

Artifact

Protected_Header

```
 &(CWT_Claims: 15) =>
CWT_Claims
 ? &(alg          : 1) => int
 ? &(content_type: 3) => tstr /
uint
 ? &(kid          : 4) => bstr
 ? &(x5t          : 34) =>
COSE_CertHash
```

CWT_Claims

```
        issuer :
tstr,   subject
 *      : int => any
```

```
Signed_Statement =

#6.18(COSE_Sign1)
```

Envelope

Envelope : COSE_SIGN1

```
protected    : bstr .cbor
Protected_Header,
unprotected  : Unprotected_Header,
payload      :
nil /
signature    :
bstr / nil,
bstr
```

Protected_Header

```
&(CWT_Claims: 15) =>
CWT_Claims
? &(alg          : 1) => int
? &(content_type: 3) => tstr /
uint
? &(kid          : 4) => bstr
? &(x5t          : 34) =>
COSE_CertHash
* int => any
```

CWT_Claims

```
issuer :
tstr,
subject
: tstr /
* int => any
```

Issuer

Statement

Artifact

```
Signed_Statement =

#6.18(COSE_Sign1)
```

Envelope

```
Envelope : COSE_SIGN1
       protected     : bstr .cbor
Protected_Header
       unprotected : Unprotected_Header,
       payload       :
bstr / nil,
       signature     :
bstr
```

```
Protected_Header
 &(CWT_Claims : 15) =>
CWT_Claims
 ? &(alg          : 1) => int
 ? &(content_type: 3) => tstr /
uint
 ? &(kid          : 4) => bstr
 ? &(x5t          : 34) =>
COSE_CertHash
 * int => any
```

```
CWT_Claims
       issuer :
tstr,  subject
 *     : int => any
```

```
Unprotected_Header
 ? &(receipts : 394)  => [+
Receipt]
 ? &(x5chain  : 33) => COSE_X509
 * int => any
```

Issuer

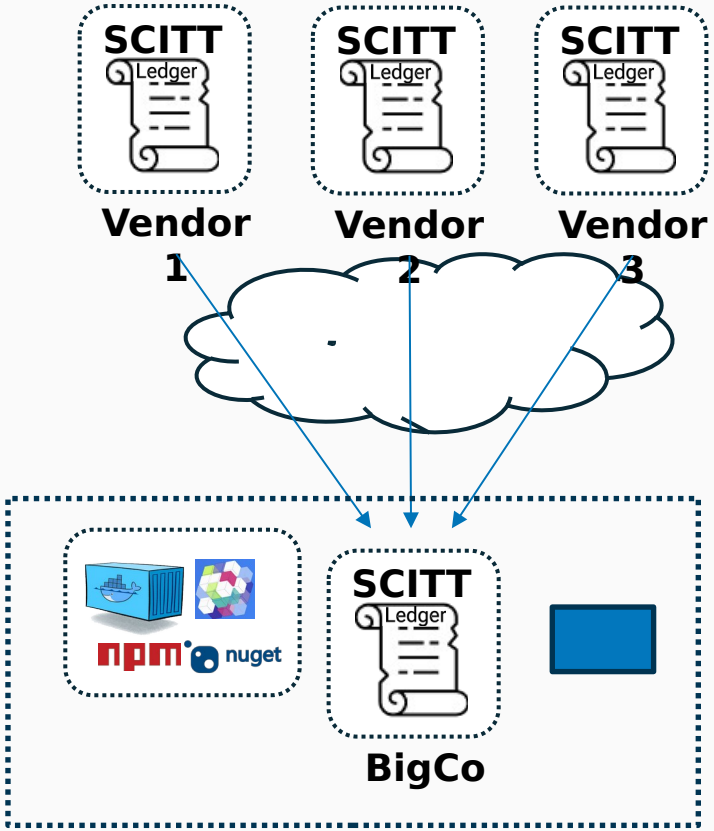
Statement

Artifact

IETF

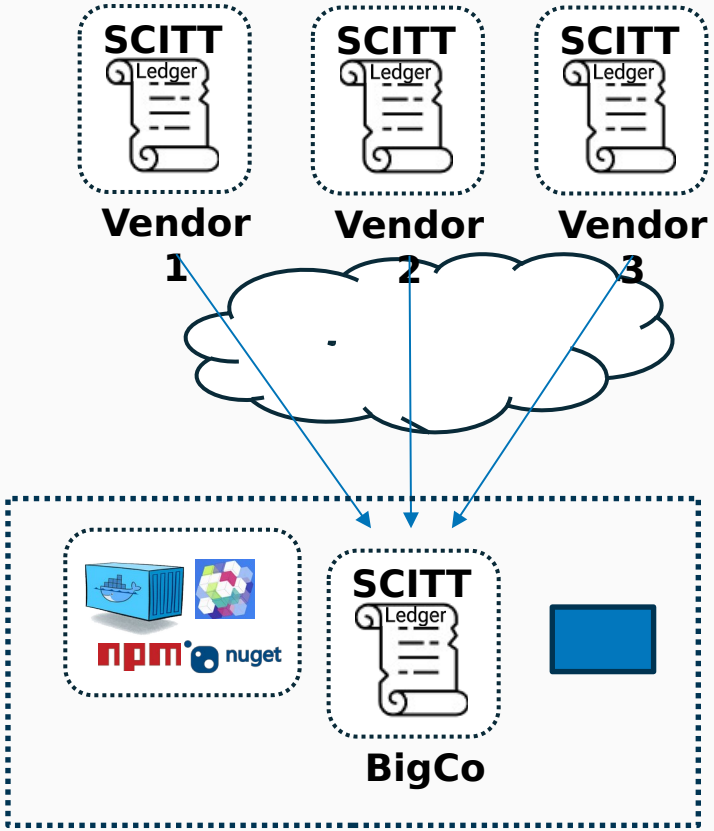
Trusted Update Streams

- What is the most current information, about an artifact?
  - Does it have test case data?
  - What certifications does it have?
  - What's the latest version, and is it applicable to "my" environment?
  - What version has "the business" decided is appropriate for its environments?
  - Who is making the statements, and are they trusted for "my" environment?
  - ***"Latest" does not equal the most reliable or safe!***



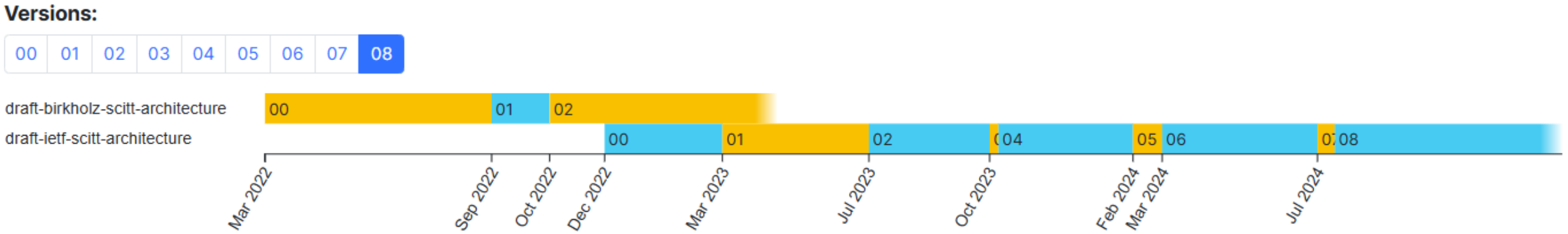
Trusted Update Streams

- What is the most current information, about an artifact?
  - Does it have test case data?
  - What certifications does it have?
  - What's the latest version, and is it applicable to "my" environment?
  - What version has "the business" decided is appropriate for its environments?
  - Who is making the statements, and are they trusted for "my" environment?
  - ***"Latest" does not equal the most reliable or safe!***
- SCITT doesn't define content types (but uses them)
- It does enable **Trust** decisions for **Who** published **What** & **When**



Recap Since 119

Henk



SCITT WG Architecture

Henk Birkholz <henk.birkholz@ietf.contact>
Roy Williams <roywill@exchange.microsoft.com>

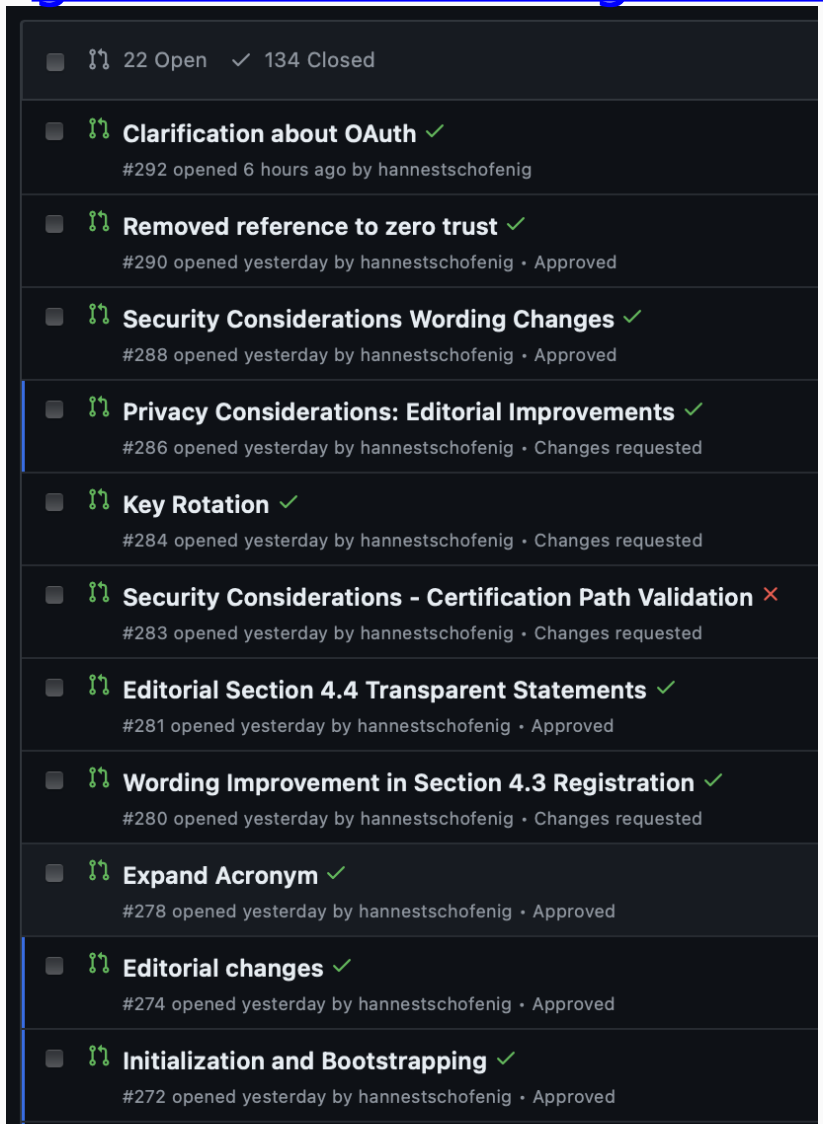
Architecture PRs

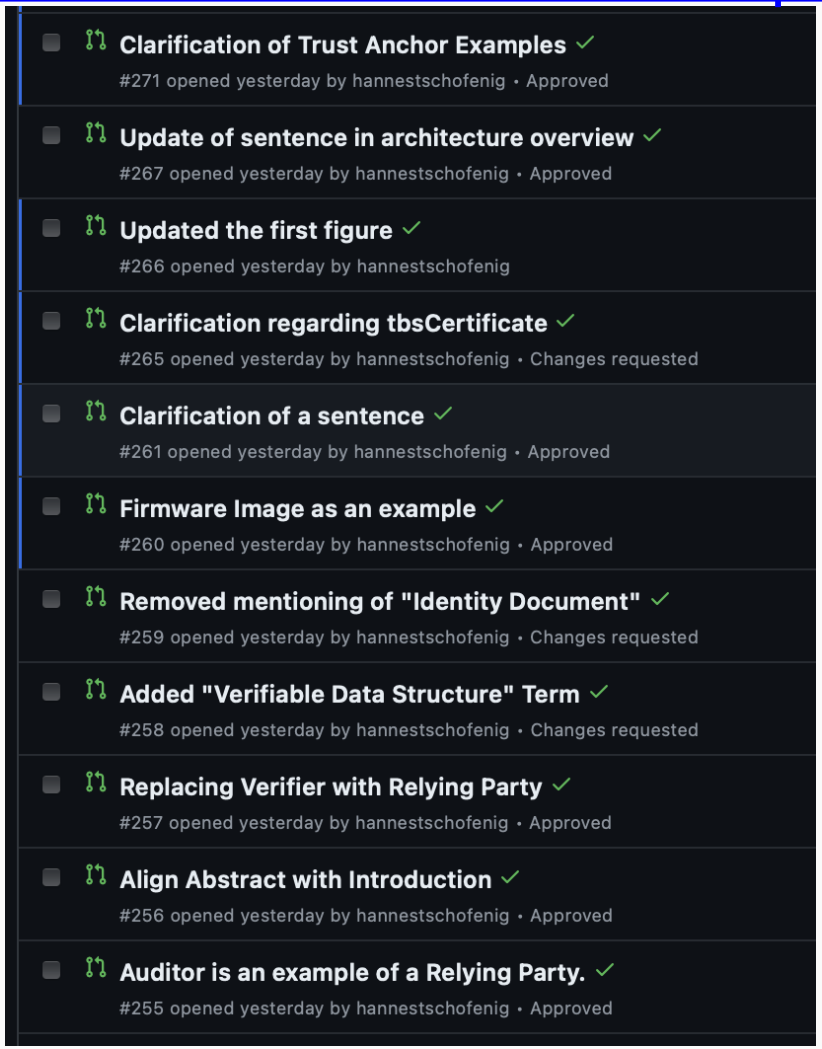
Recent Changes

- Finished outsourcing COSE Receipts: https://datatracker.ietf.org/doc/draft-ietf-cose-merkle-tree-proofs/05/
  - Final step: moved out registration of Header Parameter 394
- Added details on "new"/"refreshed" Receipts
- Final polish to the overview diagram
- Establishing text on x5chain optional use in the unprotected header, which builds on the mandatory x5t in the protected header
- I-D for Hash Envelope spawned in COSE (for SCITT detached payload) https://datatracker.ietf.org/doc/draft-steele-cose-hash-envelope/01/

22 PRs, 13 already approved!

github.com/ietf-wg-scitt/draft-ietf-scitt-architecture/pulls



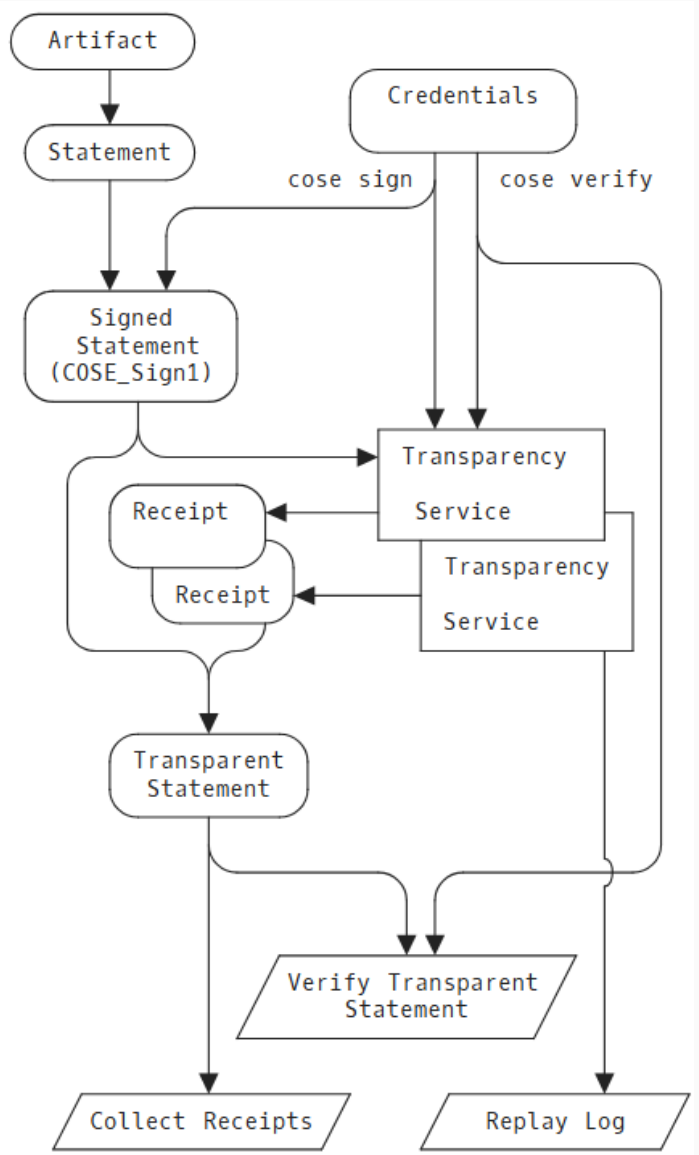


Noteworthy Open Work Items

- Consolidating function names and role names
  - “Verifier Role” becomes “Relying Party”
  - Analogously, Issuer and Relying Party are categorized as Roles
  - Auditor will be a specialization of Relying Party (and is a Role! 😄)

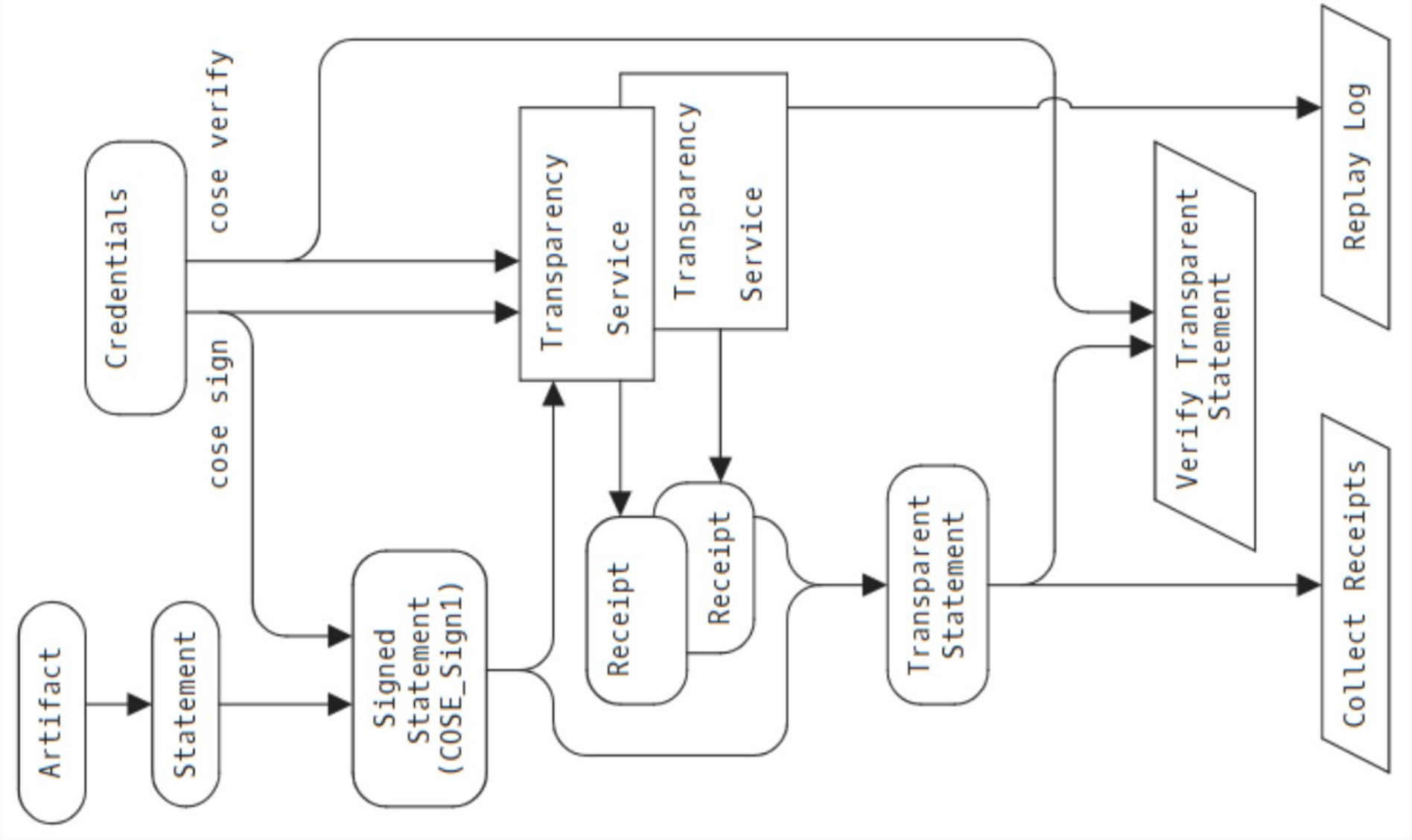
Architectural Overview

- Figure 1 is stabilizing!

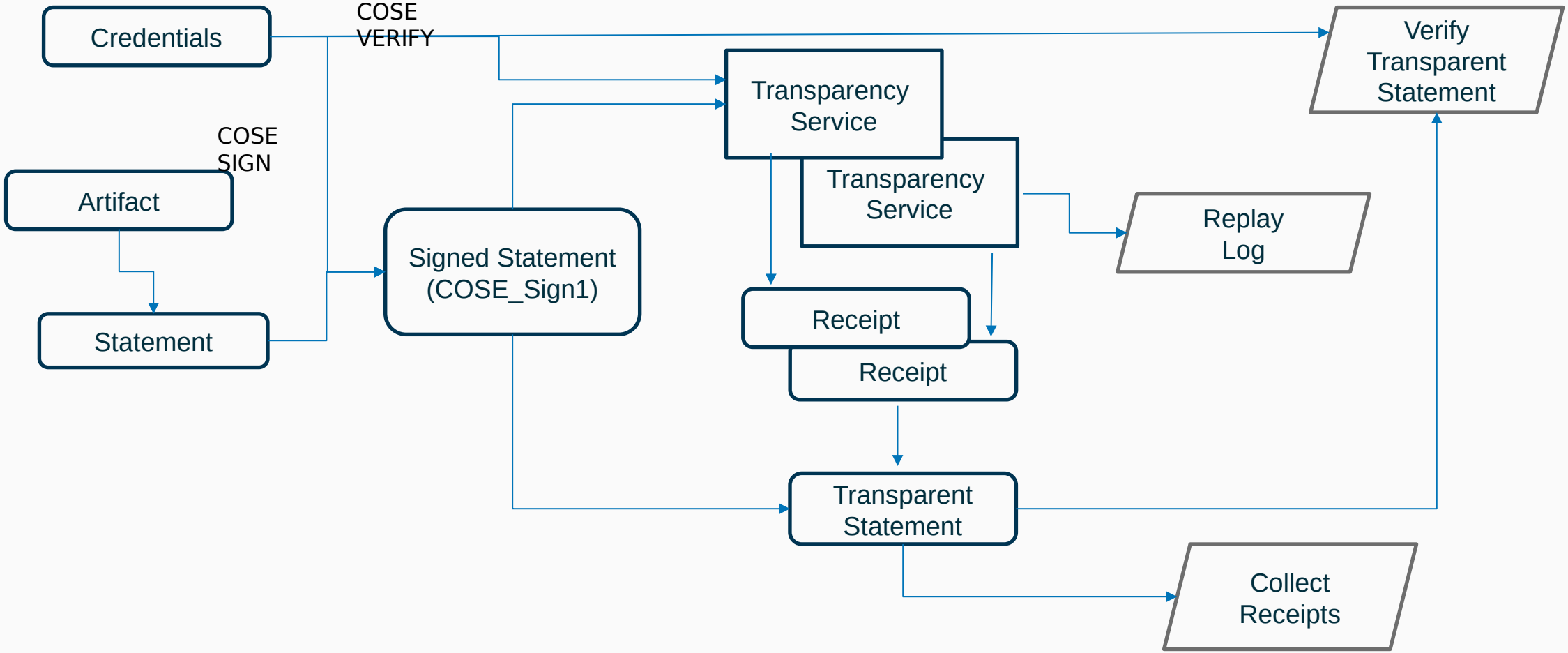


- But on a slide... Figure 1 is not very readable. See next slide 😊

Architecture Overview (tilted 90°)



Architecture Overview (tilted 90°)



SCRAPI:
SCITT REFERENCE API

Jon

SCRAPI: What does it do?

HTTP APIs for Transparency Service-related Operations

- POST Signed Statement (SBOMs, Attestations, End Of Life (EOL), New Versions, …)
- GET Receipt (Signed Proof of Inclusion)
- GET Signed Statements (Signed SBOMs, Signed Attestations, Signed EOL, Signed New Versions, …)
- GET Statements (SBOMs, Attestations, EOL, New Versions, …)
- GET Issuer Metadata
  - GET Verification Keys (for Signed Statements)
- GET Service Metadata (operating profile)
  - GET Registration Policy
  - GET Verification Keys (for Receipts)

SCRAPI Issues

- (meta) Is it roughly conformant to the rules for a draft?

- Issue #2 – Using media types for errors
- Issue #9 – API for collections of statements
- Issue #10 – Details and workflow for signed statement issue
- Issue #12 – async registration appears disallowed by 2.1.2
- Issue #13 - Status 202 requires the resolve receipts endpoint to do double d
- Issue #14 - Resolve Receipt encourages authentication
- Issue #18 – missing intent statements

SCRAPI Issues

Issuer	Relying Party
Y	Y
Y	Y
Y	N
Y	N
Y	N
N	Y
Y	N

(meta) Is it roughly conformant to the rules for a draft?

Issue #2 – Using media types for errors

Issue #9 – API for collections of statements

Issue #10 – Details and workflow for signed statement issue

Issue #12 – async registration appears disallowed by 2.1.2

Issue #13 - Status 202 requires the resolve receipts endpoint to do double d

Issue #14 - Resolve Receipt encourages authentication

Issue #18 – missing intent statements

SCRAPI Issues

Issuer	Relying Party
Y	Y
Y	Y
Y	N
Y	N
Y	N
N	Y
Y	N

(meta) Is it roughly conformant to the rules for a draft?

Issue #2 – Using media types for errors

Issue #9 – API for collections of statements

Issue #10 – Details and workflow for signed statement issue

Issue #12 – async registration appears disallowed by 2.1.2

Issue #13 - Status 202 requires the resolve receipts endpoint to do double d

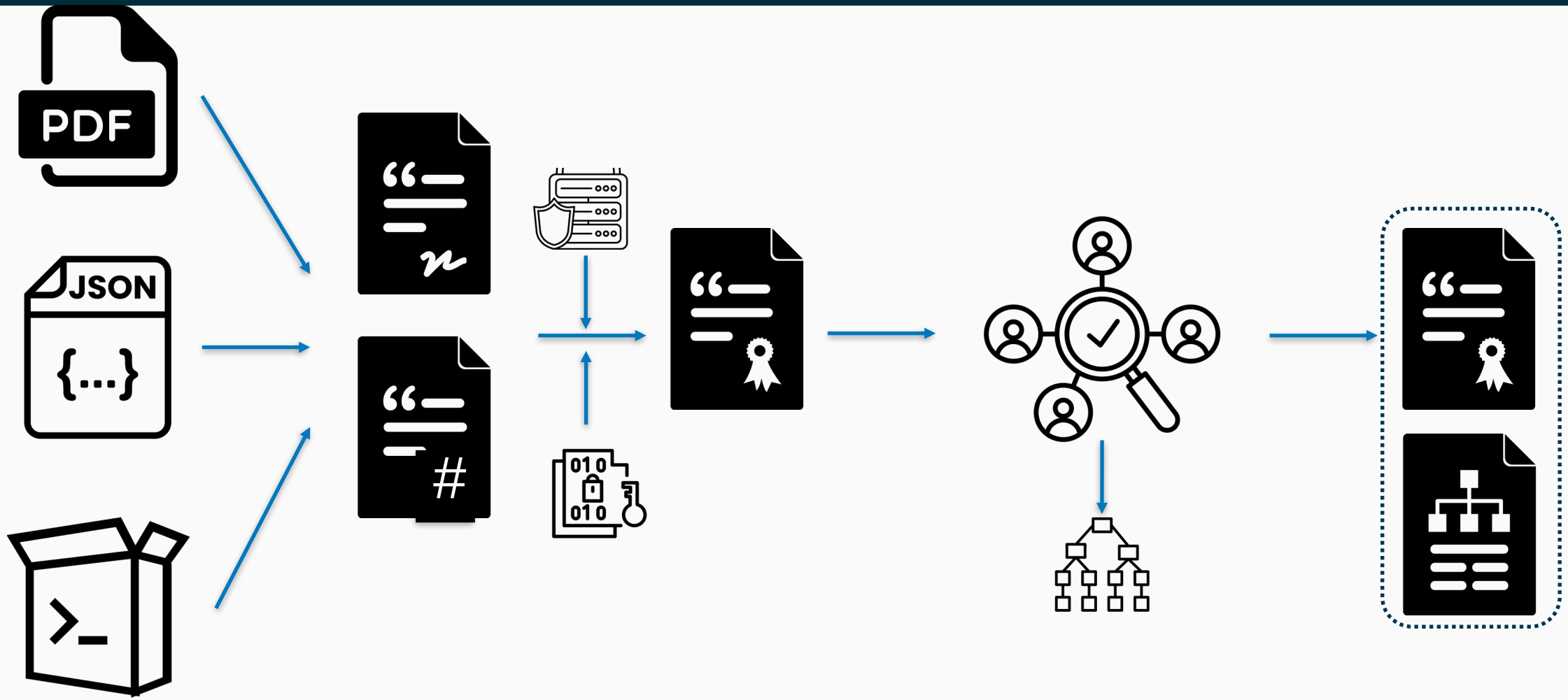
Issue #14 - Resolve Receipt encourages authentication

Issue #18 – missing intent statements

SCRAPI: Issuer role: End-to-end code

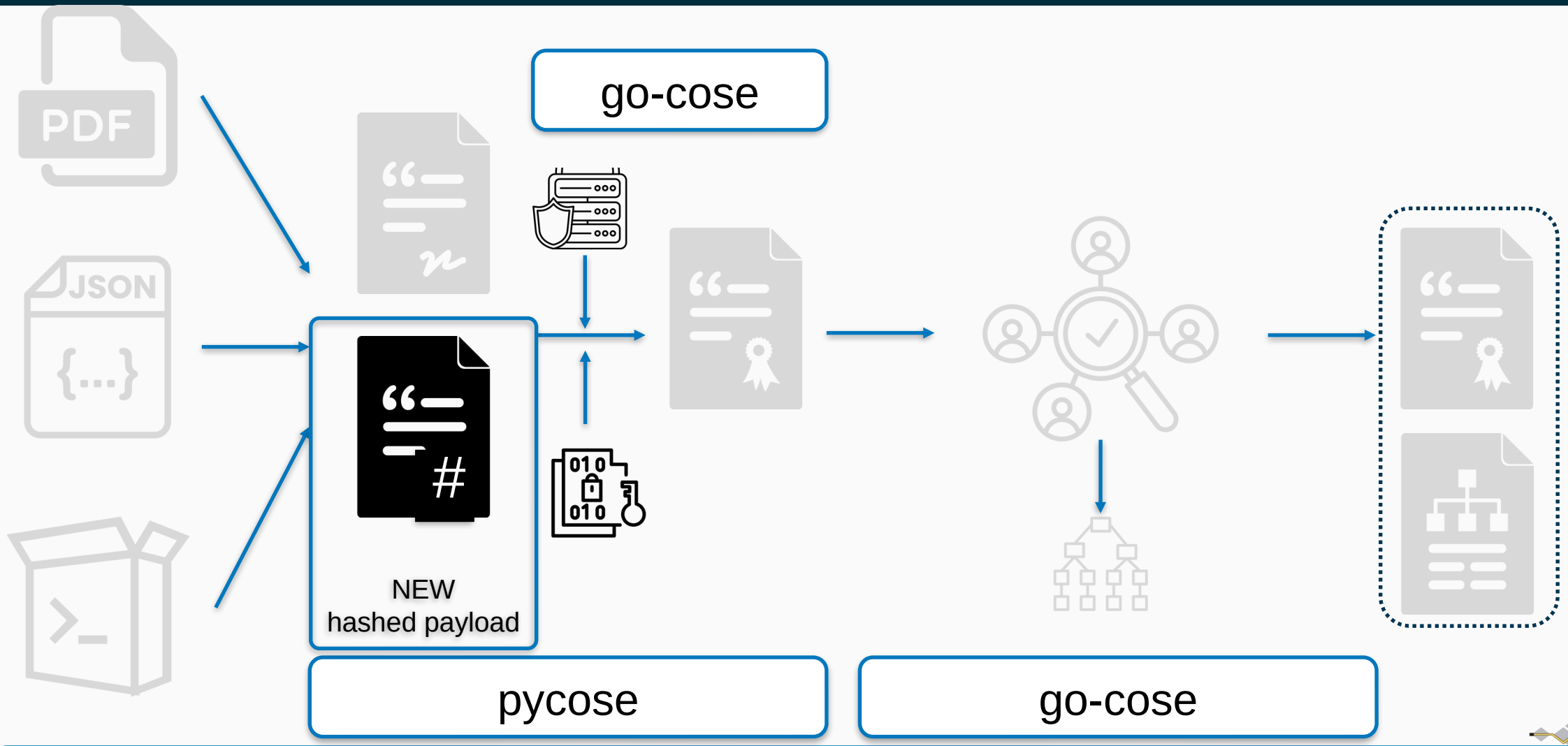
Hacking the Issues

End-to-end code validation of Issuer role workflow

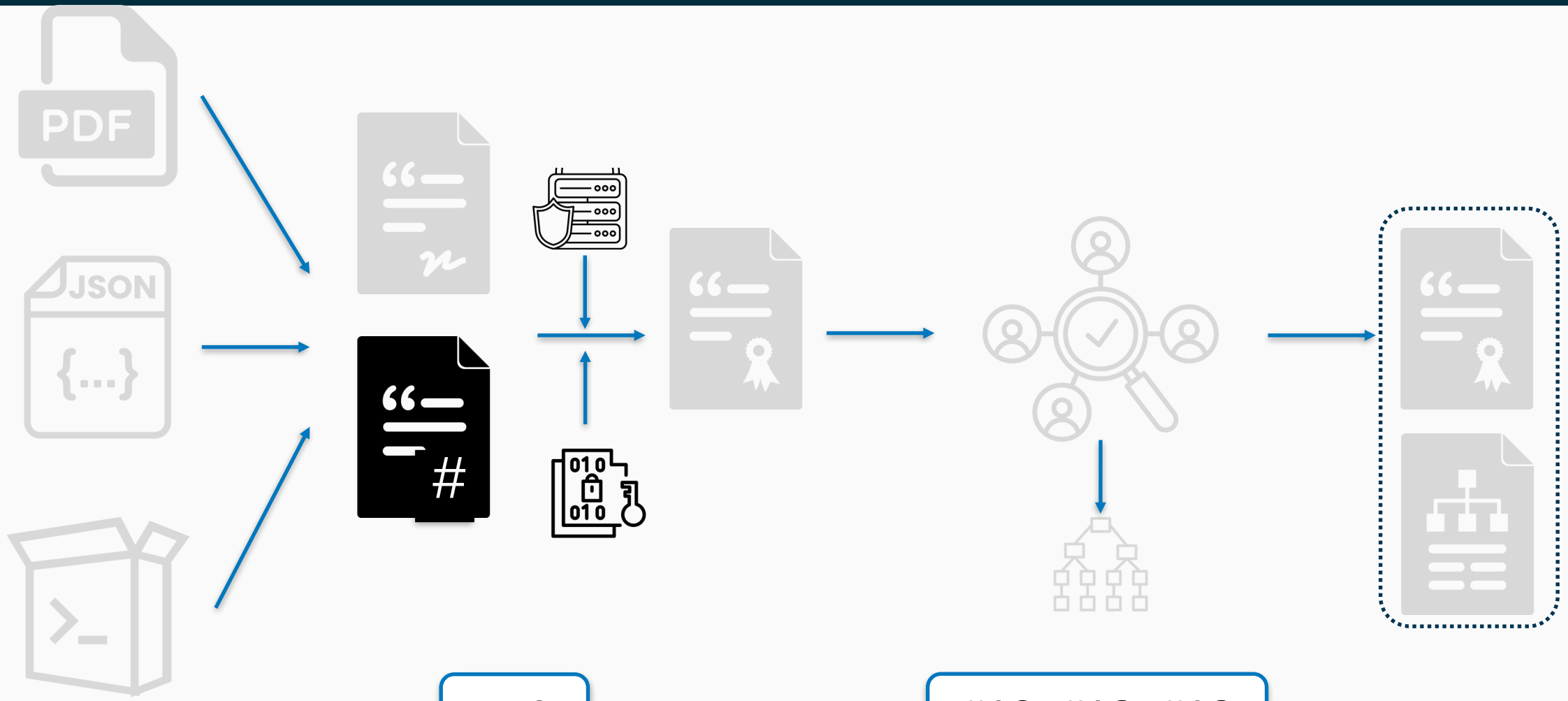


Artifact Payload Signed Statement Transparent Statement

End-to-end code validation of Issuer role workflow



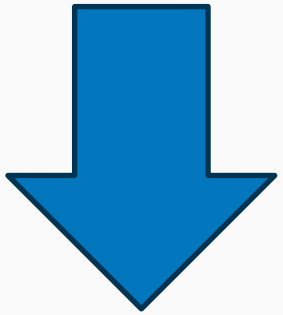
End-to-end code validation of Issuer role workflow



#10

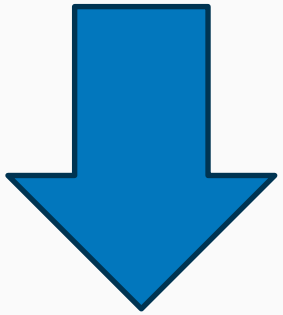
#10, #12, #13

```
{
    "author": "fred",
    "title": "my biography",
    "reviews": "mixed"
}
```



```
sha2 payload.json

SHA-256 (payload.json) =
fabacbdde60208761d086caa5930318259089b353150d480672c1c8e84100c85
```

```
{
    "author": "fred",
    "title": "my biography",
    "reviews": "mixed"
}
```



```
sha2 payload.json

SHA-256 (payload.json) =
fabacbdde60208761d086caa5930318259089b353150d480672c1c8e84100c85
```

cbor decoded signed statement:

```
protected headers:
{<class 'pycose.headers.Algorithm'>: <class 'pycose.algorithms.Es256'>,
 <class 'pycose.headers.ContentType'>: 'application/hashed+cose',
 <class 'pycose.headers.KID'>: b'testkey',
 13: {1: 'sample.synsation.io',
      2: 'my-product-id',
      8: {1: {-3: b".0=\x01\xbe'D\xe9^\x03\xdd\xea\xf6\xfe\x88\xecF\xef\xbc\xfd"
                  b'\xc4\xab\xfb\x822t\x810\xa6\xa3V\xdc',
              -2: b'\xc1\xa2\x87\x1a\xb4;\xde\x88\xa4\xab\xa9\x9b\x05\xb8\x92l'
                  b'a<\xfdW}S\xd2\xcaHO\xdb\x1b5B\x81\x01',
              -1: 1,
              1: 2}}},
-6800: -16,
-6801: 'https://storage.example/my-product-id’
-6802 application/spdx+json}
```

```
unprotected headers:
{}
```

```
payload:  b'\xfa\xba\xcb\xdd\xe6\x02\x08v\x1d\x08l\xaaY01\x82Y\x08\x9b51P\xd4\x80g,\x1c\x8e\x84\x10\x0c\
x85'
payload hex:  fabacbdde60208761d086caa5930318259089b353150d480672c1c8e84100c85
```

cbor decoded transparent statement:

```
protected headers:
{<class 'pycose.headers.Algorithm'>: <class 'pycose.algorithms.Es256'>,
 <class 'pycose.headers.ContentType'>: 'application/json',
 <class 'pycose.headers.KID'>: b'testkey',
 13: {1: 'sample.synsation.io',
      2: 'my-product-id',
      8: {1: {-3: b".0=\x01\xbe'D\xe9^\x03\xdd\xea\xf6\xfe\x88\xecF\xef\xbc\xfd"
                  b'\xc4\xab\xfb\x822t\x810\xa6\xa3V\xdc',
              -2: b'\xc1\xa2\x87\x1a\xb4;\xde\x88\xa4\xab\xa9\x9b\x05\xb8\x92l'
                  b'a<\xfdW}S\xd2\xcaHO\xdb\x1b5B\x81\x01',
              -1: 1,
              1: 2}}},
-6800: -16,
-6801: 'https://storage.example/my-product-id'
-6802 application/spdx+json}
```

```
unprotected headers:
{'receipts': [b'\xd2\x84Xs\xa4\x018"\x04X6scitt-counter-signing/d8b29e6d0e7e437'
              b'581deae29d037390a\x19\x01\x86\x00\x19\x01\x87x-did:web:app.da'
              b'tatrails.ai:archivist:v1:didweb\xa0@X`\xda\xc1$\x92 8M7R'
              b' T\xb2\xeb\x89\nX\xb9\x93U\x85\xb1*\xca\xbe(\xd2HY\xee'
              b'\xf3H\x1e:\xb7b\x80K\x00\xbfF%&Q\x8b!v\xc8tB\x8c\x87\tj'
              b'\x19\x02\xba\x1b\x87G\x90\x18$1\xd4\xce84\xfb\xa9\t\xc2%k'
              b'i\xe2\xfa3\xe9\x1dV\xff[t!\xc1i5\xc94\xce\xcf\xca\x85'
              b'\xe9\xfc\xd1']}
```

```
payload:  b'\xfa\xba\xcb\xdd\xe6\x02\x08v\x1d\x08l\xaaY01\x82Y\x08\x9b51P\xd4\x80g,\x1c\x8e\x84\x10\x0c\
x85'
payload hex:  fabacbdde60208761d086caa5930318259089b353150d480672c1c8e84100c85
```

SCRAPI: Relying Party Issues: Resolvers

One main thing(?) to overcome

Several 'resolve' APIs. Do we have what we need?

```
urn:ietf:params:scitt:signed-statement:sha-
256:base64url:5i6UeRzg1...qnGmr1o
```

2.2.2. Resolve Statement

Authentication **SHOULD** be implemented for this endpoint.

This endpoint enables Transparency Service APIs to act like Artifact Repositories, and serve `payload` values directly, instead of indirectly through Receipts.

Request:

```
GET /statements/urn...qnGmr1o HTTP/1.1
Host: transparency.example
Accept: application/pdf
```

2.2.3. Resolve Signed Statement

Authentication **SHOULD** be implemented for this endpoint.

This endpoint enables Transparency Service APIs to act like Artifact Repositories, and serve Signed Statements directly, instead of indirectly through Receipts.

Request:

```
GET /signed-statements/urn...qnGmr1o HTTP/1.1
Host: transparency.example
Accept: application/cose
```

2.2.4. Resolve Receipt

Authentication **SHOULD** be implemented for this endpoint.

Request:

```
GET /receipts/urn...qnGmr1o HTTP/1.1
Host: transparency.example
Accept: application/cose
```

Response:

If the Signed Statement requested is already included in the Append-Only Log:

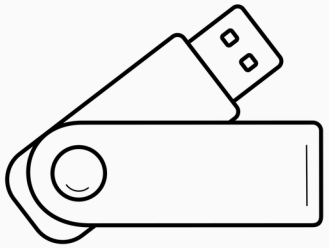
```
HTTP/1.1 200 Ok
Location: https://transparency.example/receipts/urn...qnGmr1o
Content-Type: application/cose

Payload (in CBOR diagnostic notation)

18([                              / COSE Sign1         /
  h'a1013822',                    / Protected Header   /
  {},                             / Unprotected Header /
  null,                           / Detached Payload   /
  h'269cd68f4211dffc...0dcb29c' / Signature          /
])
```

IETF

Several 'resolve' APIs. Do we have what we need?

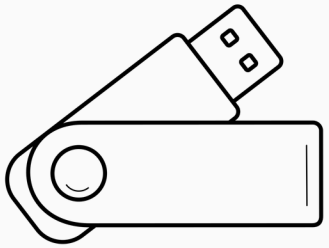
2 main questions:

- Do these APIs work?
  - Are the identifiers sensible?
- Do they solve the problem?
  - What does a reasonable relying party have in their hands when they want to check a software supply chain artifact?
  - What do they want to check?



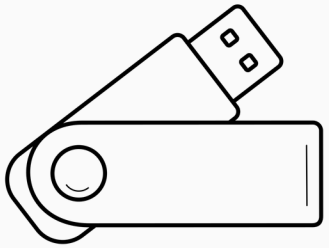
Several 'resolve' APIs. Do we have what we need?

If I find an artifact under a rock and want to see if my chosen Transparency Service knows anything about it, I can construct the URN from the file and call the locator to see if this exact file has any Statements.



Several 'resolve' APIs. Do we have what we need?

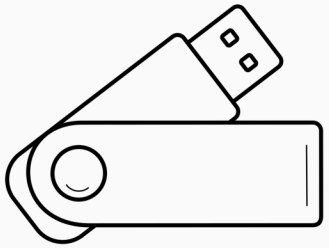
If I find an artifact under a rock and want to see if my chosen Transparency Service knows anything about it, I can construct the URN from the file and call the locator to see if this exact file has any Statements.



What I cannot do is locate associated Statements, like VEXs or SLSA documents or similar, because the hash is for the *payload*, not the *subject*.

Conclusions/suggestions

- Since the URN is trivially derived from the artifact, is it necessary for the TS to return it at all? Could a different identifier with different qualities be more useful?
- Since supply chain decisions typically require more than one input, I need a way to find associated statements. Thoughts:
  - Resolve by subject
  - Resolve by issuer
  - Resolve by arbitrary header element



Suggestion: constrain to issuer and subject for now

WHY?

- Maintains close connection to architecture and key interoperability points
- Avoids making additional implied requirements over the 'auxilliary services'
- Improves interoperability

WHAT WOULD IT RETURN?

- A paged list
- Of Transparent Statements

2.2.x Locate Statements
Authentication SHOULD be implemented for this endpoint

Request:

```
GET /transparent-statements/?params
HOST: transparency.example
ACCEPT: application/cose
```

Params:

Issuer: OPTIONAL. Limit to all Transparent Statements for Statements with this issuer in its CWT Claims header

Subject: OPTIONAL. Limit to all Transparent Statements for Statements with this subject in its CWT Claims header

IETF

Hackathon Report

Jon / Orie

Hackathon Plan

- Multiple sub-teams working on SCITT Architecture (draft-07) and SCRAPI (draft-01)
- Address open Issues on SCRAPI and use to double-check architecture
- Practical focus: is it reasonable for non-SCITT people to write code that works in the real world?
- Prove it with real world applications: remote signing and vCon

47

What got done

- Lots!
  - 25ish PRs on the Architecture doc! https://github.com/ietf-wg-scitt/draft-ietf-scitt-architecture/pulls
  - End-to-end client code for SCITT Issuer role https://github.com/datatrails/datatrails-scitt-samples
  - Use case application (Recover better from July 19th outage) https://github.com/OR13/ietf-120-scitt-cloudstrike-example/tree/main
  - Also very exciting VCONs collaboration which was presented separately

48

What we learned

- Lots!
  - Outstanding questions regarding search/locate interface much clearer
  - Roles and definitions much clearer
  - More care/QA needed in COSE encodings
  - IT IS POSSIBLE for a regular person to include SCITT in regular supply chain/production tooling!

49

IETF 120 Hackathon Project: Incident Response

Objective: Show how SCITT and Transparency Services could be used by Computer Security Incident Response (CSIRT) in incidents similiar to July 19th, in the future.

🚧 I have not tested any of this.

🚧 This is just an example, developed in the IETF 120 Hackathon.

See the fixer script here for more details: https://github.com/alexdelifer/crowdstrike-fixer.

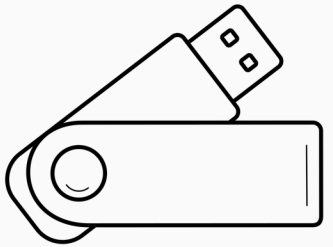
CSIRT team arrives at a data center impacted by the incident.

The building security team wants to confirm the CSIRT team is in possession of recovery media that is authorized for use in this data center.

The building security team checks that the bootable media and fixer media are signed by certificates that are trusted by the CSIRT team.

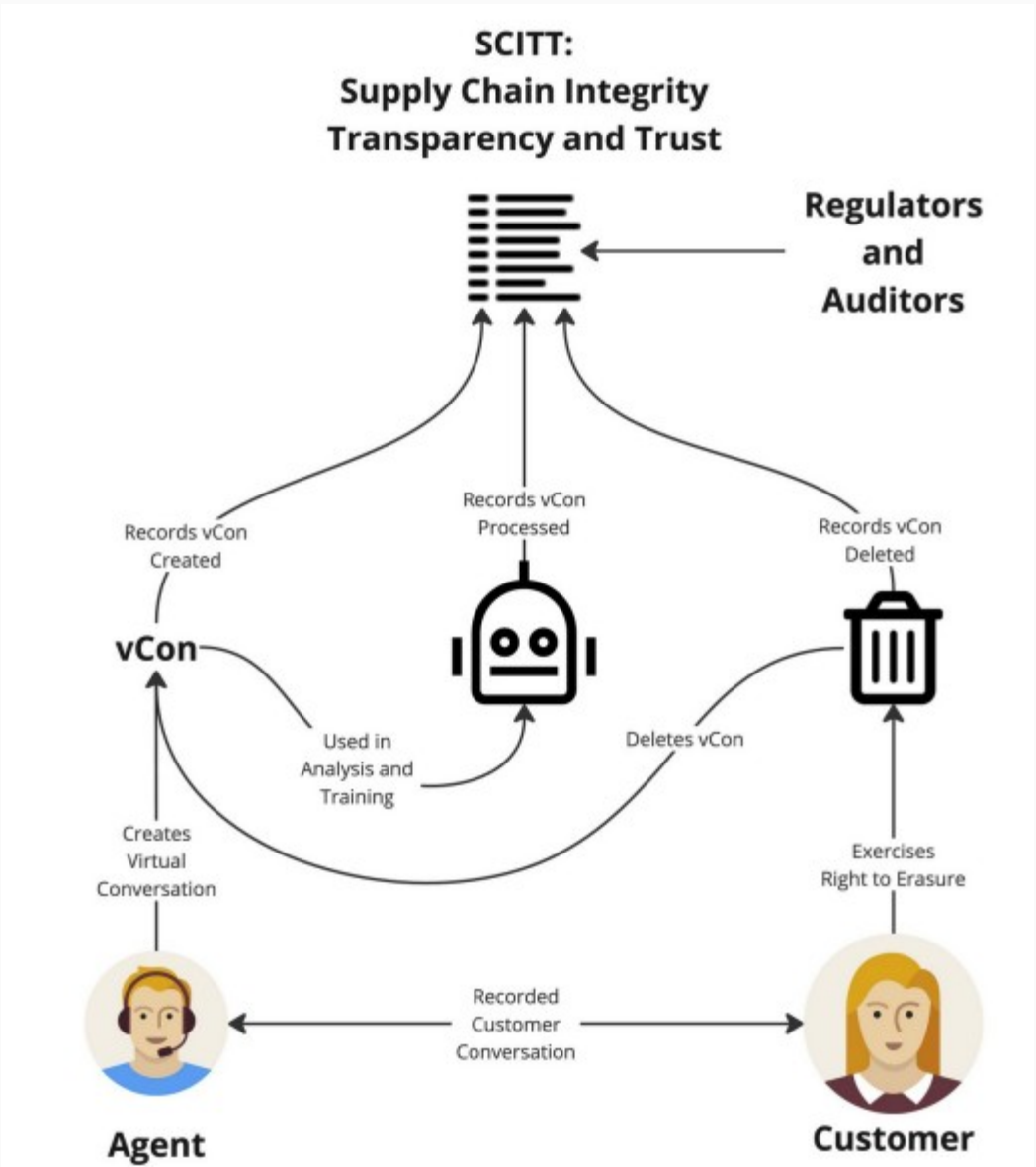
Next the building security check that the signed recovery script are notarized in the data center notary.

The following bash script example sketches some of the operations that the building security team might perform in order to establish that the incident response team is in possession of authorized recovery tools.



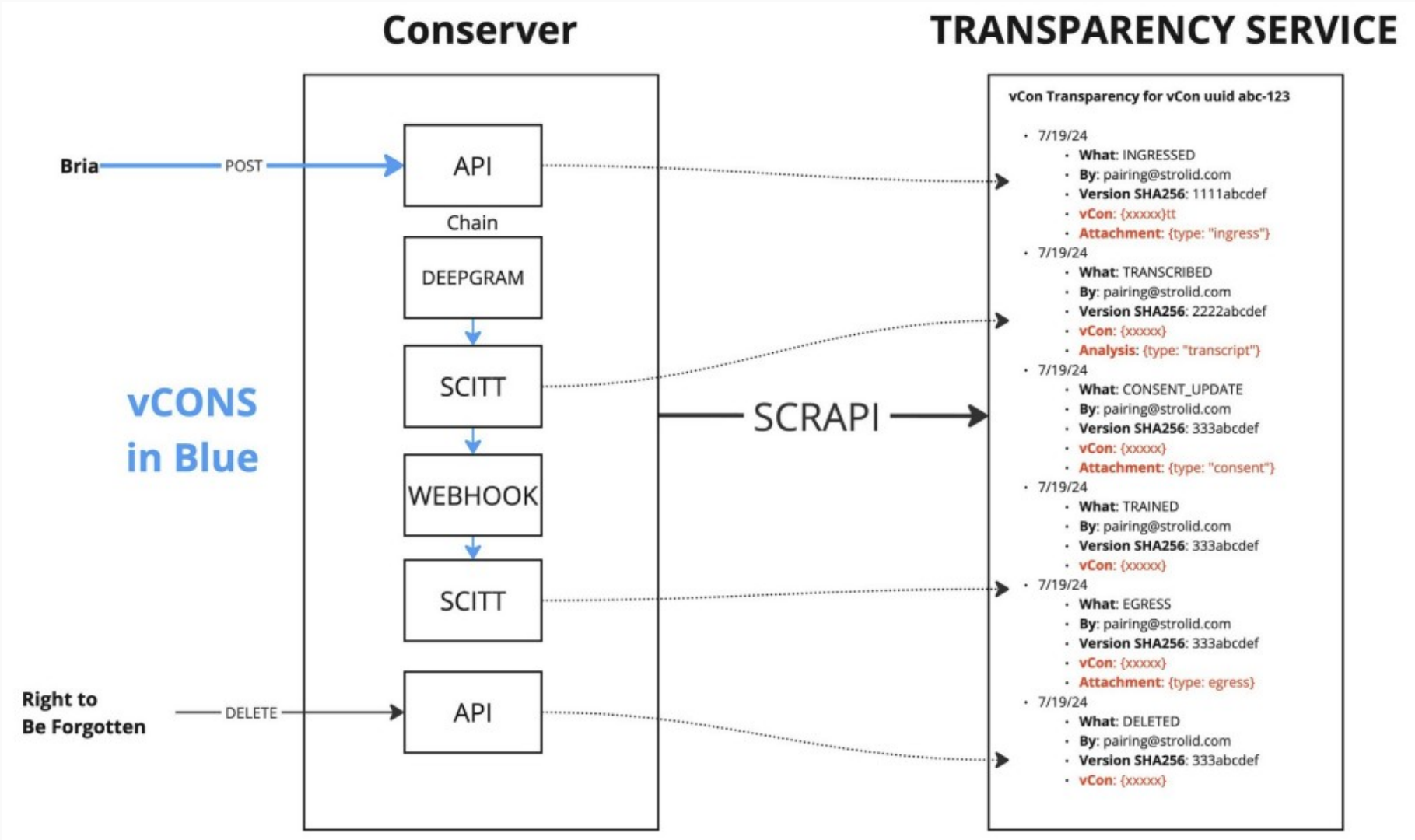
```
# We assume the bulding security team has certificates for the CSIRT and Data Center Notary,
# and that scitt-verify script has access to these certificates

# download the script
curl -fLs https://raw.githubusercontent.com/alexdelifer/crowdstrike-fixer/main/autorun > aut

# hash the script
sha256sum autorun
# 80a4cce35fbd953a497b551756d59ea7f071b77cb39a044f4ff0ff08af1aa19e  autorun

# Then the building security team retrieves the executables from the recover media
# And they confirm the hashes and signatures are matching and acceptable...

# verify the transparency of the recovery script
# scitt-verify ./autorun.transparency.cbor ./autorun

# Data Center Notary Receipt Signature Verified with Certificate Thumbprint:
# ✅ 52bdc3ed42f3a233e36312d85670625003df560817dfd3fcde071471ca309149

# CSIRT Signature Verified with Certificate Thumbprint:
# ✅ 72137c02c2d762edb2ed88f225a605404093df02185eaf7640ed25c1a423c3c0

# Payload Verified:
# ✅ 80a4cce35fbd953a497b551756d59ea7f071b77cb39a044f4ff0ff08af1aa19e
```

SCITT enabled vCons



Humans have the right to know how their data was processed. Especially when there are robots involved

SCITT enabled vCons



The Team (not quite all of it...)

Next Steps

Chairs

SCITT Drafts

- Software Supply Chain Uses Cases
  https://datatracker.ietf.org/doc/draft-ietf-scitt-software-use-cases/
- SCITT Architecture
  https://datatracker.ietf.org/doc/draft-ietf-scitt-architecture
  - Ask: Working Group Last Call
- SCITT Reference API (SCRAPI)
  https://datatracker.ietf.org/doc/draft-ietf-scitt-scrapi/

AOB (Open Mic)

Wrap-Up

Thank you