COSE Hash Envelope

draft-steele-cose-hash-envelope

Current State - Attached
To Be Signed Bytes

COSE_SIGN1 (Envelope)
- protected : bstr . cbor Protected_Header
- payload : bstr

Protected_Header
- ? &(content_type: 3) => tstr / uint
- * int => any

Issuer
- Payload
- manifest.spdx.json

application/spdx+json
Signed Bytes

**Cose_Sign1 (Envelope)**
- protected : bstr .cbor Protected_Header
- payload : bstr
- signature : bstr

Statement

Issuer
Registering on a Verifiable Data Structure (VDS)

**COSE_SIGN1 (Envelope)**
- **protected**: `bstr .cbor Protected_Header`
- **payload**: `bstr`
- **signature**: `bstr`
- **unprotected**: `Unprotected_Header`

**How large is the COSE_Sign1 Envelope?**
- Protected Header: \(~1k\)
- Unprotected Header: \(0\)
- Signature: \(~1k\)

- Is `size` the constraint?
- Is the **payload** already stored somewhere else?
- Do we need to continually pass the payload for signature checking?
Registering on a Verifiable Data Structure (VDS)

**COSE_SIGN1 (Envelope)**
- **protected**: `bstr .cbor Protected_Header`
- **payload**: `bstr`
- **signature**: `bstr`
- **unprotected**: `Unprotected_Header`

**How large is the COSE_Sign1 Envelope?**
- Protected Header: ~1k
- Unprotected Header: 0
- Signature: ~1k
- Payload: 1k-50gb

➢ Is **size** the constraint?
➢ Is the **payload** already stored somewhere else?
➢ Do we need to continually pass the payload for signature checking?
COSE Detached Payloads
Detached Payloads

**COSE_SIGN1 (Envelope)**
- **protected**: `bstr .cbor Protected_Header`
- **payload**: `nil`
- **signature**: `bstr`
- **unprotected**: `Unprotected_Header`

**Unprotected_Header**
- `? &payload_location => tstr "https://sbom.sh/retrieve/45c86..."
  * int => any`
Detached Payloads

**Unprotected_Header**

? &payload_location => tstr "https://sbom.sh/retrieve/45c86..."

* int => any

**COSE_SIGN1 (Envelope)**

protected : bstr . cbor Protected_Header

payload : nil

signature : bstr

unprotected : Unprotected_Header

**VDS**

Transparency Service

Verify

External Storage
Detached Payloads

COSE_SIGN1 (Envelope)
- protected: bstr.cbor Protected_Header
- payload: nil
- signature: bstr
- unprotected: Unprotected_Header

Unprotected_Header
- ? &payload_location => tstr "https://sbom.sh/retrieve/45c86..."
- * int => any

VDS
- Transparency Service
- IF NOT accessible

Verify
Content of a Payload

- Inline content (binary)
- Small File
- Large file
- Collections of files: large and/or small, likely packaged in another file (zip/tar) or referenced by a manifest
- File by Reference: URI to the location: docker image, npm package, vcon, youtube video
- Manifest: Collections of files, each referenced by a unique id (eg: docker image, npm package, vcon, youtube video)

Persistence

Where is the Payload persisted?

- VDS
- Transparency Service
- External Storage

COSE_SIGN1 (Envelope)

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

Verify
HASH Envelope
Options Considered, → Proposal
### Content of a Payload

- **Inline content (binary)**
- **Small File**
- **Large file**
- **Collections of files**
  - Large and/or small
  - Likely packaged in another file (zip/tar) or referenced by a manifest
- **File by Reference:** URI to the location: docker image, npm package, vcon, youtube video
- **Manifest:** Collections of files, each referenced by a unique id (e.g., docker image, npm package, vcon, youtube video)

### Payload Encasing

**Inline:**
- `payload`: `<statement>`
- `content-type`: Type of the payload
  - `(application/json, application/bin,)`

**Hash:**
- `type`: `applicationhashed+cose`
- `payload`: Hash of the content, minimizing the COSE Envelope Size
- `payload pre-image`
- `content-type`: Type of the hashed content
  - `(application/spdx+json)`
- `detached-hash-algorityhm`: sha-256 | SHA3-512
- `payload-location`: added to resolve a possible location for the payload

**Detached Payload:**
- `payload`: `nil`
- `content-type`: the type of the detached content
  - `(application/json, application/bin,)`
- `payload-location`: added to resolve a possible location for the payload

### Persistence

**VDS Transparency Service**

**External Storage**

**Legend**

- Most relevant
- Possible
- Least relevant
- Least likely

**Verify**
Content of a Payload

- **Inline content** (simple values)
- **Small File**
- **Large file**
- **Collections of files**
  - Large and/or small
  - Likely packaged in another file (zip/tar) or referenced by a manifest
- **File by Reference:** URI to the location: docker image, npm package, vcon, youtube video
- **Manifest:** Collections of files, each referenced by a unique id (eg: docker image, npm package, vcon, youtube video)

Payload Encasing

**Hash:**
- **type:** application/hashed+cose
- **payload:** Hash of the content, minimizing the COSE Envelope Size
- **payload pre-image content-type:** Type of the hashed content (application/spdx+json)
- **detached-hash-algorithm:** sha-256
- **payload-location:** added to resolve a possible location for the payload

Persistence

- **VDS**
  - Transparency Service
- **External Storage**
  - Never wonder what size constraint will fail
  - Builds upon existing storage services

Legend

- Most relevant
- Possible
- Least relevant
- Least likely
HASH Envelope Protected Header

{
  1: -7, / Algorithm (ECDSA) /
  16: application/hashed+cose, / Type (RFC 9596) /
  TBD_1: -16, / Payload Hash Algorithm (SHA256) /
  TBD_2: application/spdx+json, / Payload Pre-Image Content Type /
  TBD_3: https://blob.example/24f...9c9, / Payload Location /
}

Fetching a resource can be negotiated by the content type (TBD_2)
Example: SPDX supports json, yaml, xml, …
Reference Validations & Implementations

- SCITT GitHub Actions
  - github.com/datatrails/scitt-action
  - github.com/digicert/scitt-action

- vCon
  - github.com/vcon-dev/vcon-server
Next Steps

- Please review the draft
- Is there enough interest for WG Adoption