Oauth (Token) Status List

A simple and scalable credential revocation/status mechanism
[Formerly known as JWT CWT Status List]
A Refresher - The Problem

How to enable the issuer of a token (e.g. CWT or JWT) to communicate dynamic status information about a token after it is issued and before it expires.

Example - An SD-JWT Verifiable Credential where the Issuer would like to communicate whether the credential is revoked or not.
Key Requirements

- Scalable: Must scale to millions (100’s millions) of credentials
- Issuer Herd Privacy: Able to protect Relying Parties and Holders/Users from Issuer knowing where a given token is being verified/used
- Work with common formats: Support JOSE/COSE based tokens/credentials, i.e. can be used natively for ISO mdoc and IETF SD-JWT-VC
- Caching Support: Enable verifying parties to cache status lists for offline verification
Proposed Solution

- Byte array based status list (for large amounts of credentials)
- Status is indicated by the value of a specific index in the status list
- Status List is Gzip-compressed and the outcome base64 encoded
- Signed and delivered as JWT/CWT
Example: Referenced Token

```json
{
  "alg": "ES256",
  "kid": "11"
}
...
//other claims
"status": {
  "uri": "https://example.com/statuslists/1",
  "idx": 5
}
}
```
Example: Status List JWT

```json
{
  "alg": "ES256",
  "kid": "12",
  "typ": "statuslist+jwt"
}

{
  "exp": 1687517770,
  "iat": 1686912970,
  "iss": "https://example.com",
  "status_list": {
    "bits": 1,
    "lst": "H4sIAMo_jGQC_zvp8hMAZLRLMQMAAAAg"
  },
  "sub": "https://example.com/statuslists/1"
}
```
Example: How it fits together

```
"status": {
  "idx": 5,
  "uri": "https://example.com/statuslists/1",
}

"sub": "https://example.com/statuslists/1"
"status_list": {
  "bits": 1,
  "lst": "H4sIAMo_jGQC_zvp8hMAZLRLMQMAAAA"
}
```

0x0 = VALID
0x1 = INVALID

100101000100

Deflate gzip
Further Features

- Status Type can be extended to represent more than 1 bit, i.e. “valid”/“invalid”
  - e.g. for suspension
  - Status Types are defined by the specification, extensible by IANA registry
- Fetching protocol over HTTP GET
  - Additional caching guidance by the Status List Provider by using HTTP Cache Control
  - Using Media Types (e.g. application/status-list+jwt) to differentiate between status list formats
JWT Status List Example sizes

- Average revocation rate on the web: 1.2%
- Average Status List size: depends on several factors
  - Number of entities managed by the Issuer
  - Usage of batch credential issuance
  - Usage of decoy entries
- These sizes can be reduced by additional HTTP compression due to base64 encoding (~25%)

<table>
<thead>
<tr>
<th>List Size (total number of entries)</th>
<th>0.1% revoked</th>
<th>1% revoked</th>
<th>2% revoked</th>
<th>5% revoked</th>
<th>10% revoked</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.000</td>
<td>433 bytes</td>
<td>660 bytes</td>
<td>868 bytes</td>
<td>1.258 bytes</td>
<td>1.717 bytes</td>
</tr>
<tr>
<td>100.000</td>
<td>806 bytes</td>
<td>2.913 bytes</td>
<td>4.796 bytes</td>
<td>9.616 bytes</td>
<td>12.908 bytes</td>
</tr>
<tr>
<td>1.000.000</td>
<td>4.241 bytes</td>
<td>25.302 bytes</td>
<td>42.550 bytes</td>
<td>80.441 bytes</td>
<td>123.185 bytes</td>
</tr>
<tr>
<td>10.000.000</td>
<td>39.146 bytes</td>
<td>246.938 bytes</td>
<td>417.993 bytes</td>
<td>794.874 bytes</td>
<td>1.225.229 bytes</td>
</tr>
</tbody>
</table>
Progress Update

- Working Group Adoption of draft
- Changed draft title
- Defined the HTTP protocol for status list retrieval
- IANA registrations for Media Types and JWT claims
- Privacy Considerations
- Updated Terminology Verifier -> Relying Party
- Gathered some early implementation detail on the approaches performance from a representation size efficiency perspective
Work in Progress

- Option for unsigned Status List over HTTP endpoint
- Switching compression to Zlib (suited better, no dynamic headers)
- Discussion on the Draft Title
  - OAuth Status List (current)
  - OAuth Token Status List
  - Token Status List
  - Bitarray Status List
- Design considerations for introduction
- CWT representations
- Security and implementation considerations
- Testing the current specification with implementations
- Discussion on more privacy-preserving options
- Comparison to/Lessons learned from existing revocation approaches
Questions?
Links

- Git Repository -> https://github.com/vcstuff/draft-looker-oauth-jwt-cwt-status-list
  - Please use Github Issues for feedback
Backup
Security Considerations

- Correct decoding, parsing and validation of the encoded status list: risk to fetch erroneous status data
  - Easy to implement algorithms
  - Test vectors for implementers
- Cached and stale status lists, Verifier should be aware if they fetch the up-to-date data
  - Status List contains expiration date
  - HTTP caching mechanisms used in the retrieval protocol (next version)
- Status list only provides the up-to date/latest status, no historical data
  - May be provided by the underlying hosting architecture with additional API if necessary
  - Historical information is not necessary for most use-cases
Privacy Considerations

- **Herd Privacy**
  - Privacy depends on the size of the status list
  - More entities means better herd privacy but larger file size and worse scalability

- **Profiling/Tracking: Verifiers may regularly fetch the status list to create a profile**
  - Less number of Status Types prevents additional information leakage
  - Reissue/refresh tokens regularly

- **Malicious Issuers: issuers may generate unique status lists per credential**
  - Theoretically possible, observable by Verifiers through metadata
Implementation/Privacy Considerations

- **Correlation Risks**
  - Issuers should avoid using sequential indices, instead use randomized indices over multiple status lists
  - Issuers are recommended to use decoy/dead entries that are never assigned and other obfuscation mechanisms
  - Issuers using batch credential issuance should use individual indices per credential
    - Batch revocation might reveal some correlation of presented credentials
- **Third Party Hosting/CDN**
  - Improves availability and scalability as Status List can be provided by third parties
  - Privacy may be increased if hosting of the status list is done by a third party instead of the issuer as it reduces tracking possibilities for the issuer but adds another party
Other approaches?

- Accumulator/ZKP-based approaches
- OCSP/Validity credentials
- X.509 Certificate Revocation Lists
Accumulator/ZKP-based approaches

- Revocation scheme based on cryptographic accumulators (usually RSA or EC)
- Provides the best privacy properties (no tracking, one time proof of non-revocation)
- Has a bad scalability
  - Hyperledger Indy revocation registries were capped to 32768 entities
- Requires additional effort for the wallet
  - Fetch accumulator and delta updates from the registry
  - Complicated cryptographic computation (witness update) to perform proof to the Relying Party
- Not standardized
- Some of the better scaling variants are based on pairing-based cryptography
  - Not well tested, not ready for production

→ This approach offers great potential for privacy but is still technically immature
OCSP Stapling/Validity credentials

- RFC 2560/6960 - ASN.1-based status information is fetched by the Holder from the Issuer directly and “stapled” to the credential
- OCSP Stapling/Validity credentials reveal usage information directly to the Issuer
  - Loss of privacy towards the issuer
  - More privacy towards Relying Party as they are not able to re-check the status
- Has significant challenges for scalability
  - Overall system complexity scales with the number of holders → more Holders than Relying Parties expected
  - Validity Responses by the Issuer must be computed dynamically → high cost
- Requires less strict freshness to scale better (holders don’t have to re-request status too often)
  - Relying Parties cannot directly communicate their requirements for freshness
- Very little existing work how this concept would apply to the VC ecosystem (validity credentials)

→ This approach is doable but adds system complexity for Issuers and Holders and requires further adoption to VCs
X.509 Certificate Revocation Lists

- RFC 5280 - ASN.1-based CRL for X.509 certificates
- In production, but has scalability issues
  - This is why browsers are using curated CRLSets/Bloom filters
- Similar privacy attributes as status list (also provides herd privacy for lookups)
- Supports historic data
- No good technological fit to formats chosen for PID/EAA

→ This approach is similar to JWT/CWT Status List but conveys more information resulting in larger payloads
## Comparison between Status List and CRL

<table>
<thead>
<tr>
<th></th>
<th>IETF JWT/CWT Status List</th>
<th>IETF CRL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technological fit</td>
<td>SD-JWT / mdoc (JSON/CBOR)</td>
<td>X.509 (ASN.1)</td>
</tr>
<tr>
<td>size</td>
<td>grows with revocation rate</td>
<td>grows with revocation rate and time</td>
</tr>
<tr>
<td>data</td>
<td>only includes up-to-date data</td>
<td>includes up-to-date and historic data</td>
</tr>
<tr>
<td>Data representation</td>
<td>Gzip-compressed byte array</td>
<td>ASN.1-Sequence containing Serial number and timestamp</td>
</tr>
<tr>
<td>Example size for n=100,000 p=0.01</td>
<td>2,9 kB (compressible by ~25%)</td>
<td>35 kB (compressible by ~35%)</td>
</tr>
</tbody>
</table>