RATS Agenda – Thursday, July 29\textsuperscript{th} – Session I

Room 8, RATS Session 2  
Time zone: PDT (UTC-7)

12:00 : 12:05 **Agenda bash & logistics**  
(5 min) Nancy Cam-Winget, Kathleen Moriarty, Ned Smith

12:05 : 12:10 **Open Mic**  
(5 min)

12:10 : 12:35 **Attestation Results, Trusted Path Routing**  
(25 min) Eric Voit  
draft-voit-rats-attestation-results, draft-voit-rats-trustworthy-path-routing

12:35 : 12:40 **Attestation Event Stream Subscription**  
(5 min) Henk Birkholz, Eric Voit  
draft-birkholz-rats-network-device-subscription

12:40 – 13:00 **Trusted Identities**  
(20 min) Meiling Chen

13:00 – 13:30 **Break**  
(30 min)
Attestation Results and Trusted Path Routing

- Eric Voit
Summary

• Contents
  • Object definitions for Attestation Results (AR) to support Secure Interactions between Attester and Relying Party
  • How the Attester can augment AR to improve scale and speed of appraisal
  • State Machine for the Appraisal Policy for Attestation Results

• Two implementations
  • Trusted Path Routing (Proprietary – Cisco)
  • Veraison (Open Source – Confidential Compute Consortium)

• Ask: WG Adoption
Remote Attestation in a Heterogenous World

- Many types of Attesting Environments (AE)
- What may be trusted by Relying Party
  - Identity: Hardware type, software build, developer, ...
  - Verifier Appraisals: Sw integrity, config ok, attester recognized, ...
  - Freshness: Nonce, trusted timestamp, ...

  Support varies by AE chip type > Attester > Verifier

- Relying Party cannot support \( \infty \) language permutations
  - And a mix and match across L1 \( \leftrightarrow \) L7 platforms is coming if IETF RATS succeeds

- Need: Shared definitions/structures for Verifier Appraisals coming to Relying Party
  - Will help scale and Interop
  - Reduce transcoding/mapping between sequentially bound sets of Attesters
  - Could be encoded in EAT, YANG, CDDL, etc...
Verifier Appraisal

- Periodic appraisal and generation of Attestation Results
- One to Many Trustworthiness Claims assigned during an appraisal cycle
- Attestation Results signed and returned to Attester (for scale/speed)
## Normalizing Trustworthiness Claims

<table>
<thead>
<tr>
<th>Trustworthiness Claim</th>
<th>Confidential Compute</th>
<th>Attesting Environments</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Process-based (SGX, TrustZone)</td>
<td>VM-based (SEV, TDX, ACCA)</td>
</tr>
<tr>
<td>ae-instance-recognized</td>
<td>Optional</td>
<td>Optional</td>
</tr>
<tr>
<td>ae-instance-unknown</td>
<td>Optional</td>
<td>Optional</td>
</tr>
<tr>
<td>hw-authentic</td>
<td>Implicit</td>
<td>Chip dependent</td>
</tr>
<tr>
<td>hw-verification-fail</td>
<td>Implicit if not ok</td>
<td>Chip dependent</td>
</tr>
<tr>
<td>executables-verified</td>
<td>Optional</td>
<td>Optional</td>
</tr>
<tr>
<td>executables-refuted</td>
<td>Optional</td>
<td>Optional</td>
</tr>
<tr>
<td>file-system-anomaly</td>
<td>n/a</td>
<td>Optional</td>
</tr>
<tr>
<td>source-data-integrity</td>
<td>Optional</td>
<td>Optional</td>
</tr>
<tr>
<td>config-secure</td>
<td>Optional</td>
<td>Optional</td>
</tr>
<tr>
<td>config-insecure</td>
<td>Optional</td>
<td>Optional</td>
</tr>
<tr>
<td>target-isolation</td>
<td>Implicit</td>
<td>Implicit</td>
</tr>
<tr>
<td>runtime-confidential</td>
<td>Implicit</td>
<td>Implicit</td>
</tr>
<tr>
<td>secure-storage</td>
<td>Implicit</td>
<td>Chip dependent</td>
</tr>
</tbody>
</table>

Specific claim definitions, extensible

- affirming
- detracting
Normalized Trustworthiness Claims ≠ the same Relying Party policy disposition

• Even with Normalized Trustworthiness Claims, Attesters need not be treated equivalently by the Relying Party

  • Variance in underlying protections of SGX, TrustZone, SEV, TPM, etc. could mean different disposition via the Appraisal Policy for Attestation Results.

  • Each Verifier, or Verifier version, or Verifier appraisal of a specific type of Attester may be trusted differently for different claims
Trustworthiness Claim Delivery
Based on draft-ietf-rats-architecture: Passport Model
Attestation Results Augmented Evidence

- Input to Relying Party’s Appraisal Policy for Attestation Results
- How to review the AR-augmented evidence to ensure no tampering
Attestation Results Augmented Evidence

objects needing specification

<table>
<thead>
<tr>
<th>Trustworthiness Claims of the Verifier</th>
<th>Verified Identity instance(s)</th>
<th>Verifiable Freshness</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Identity</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Attesting Environment</td>
<td>chip vendor</td>
<td>Random Number</td>
</tr>
<tr>
<td>Hardware</td>
<td>chip type</td>
<td>nonce</td>
</tr>
<tr>
<td><strong>Integrity</strong></td>
<td></td>
<td>Synchronized Clocks</td>
</tr>
<tr>
<td>Files</td>
<td>target environment</td>
<td>timestamp</td>
</tr>
<tr>
<td><strong>Confidentiality</strong></td>
<td></td>
<td>Epoch</td>
</tr>
<tr>
<td>Config</td>
<td>target developer</td>
<td>tuda sync token</td>
</tr>
<tr>
<td>Target Environment</td>
<td>ae instance</td>
<td>epoch id</td>
</tr>
<tr>
<td>Data</td>
<td>verifier developer</td>
<td></td>
</tr>
<tr>
<td></td>
<td>verifier build</td>
<td></td>
</tr>
</tbody>
</table>

- Categories defined in this draft
- Specific objects to be defined in other drafts

**Defined in this draft**
- ae-instance-recognized
- ae-instance-unknown
- hw-authentic
- hw-verification-fail
- executables-verified
- executables-refuted
- file-system-anomaly
- source-data-integrity
- config-secure
- config-insecure
- target-isolation
- runtime-confidential
- secure-storage

**Defined in this draft**
- Categories defined in draft-ietf-rats-architecture Section 10
Current topics being worked by authors

• Categorizing ‘Trustworthiness Claims’ into ‘Endorsements’ and ‘Capabilities’?
• Datatype of ‘Trustworthiness Claims’: move from identities to enumerations?
• Follow-up drafts. E.g., Encoding in EAP for TLS transport
Summary

• Contents
  • Object definitions for Attestation Results (AR) to support Secure Interactions between Attester and Relying Party
  • How the Attester can augment AR to improve scale and speed of appraisal
  • State Machine for the Appraisal Policy for Attestation Results

• Two implementations
  • Trusted Path Routing (Proprietary – Cisco)
  • Veraison (Open Source – Confidential Compute Consortium)

• Ask: WG Adoption
Trusted Path Routing

draft-voit-rats-trustworthy-path-routing-03
IETF 111, July 29th 2021, RATS WG

Eric Voit
Cisco
evoit@cisco.com

Chennakesava Reddy Gaddam
Cisco
chgaddam@cisco.com

Guy Fedorkow
Juniper
gfedorkow@juniper.net

Henk Birkholz
Fraunhofer SIT
henk.birkholz@sit.fraunhofer.de
Trusted Path Routing

- Link adjacencies added to Trusted Topology based on latest Relying Party’s appraisal of AR Augmented Evidence

replicate through Existing Routing Protocol

Attested Topology

Relying Party

Global Routing

Attesting Policy for Attestation Results

must executables-verified

nonce

Evidence x

Attestation Results x+

Evidence x-

Attestation Results x+

Augmented Evidence y

Verifier

TPM

time(X)

time(Y)

Attester

© 2021 Cisco and/or its affiliates. All rights reserved.
Trusted Path Routing - Demo

- Custom topologies dynamically maintained based on Attestation Results
Changed since last draft version

- Extracted the elements to draft-voit-rats-attestation-results:
  - Trustworthiness Claims, Relying Party State Machine, Call Flow.
- Alignment of WGLC comments received on Charra YANG model
- Authorship updated
Next Steps

• Continued alignment with draft-voit-rats-attestation-results (e.g., Trustworthiness Claims structures)

• Definition of EAP payload (separate draft)

• No assertion to adopt until WG makes progress/adopts draft-voit-rats-attestation-results
Attestation Event Stream Subscription

- Henk Birkholz
- Eric Voit
Trusted Identities

- Meiling Chen
Use TEE Identification in EAP-TLS

draft-chen-rats-tee-identification-01

IETF111-2021-RATS

Meiling Chen /China Mobile
Objective

- Uses TEE and EAP-TLS to create a secure and trusted procedure to authenticate a device’s identity.
- Can be used in transport layer as identity authentication
- Can be used in link layer to determine if the access of network is permitted
Justifications

- RATs needs a mechanism to authenticate identity
- TLS protocol is secure, but the device that processes this protocol cannot be fully trusted
Architecture of TEE Identification use EAP-TLS

REE

TEE

Certificates

Key Derivation

Inner middle layer

EAP-TLS Client

EAP-TLS Server

IML: Key derivation
Response to EML about EAP-TLS encryption and decryption relevant message.

EML: Communicate with EAP-TLS Server
Request encryption and decryption relevant messages from IML.
enum{
    Random;
    keyshareExtension;
    PreSharedKeyExchange
    CertificateList
    CertificateVerify
    Finished
    NewSessionTicket
    ApplicationData
    Alert
}ParameterType

Struct{
    bool request;//true: request; false response. If it's request message, then th
    ParameterType type
    uint24 length
    select(type){
        case Random randomValue
        case KeyshareExtension keyshareextensionValue
        case PreSharedKeyExchange value;
        case CertificateList
        case CertificateVerify
        case Finished
        case NewSessionTicket
        case ApplicationData
        case Alert
    }
}MiddleLayerMessage
Information pre-stored in TEE

Certificate that complies with X509.3 or other. If using EAP-TLS as the authentication protocol, then the ID of the TEE enabled device is the certificate complies X509.3.

Key derivation process in TEE

Key derivation process must be executed in TEE.
Message 1: KeyShareExtension request from EML to IML.

Message 2: responses to message1 and returns the KeyShareExtension response to EML.

Message 3: includes plaintext ServerHello message and encrypted Server Params and Auth, also includes the entire handshake context which will be used to create CertificateVerify and Finished context.
● Message 4: encrypted TLS Client Certificate, TLS CertificateVerify and TLS Finished Message will be included.

● Message 5: encrypted application data 0x00 will be sent to IML to decode.

● Message 6: plaintext will be sent to EML. Then EML will make the determination if the authentication procedure is finished.
Other branches of EAP-TLS procedure

- Ticket Establishment: message 5
- Resumption: message 1 for request, message 2 for response
- Termination: message 4/6
- HelloRetry Request: plaintext from Server to Client

Message 1-6 also contains the branches of TLS procedure
Security Consideration

1. Exhaustive attack from REE
   prioritized problem need to be solved, one possible solution:
   use a counter or timer to limited the access frequency from REE to TEE

2. Deny of Service
   the integrity of encrypted message could be tampered by malicious REE or other parties.
ToDo

• Prevent or mitigate exhaustive attack from REE.
• How to identify if the device enables TEE function.
Thank You!
Room 8, RATS Session 3
Time zone: PDT (UTC-7)

13:00 – 13:30 Break
(30 min)

13:30 – 13:50 SUEID and EAT's relation to IDevID
(20 min) Laurence Lundblade

13:50 : 14:10 Claims to carry Attestation Results to Relying Parties
(20 min) Laurence Lundblade

14:10 – 14:20 TEEP requirements for EAT
(10 min) Dave Thaler

14:20 – 14:30 Open Mic
(10 min)
SUEID and EAT relation to DevID

- Laurence Lundblade
SUEID and IDevID
Both UEID and SUEID have the same format, one of these:

- 16, 24 or 32-byte binary string created with a crypto-quality random number generator or equivalent
- 6, 8-byte binary string that is an IEEE EUI – a MAC address is an IEEE EUI
- 14-byte binary string that is an IMEI – a mobile phone serial number

<table>
<thead>
<tr>
<th>UEID</th>
<th>SUEID</th>
</tr>
</thead>
<tbody>
<tr>
<td>One per device (or none)</td>
<td>One or more per device (or none)</td>
</tr>
<tr>
<td>Assigned at manufacture and never changes</td>
<td>Created and destroyed in device life-cycle events like ownership change and factory reset</td>
</tr>
<tr>
<td>No label</td>
<td>A simple string label to distinguish one from another</td>
</tr>
<tr>
<td>Like an IDevID</td>
<td>Like an LDevID</td>
</tr>
</tbody>
</table>

There is one additional type byte, so the actual lengths are: 17, 25, 33, 7, 9 or 15 bytes
Three ways EAT implementations relate to IDevID implementations

- The EAT protocol is used with an IDevID – Both are implemented and work together
- EAT as a competitor to IDevID to provide identity and manufacturer info – It’s one or the other
- EAT claims are added into an IDevID – Parts of EAT are stuffed into an IDevID implementation
The EAT protocol used with an IDevID

Both the EAT and the IDevID may have a device and vendor identifier:
- The UEID and OEM ID in EAT
- In the X.509 subject field in the Endorsement/IDevID
- These should probably be made identical or one derived from the other
IDevID used for identity – can be thought of as a competitor to EAT

This provides signed / secured device and manufacturer identity in the certificate.
This architecture can be seen as competing with EAT which can also provide this.
EAT inside a DevID Certificate

- EAT claims can be put into an X.509 v3 extension in a DevID certificate
  - Option 1: define ASN.1 syntax and OID for each EAT claim that is to be included
  - Option 2: one OID that contains a CBOR/UCCS format EAT

Note:
- Only works for static EAT claims because DevIDs are not generated on device
  - For example, can’t work with GPS location, debug status, some SW measurements
- EAT is not functioning as the protocol between device and relying party that proves the identity of the device, some message/protocol is still required
IDevID expanded with EAT claims

Device

DevID Module
signs the nonce

Private Key

Endosser / Manufacturer

Verifier / Relying Party

nonce

Signed nonce

Endorsement / IDevID certificate
EAT claims

Root Certificate

Can provide EAT claims in a largely IDevID-compatible way
Only static EAT claims since EAT is not generated on device
Claims to carry Attestation Results to Relying Parties

- Laurence Lundblade
Attestation Results
Purpose of Attestation

- The end purpose of RATS is to give results to the Relying Party
- The Relying Party makes the decision to allow the financial transaction, to allow the device on the network, to believe the data received,…
- RATS exists to serve the Relying Party
  - Relying party may use machine learning and want every scrap of information of even remote value
- EAT is a relatively obvious choice to convey Attestation Results to the Relying Party
  - Supports JSON, a common representation for the server side
  - Flexible security options: EAT/CWT or UCCS + TLS or UCCS + other
  - Many claims are appropriate to pass directly through the Verifier to the Relying Party
<table>
<thead>
<tr>
<th>Claims that are useful to pass-through Verifier to Relying Party</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nonce</td>
</tr>
<tr>
<td>Must have a nonce from the relying party</td>
</tr>
<tr>
<td>UEID, SUEID</td>
</tr>
<tr>
<td>Relying parties like device identification when privacy policy allows</td>
</tr>
<tr>
<td>OEM ID</td>
</tr>
<tr>
<td>Identifies manufacturer of device</td>
</tr>
<tr>
<td>HW Version</td>
</tr>
<tr>
<td>Sometimes useful to relying party</td>
</tr>
<tr>
<td>Boot and debug status</td>
</tr>
<tr>
<td>Useful when higher security is required</td>
</tr>
<tr>
<td>Location</td>
</tr>
<tr>
<td>Often useful to relying party</td>
</tr>
<tr>
<td>Uptime and boot seed</td>
</tr>
<tr>
<td>Sometimes useful to relying party</td>
</tr>
<tr>
<td>Software manifest</td>
</tr>
<tr>
<td>Contain software versions</td>
</tr>
<tr>
<td>Software results – The results of a software measurement (in a Github, not yet in an EAT draft)</td>
</tr>
<tr>
<td>Some (TEE-based) Attesters can measure AND validate subsystems and thus measurement results can go directly from Attester to Relying Party</td>
</tr>
<tr>
<td>Key material, particularly a public key</td>
</tr>
<tr>
<td>This may enable further protocols between the device and Relying Party (e.g., FIDO, payments, Android key store...)</td>
</tr>
<tr>
<td>Submodules</td>
</tr>
<tr>
<td>For example, many submodules (the TEE, the HLOS, the Secure Element) may participate in a payment</td>
</tr>
</tbody>
</table>
# Claims Generated by the Verifier for the Relying Party

<table>
<thead>
<tr>
<th>Token ID</th>
<th>Identifies the particular report to the RP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time stamp</td>
<td>When the results were generated</td>
</tr>
<tr>
<td>Nonce</td>
<td>Freshness between Verifier and Relying Party</td>
</tr>
<tr>
<td>Security Level</td>
<td>If the Attester doesn’t include the claim, the Verifier may have information to know the security level and report it</td>
</tr>
<tr>
<td>Software Results (described only in Github document, not yet in a published EAT draft)</td>
<td>RP will be very interested in the results of the measurement comparison to reference values</td>
</tr>
<tr>
<td>Digital Letter of Authorization – List of certifications received by device (described only in Github document, not yet in a published EAT draft)</td>
<td>Lists certifications granted to the device. For example, Common Criteria or Global Platform certifications</td>
</tr>
</tbody>
</table>
# A DLOA

## Digital Letter of Approval (format is XML)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authority_Label</td>
<td>Names the authority that issued the certification</td>
<td>EMVCo</td>
</tr>
<tr>
<td>LOA_Identifier</td>
<td>More or less a serial number for the certification</td>
<td>PCN0156.13</td>
</tr>
<tr>
<td>LOA_Scope</td>
<td>Scope of the LOA</td>
<td>(unable to find example)</td>
</tr>
<tr>
<td>Platform_Label</td>
<td>Manufacturer identified by OID plus product identified by text string</td>
<td>1.2.840.114283/My_Platform_Label_1a</td>
</tr>
<tr>
<td>Issuance_Date</td>
<td>Date issued</td>
<td>19 Jun 2018</td>
</tr>
<tr>
<td>Expiration_Date</td>
<td>Date of expiration</td>
<td>19 Jun 2022</td>
</tr>
</tbody>
</table>

Digitally signed with XML signature

- A digital instantiation of the letter of approval typically issued by a certification authority
- Always retrieved by URL from a DLOA registrar
DLOA Claim

- An array of one or more references to a DLOA
- Each DLOA reference contains:
  - Fields to construct URL to fetch DLOA
  - Registrar URI
  - Platform Label
  - Application Label if DLOA is for an application, not a platform
- DLOA claim must only be present if certification was granted
- A DLOA’s scope is limited to the submodule it is in

```plaintext
dloas-claim = (
    dloas => [ + dloa-type ]
)
dloa-type = [
    dloa_registrar: ~uri
    dloa_platform_label: text
    ? dloa_application_label: text
]
```
# The `swresult` Claim

A high-level summary report of the verification of a software measurement  
Each claim may contain multiple results

<table>
<thead>
<tr>
<th>An individual result is an array of three or four items</th>
</tr>
</thead>
<tbody>
<tr>
<td>The name of the measurement system or scheme (required)</td>
</tr>
</tbody>
</table>
| objective – what software measured (required) | Enumerated type that is one of:  
  • all  
  • firmware  
  • kernel  
  • privileged  
  • system-libs  
  • partial |
| verification result (required) | Enumerated type that is one of:  
  • verification-not-run  
  • verification-indeterminate  
  • verification-failed  
  • fully-verified  
  • partially-verified |
| objective name (optional) | Textual name of the objective. For example, “Android kernel” |
TEEP Requirements for EAT

● Dave Thaler
TEEP Requirements for EAT

Dave Thaler <dthaler@microsoft.com>
# TEEP WG has requirements for abstract data in Attestation Results (e.g., to do remediation)

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Claim</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device unique ID</td>
<td>device-identifier</td>
<td>draft-birkholz-rats-suit-claims, §3.1.3</td>
</tr>
<tr>
<td>Vendor of the device</td>
<td>vendor-identifier</td>
<td>draft-birkholz-rats-suit-claims, §3.1.1</td>
</tr>
<tr>
<td>Class of the device</td>
<td>class-identifier</td>
<td>draft-birkholz-rats-suit-claims, §3.1.2</td>
</tr>
<tr>
<td>TEE hardware type</td>
<td>chip-version-scheme</td>
<td>draft-ietf-rats-eat, §3.7</td>
</tr>
<tr>
<td>TEE hardware version</td>
<td>chip-version-scheme</td>
<td>draft-ietf-rats-eat, §3.7</td>
</tr>
<tr>
<td>TEE firmware (e.g., TF-A) ID</td>
<td>component-identifier</td>
<td>draft-birkholz-rats-suit-claims, §3.1.4</td>
</tr>
<tr>
<td>TEE firmware version</td>
<td>version</td>
<td>draft-birkholz-rats-suit-claims, §3.1.8</td>
</tr>
<tr>
<td>TEE software (e.g., OP-TEE) ID</td>
<td>component-identifier</td>
<td>draft-birkholz-rats-suit-claims, §3.1.4</td>
</tr>
<tr>
<td>TEE software version</td>
<td>version</td>
<td>draft-birkholz-rats-suit-claims, §3.1.8</td>
</tr>
<tr>
<td>Freshness proof (nonce)</td>
<td>nonce</td>
<td>draft-ietf-rats-eat, §3.3</td>
</tr>
<tr>
<td>Freshness proof (timestamp)</td>
<td>iat</td>
<td>draft-ietf-rats-eat, §3.2</td>
</tr>
<tr>
<td>Freshness proof (epoch ID)</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>
3. **SUIT Claims**

3.1. **System Properties Claims**

3.1.1. **vendor-identifier**

3.1.2. **class-identifier**

3.1.3. **device-identifier**

3.1.4. **component-identifier**

3.1.5. **image-digest**

3.1.6. **image-size**

3.1.7. **minimum-battery**

3.1.8. **version**

3.2. **Interpreter Record Claims**

3.2.1. **record-success**

3.2.2. **component-index**

3.2.3. **dependency-index**

3.2.4. **command-index**

3.2.5. **nominal-parameters**

3.2.6. **nominal-parameters**

3.3. **Generic Record Conditions (TBD)**

---

Per past RATS list discussion, these claims are not SUIT specific.
Options:
A. RATS WG, even if some claims are SUIT specific
B. SUIT WG, even if some claims are not SUIT specific
C. Split doc: SUIT WG for SUIT claims, RATS WG for general claims

My preference: option C with general claims added into EAT spec
• “System Properties Claims” fall under RATS charter item for “claims which provide information about system components characteristics scoped by the specified use-cases”
TEEP implementation requirements

• From draft-ietf-teep-protocol D.3.1 example:

```json
/ eat-claim-set = /
{
  / issuer /  1: "joe",
  / timestamp (iat) /  6: 1(1526542894)
  / nonce /  10: h'948f8860d13a463e8e'
  / secure-boot /  15: true,
  / debug-status /  16: 3, / disabled-permanently /
  / security-level /  14: 3, / secure-restricted /
  / device-identifier /  <TBD>: h'e99600dd921649798b013e9752dcf0c5',
  / vendor-identifier /  <TBD>: h'2b03879b33434a7ca682b8af84c19fd4',
  / class-identifier /  <TBD>: h'9714a5796bd245a3a4ab4f977cb8487f',
  / chip-version-scheme /  <TBD>: "MyTEE v1.0",
  / component-identifier /  <TBD>: h'60822887d35e43d5b603d18bcaa3f08d',
  / version /  <TBD>: "v0.1"
}
```

• Need early assignment to unblock implementations
The hardware version is a simple text string the format of which is set by each manufacturer. The structure and sorting order of this text string can be specified using the version-scheme item from CoSWID [CoSWID].

draft-ietf-sacm-coswid:

$version-scheme /= multipartnumeric
$version-scheme /= multipartnumeric-suffix
$version-scheme /= alphanumeric
$version-scheme /= decimal
$version-scheme /= semver
$version-scheme /= uint / text
draft-birkholz-rats-suit-claims

• `version` => `version-value`
  • Should probably be `$version-scheme`
  • Can this replace (be renamed from) `chip-version-scheme-claim`?
• `device-identifier` => `RFC4122_UUID`
• `vendor-identifier` => `RFC4122_UUID`
• `class-identifier` => `RFC4122_UUID`
• `class-identifier` => `[ + identifier ]`
  • “A binary identifier can represent a CoSWID [I-D.ietf-sacm-coswid] tag-id.”
Thank You!