

Workgroup: RATS Working Group  
Internet-Draft:  
draft-ietf-rats-architecture-11

Published: 30 March 2021

Intended Status: Informational

Expires: 1 October 2021

Authors: H. Birkholz            D. Thaler  
          Fraunhofer SIT        Microsoft  
          M. Richardson            N. Smith  
          Sandelman Software Works    Intel  
          W. Pan  
          Huawei Technologies

## **Remote Attestation Procedures Architecture**

### **Abstract**

In network protocol exchanges it is often useful for one end of a communication to know whether the other end is in an intended operating state. This document provides an architectural overview of the entities involved that make such tests possible through the process of generating, conveying, and evaluating evidentiary claims. An attempt is made to provide for a model that is neutral toward processor architectures, the content of claims, and protocols.

### **Note to Readers**

Discussion of this document takes place on the RATS Working Group mailing list (rats@ietf.org), which is archived at <https://mailarchive.ietf.org/arch/browse/rats/>.

Source for this draft and an issue tracker can be found at <https://github.com/ietf-rats-wg/architecture>.

### **Status of This Memo**

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 1 October 2021.

## Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

- [1. Introduction](#)
- [2. Reference Use Cases](#)
  - [2.1. Network Endpoint Assessment](#)
  - [2.2. Confidential Machine Learning Model Protection](#)
  - [2.3. Confidential Data Protection](#)
  - [2.4. Critical Infrastructure Control](#)
  - [2.5. Trusted Execution Environment Provisioning](#)
  - [2.6. Hardware Watchdog](#)
  - [2.7. FIDO Biometric Authentication](#)
- [3. Architectural Overview](#)
  - [3.1. Appraisal Policies](#)
  - [3.2. Reference Values](#)
  - [3.3. Two Types of Environments of an Attester](#)
  - [3.4. Layered Attestation Environments](#)
  - [3.5. Composite Device](#)
  - [3.6. Implementation Considerations](#)
- [4. Terminology](#)
  - [4.1. Roles](#)
  - [4.2. Artifacts](#)
- [5. Topological Patterns](#)
  - [5.1. Passport Model](#)
  - [5.2. Background-Check Model](#)
  - [5.3. Combinations](#)
- [6. Roles and Entities](#)
- [7. Trust Model](#)
  - [7.1. Relying Party](#)
  - [7.2. Attester](#)
  - [7.3. Relying Party Owner](#)
  - [7.4. Verifier](#)
  - [7.5. Endorser, Reference Value Provider, and Verifier Owner](#)
- [8. Conceptual Messages](#)
  - [8.1. Evidence](#)

- [8.2. Endorsements](#)
- [8.3. Attestation Results](#)
- [9. Claims Encoding Formats](#)
- [10. Freshness](#)
  - [10.1. Explicit Timekeeping using Synchronized Clocks](#)
  - [10.2. Implicit Timekeeping using Nonces](#)
  - [10.3. Implicit Timekeeping using Epoch IDs](#)
  - [10.4. Discussion](#)
- [11. Privacy Considerations](#)
- [12. Security Considerations](#)
  - [12.1. Attester and Attestation Key Protection](#)
    - [12.1.1. On-Device Attester and Key Protection](#)
    - [12.1.2. Attestation Key Provisioning Processes](#)
  - [12.2. Integrity Protection](#)
  - [12.3. Epoch ID-based Attestation](#)
- [13. IANA Considerations](#)
- [14. Acknowledgments](#)
- [15. Notable Contributions](#)
- [16. Appendix A: Time Considerations](#)
  - [16.1. Example 1: Timestamp-based Passport Model Example](#)
  - [16.2. Example 2: Nonce-based Passport Model Example](#)
  - [16.3. Example 3: Epoch ID-based Passport Model Example](#)
  - [16.4. Example 4: Timestamp-based Background-Check Model Example](#)
  - [16.5. Example 5: Nonce-based Background-Check Model Example](#)
- [17. References](#)
  - [17.1. Normative References](#)
  - [17.2. Informative References](#)
- [Contributors](#)
- [Authors' Addresses](#)

## **1. Introduction**

The question of how one system can know that another system can be trusted has found new interest and relevance in a world where trusted computing elements are maturing in processor architectures.

Systems that have been attested and verified to be in a good state (for some value of "good") can improve overall system posture. Conversely, systems that cannot be attested and verified to be in a good state can be taken out of service, or otherwise flagged for repair.

For example:

\*A bank back-end system might refuse to transact with another system that is not known to be in a good state.

\*A healthcare system might refuse to transmit electronic healthcare records to a system that is not known to be in a good state.

In Remote Attestation Procedures (RATS), one peer (the "Attester") produces believable information about itself - Evidence - to enable a remote peer (the "Relying Party") to decide whether to consider that Attester a trustworthy peer or not. RATS are facilitated by an additional vital party, the Verifier.

The Verifier appraises Evidence via appraisal policies and creates the Attestation Results to support Relying Parties in their decision process. This document defines a flexible architecture consisting of attestation roles and their interactions via conceptual messages. Additionally, this document defines a universal set of terms that can be mapped to various existing and emerging Remote Attestation Procedures. Common topological patterns and the sequence of data flows associated with them, such as the "Passport Model" and the "Background-Check Model", are illustrated. The purpose is to define useful terminology for remote attestation and enable readers to map their solution architecture to the canonical attestation architecture provided here. Having a common terminology that provides well-understood meanings for common themes such as roles, device composition, topological patterns, and appraisal procedures is vital for semantic interoperability across solutions and platforms involving multiple vendors and providers.

Amongst other things, this document is about trust and trustworthiness. Trust is a choice one makes about another system. Trustworthiness is a quality about the other system that can be used in making one's decision to trust it or not. This is subtle difference and being familiar with the difference is crucial for using this document. Additionally, the concepts of freshness and trust relationships with respect to RATS are elaborated on to enable implementers to choose appropriate solutions to compose their Remote Attestation Procedures.

## **2. Reference Use Cases**

This section covers a number of representative and generic use cases for remote attestation, independent of specific solutions. The purpose is to provide motivation for various aspects of the architecture presented in this document. Many other use cases exist, and this document does not intend to have a complete list, only to illustrate a set of use cases that collectively cover all the functionality required in the architecture.

Each use case includes a description followed by an additional summary of the Attester and Relying Party roles derived from the use case.

### **2.1. Network Endpoint Assessment**

Network operators want a trustworthy report that includes identity and version information about the hardware and software on the machines attached to their network, for purposes such as inventory, audit, anomaly detection, record maintenance and/or trending reports (logging). The network operator may also want a policy by which full access is only granted to devices that meet some definition of hygiene, and so wants to get Claims about such information and verify its validity. Remote attestation is desired to prevent vulnerable or compromised devices from getting access to the network and potentially harming others.

Typically, solutions start with a specific component (called a root of trust) that is intended to provide trustworthy device identity and protected storage for measurements. The system components perform a series of measurements that may be signed via functions provided by a root of trust, considered as Evidence about present system components, such as hardware, firmware, BIOS, software, etc.

**Attester:** A device desiring access to a network.

**Relying Party:** Network equipment such as a router, switch, or access point, responsible for admission of the device into the network.

### **2.2. Confidential Machine Learning Model Protection**

A device manufacturer wants to protect its intellectual property. The intellectual property's scope primarily encompasses the machine learning (ML) model that is deployed in the devices purchased by its customers. The protection goals include preventing attackers, potentially the customer themselves, from seeing the details of the model.

This typically works by having some protected environment in the device go through a remote attestation with some manufacturer service that can assess its trustworthiness. If remote attestation succeeds, then the manufacturer service releases either the model, or a key to decrypt a model already deployed on the Attester in encrypted form, to the requester.

**Attester:** A device desiring to run an ML model.

**Relying Party:** A server or service holding ML models it desires to protect.

### 2.3. Confidential Data Protection

This is a generalization of the ML model use case above, where the data can be any highly confidential data, such as health data about customers, payroll data about employees, future business plans, etc. As part of the attestation procedure, an assessment is made against a set of policies to evaluate the state of the system that is requesting the confidential data. Attestation is desired to prevent leaking data via compromised devices.

**Attester:** An entity desiring to retrieve confidential data.

**Relying Party:** An entity that holds confidential data for release to authorized entities.

### 2.4. Critical Infrastructure Control

Potentially harmful physical equipment (e.g., power grid, traffic control, hazardous chemical processing, etc.) is connected to a network in support of critical infrastructure. The organization managing such infrastructure needs to ensure that only authorized code and users can control corresponding critical processes, and that these processes are protected from unauthorized manipulation or other threats. When a protocol operation can affect a critical system component of the infrastructure, devices attached to that critical component require some assurances depending on the security context, including that: a requesting device or application has not been compromised, and the requesters and actors act on applicable policies. As such, remote attestation can be used to only accept commands from requesters that are within policy.

**Attester:** A device or application wishing to control physical equipment.

**Relying Party:** A device or application connected to potentially dangerous physical equipment (hazardous chemical processing, traffic control, power grid, etc.).

### 2.5. Trusted Execution Environment Provisioning

A Trusted Application Manager (TAM) server is responsible for managing the applications running in a Trusted Execution Environment (TEE) of a client device. To achieve its purpose, the TAM needs to assess the state of a TEE, or of applications in the TEE, of a client device. The TEE conducts Remote Attestation Procedures with the TAM, which can then decide whether the TEE is already in compliance with the TAM's latest policy. If not, the TAM has to uninstall, update, or install approved applications in the TEE to bring it back into compliance with the TAM's policy.

**Attester:**

A device with a TEE capable of running trusted applications that can be updated.

**Relying Party:** A TAM.

**2.6. Hardware Watchdog**

There is a class of malware that holds a device hostage and does not allow it to reboot to prevent updates from being applied. This can be a significant problem, because it allows a fleet of devices to be held hostage for ransom.

A solution to this problem is a watchdog timer implemented in a protected environment such as a Trusted Platform Module (TPM), as described in [[TCGArch](#)] section 43.3. If the watchdog does not receive regular, and fresh, Attestation Results as to the system's health, then it forces a reboot.

**Attester:** The device that should be protected from being held hostage for a long period of time.

**Relying Party:** A watchdog capable of triggering a procedure that resets a device into a known, good operational state.

**2.7. FIDO Biometric Authentication**

In the Fast IDentity Online (FIDO) protocol [[WebAuthN](#)], [[CTAP](#)], the device in the user's hand authenticates the human user, whether by biometrics (such as fingerprints), or by PIN and password. FIDO authentication puts a large amount of trust in the device compared to typical password authentication because it is the device that verifies the biometric, PIN and password inputs from the user, not the server. For the Relying Party to know that the authentication is trustworthy, the Relying Party needs to know that the Authenticator part of the device is trustworthy. The FIDO protocol employs remote attestation for this.

The FIDO protocol supports several remote attestation protocols and a mechanism by which new ones can be registered and added. Remote attestation defined by RATS is thus a candidate for use in the FIDO protocol.

Other biometric authentication protocols such as the Chinese IFAA standard and WeChat Pay as well as Google Pay make use of remote attestation in one form or another.

**Attester:**

Every FIDO Authenticator contains an Attester.

**Relying Party:** Any web site, mobile application back-end, or service that relies on authentication data based on biometric information.

**3. Architectural Overview**

[Figure 1](#) depicts the data that flows between different roles, independent of protocol or use case.

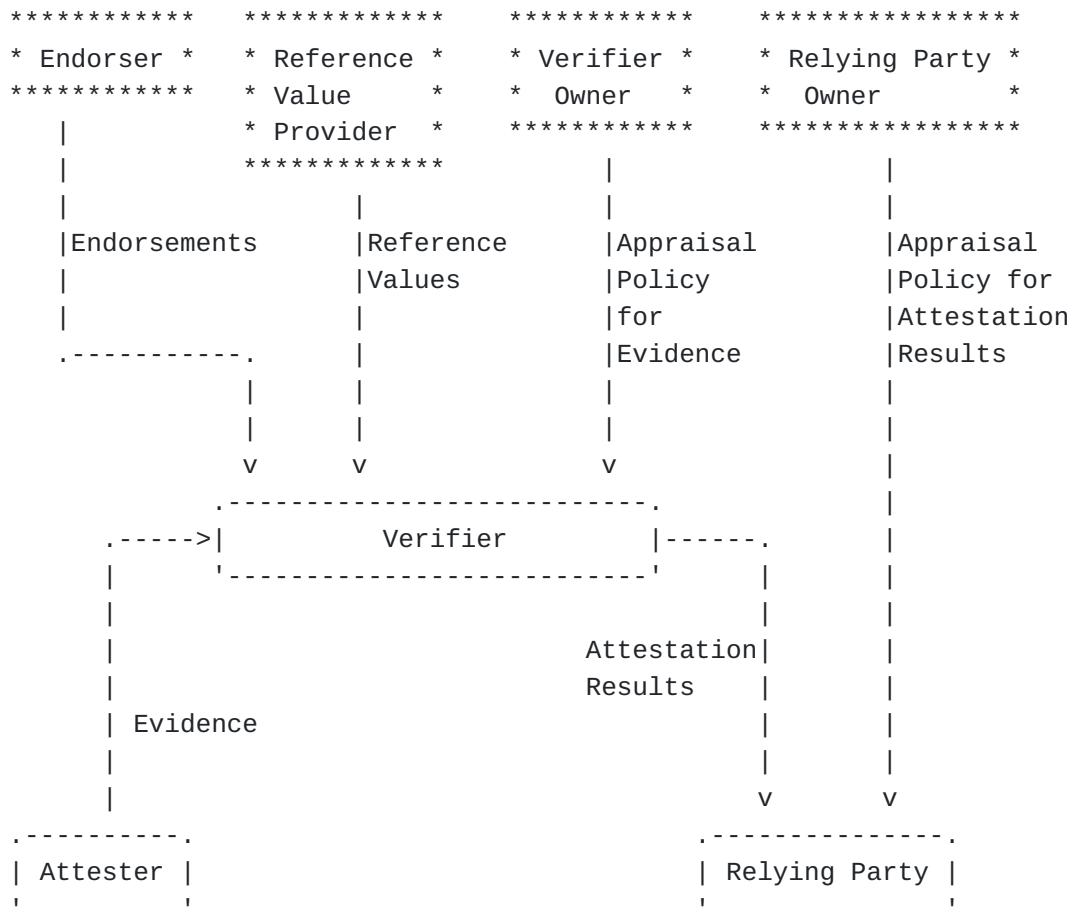


Figure 1: Conceptual Data Flow

The text below summarizes the activities conducted by the roles illustrated in [Figure 1](#).

An Attester creates Evidence that is conveyed to a Verifier.

A Verifier uses the Evidence, any Reference Values from Reference Value Providers, and any Endorsements from Endorsers, by applying an Appraisal Policy for Evidence to assess the trustworthiness of the Attester. This procedure is called the appraisal of Evidence.



Subsequently, the Verifier generates Attestation Results for use by Relying Parties. The Appraisal Policy for Evidence might be obtained from an Endorser along with the Endorsements, and/or might be obtained via some other mechanism, such as being configured in the Verifier by the Verifier Owner.

A Relying Party uses Attestation Results by applying its own appraisal policy to make application-specific decisions, such as authorization decisions. The Appraisal Policy for Attestation Results is configured in the Relying Party by the Relying Party Owner, and/or are programmed into the Relying Party. This procedure is called the appraisal of Attestation Results.

### **3.1. Appraisal Policies**

The Verifier, when appraising Evidence, or the Relying Party, when appraising Attestation Results, checks the values of some Claims against constraints specified in its appraisal policy. Examples of such constraints checking include:

- \*comparison for equality against a Reference Value, or
- \*a check for being in a range bounded by Reference Values, or
- \*membership in a set of Reference Values, or
- \*a check against values in other Claims.

The actual data format and semantics of any Appraisal Policy is implementation specific.

### **3.2. Reference Values**

Reference Values used in appraisal procedures come from a Reference Value Provider and are then used by the appraisal policy.

The actual data format and semantics of any Reference Values are specific to Claims and implementations. This architecture document does not define any general purpose format for Reference Values or general means for comparison.

### **3.3. Two Types of Environments of an Attester**

As shown in [Figure 2](#), an Attester consists of at least one Attesting Environment and at least one Target Environment. In some implementations, the Attesting and Target Environments might be combined. Other implementations might have multiple Attesting and Target Environments, such as in the examples described in more detail in [Section 3.4](#) and [Section 3.5](#). Other examples may exist. All

compositions of Attesting and Target Environments discussed in this architecture can be combined into more complex implementations.

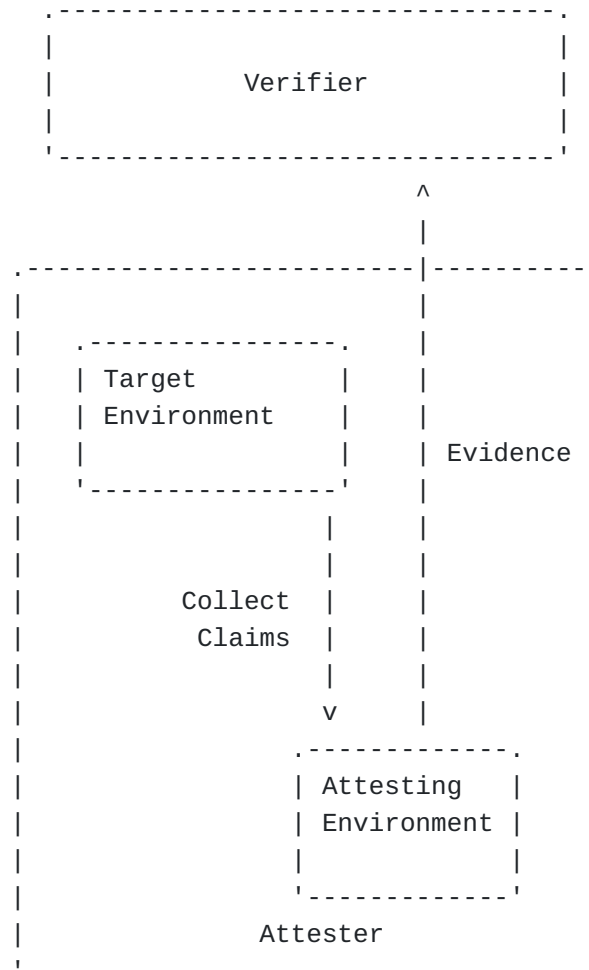


Figure 2: Two Types of Environments

Claims are collected from Target Environments. That is, Attesting Environments collect the values and the information to be represented in Claims, by reading system registers and variables, calling into subsystems, taking measurements on code, memory, or other security related assets of the Target Environment. Attesting Environments then format the Claims appropriately, and typically use key material and cryptographic functions, such as signing or cipher algorithms, to generate Evidence. There is no limit to or requirement on the types of hardware or software environments that can be used to implement an Attesting Environment, for example: Trusted Execution Environments (TEEs), embedded Secure Elements (eSEs), Trusted Platform Modules (TPMs), or BIOS firmware.

An arbitrary execution environment may not, by default, be capable of Claims collection for a given Target Environment. Execution

environments that are designed specifically to be capable of Claims collection are referred to in this document as Attesting Environments. For example, a TPM doesn't actively collect Claims itself, it instead requires another component to feed various values to the TPM. Thus, an Attesting Environment in such a case would be the combination of the TPM together with whatever component is feeding it the measurements.

### **3.4. Layered Attestation Environments**

By definition, the Attester role generates Evidence. An Attester may consist of one or more nested environments (layers). The root layer of an Attester includes at least one root of trust. In order to appraise Evidence generated by an Attester, the Verifier needs to trust the Attester's root of trust. Trust in the Attester's root of trust can be established either directly (e.g., the Verifier puts the root of trust's public key into its trust anchor store) or transitively via an Endorser (e.g., the Verifier puts the Endorser's public key into its trust anchor store). In layered attestation, a root of trust is the initial Attesting Environment. Claims can be collected from or about each layer. The corresponding Claims can be structured in a nested fashion that reflects the nesting of the Attester's layers. Normally, Claims are not self-asserted, rather a previous layer acts as the Attesting Environment for the next layer. Claims about a root of trust typically are asserted by an Endorser.

The device illustrated in [Figure 3](#) includes (A) a BIOS stored in read-only memory, (B) an operating system kernel, and (C) an application or workload.

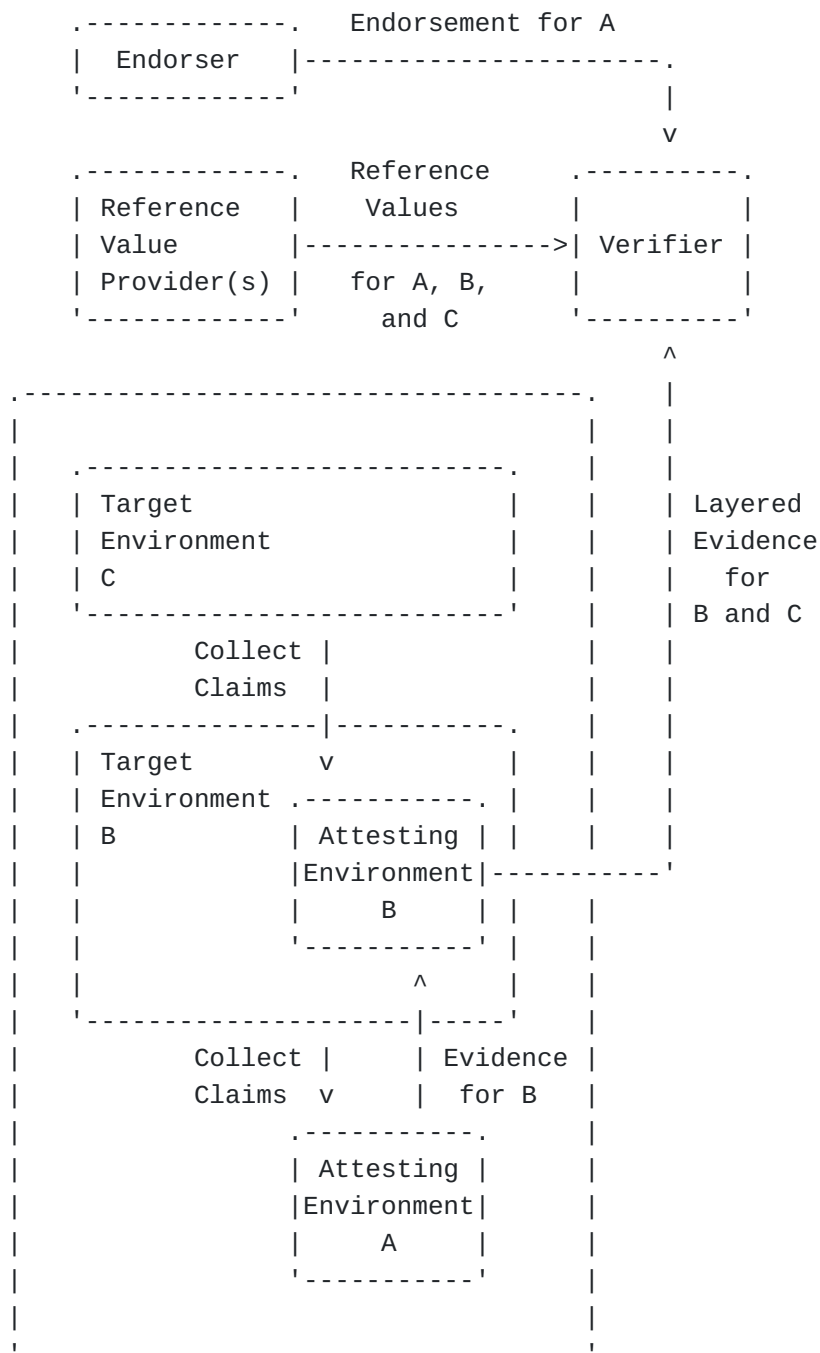


Figure 3: Layered Attester

Attesting Environment A, the read-only BIOS in this example, has to ensure the integrity of the bootloader (Target Environment B). There are potentially multiple kernels to boot, and the decision is up to the bootloader. Only a bootloader with intact integrity will make an appropriate decision. Therefore, the Claims relating to the integrity of the bootloader have to be measured securely. At this stage of the boot-cycle of the device, the Claims collected typically cannot be composed into Evidence.

After the boot sequence is started, the BIOS conducts the most important and defining feature of layered attestation, which is that the successfully measured Target Environment B now becomes (or contains) an Attesting Environment for the next layer. This procedure in layered attestation is sometimes called "staging". It is important that the new Attesting Environment B not be able to alter any Claims about its own Target Environment B. This can be ensured having those Claims be either signed by Attesting Environment A or stored in an untamperable manner by Attesting Environment A.

Continuing with this example, the bootloader's Attesting Environment B is now in charge of collecting Claims about Target Environment C, which in this example is the kernel to be booted. The final Evidence thus contains two sets of Claims: one set about the bootloader as measured and signed by the BIOS, plus a set of Claims about the kernel as measured and signed by the bootloader.

This example could be extended further by making the kernel become another Attesting Environment for an application as another Target Environment. This would result in a third set of Claims in the Evidence pertaining to that application.

The essence of this example is a cascade of staged environments. Each environment has the responsibility of measuring the next environment before the next environment is started. In general, the number of layers may vary by device or implementation, and an Attesting Environment might even have multiple Target Environments that it measures, rather than only one as shown in [Figure 3](#).

### **3.5. Composite Device**

A composite device is an entity composed of multiple sub-entities such that its trustworthiness has to be determined by the appraisal of all these sub-entities.

Each sub-entity has at least one Attesting Environment collecting the Claims from at least one Target Environment, then this sub-entity generates Evidence about its trustworthiness. Therefore, each sub-entity can be called an Attester. Among all the Attesters, there may be only some which have the ability to communicate with the Verifier while others do not.

For example, a carrier-grade router consists of a chassis and multiple slots. The trustworthiness of the router depends on all its slots' trustworthiness. Each slot has an Attesting Environment, such as a TEE, collecting the Claims of its boot process, after which it generates Evidence from the Claims.

Among these slots, only a "main" slot can communicate with the Verifier while other slots cannot. But other slots can communicate with the main slot by the links between them inside the router. So the main slot collects the Evidence of other slots, produces the final Evidence of the whole router and conveys the final Evidence to the Verifier. Therefore the router is a composite device, each slot is an Attester, and the main slot is the lead Attester.

Another example is a multi-chassis router composed of multiple single carrier-grade routers. Multi-chassis router setups create redundancy groups that provide higher throughput by interconnecting multiple routers in these groups, which can be treated as one logical router for simpler management. A multi-chassis router setup provides a management point that connects to the Verifier. Typically one router in the group is designated as the main router. Other routers in the multi-chassis setup are connected to the main router only via physical network links and are therefore managed and appraised via the main router's help. In consequence, a multi-chassis router setup is a composite device, each router is an Attester, and the main router is the lead Attester.

[Figure 4](#) depicts the conceptual data flow for a composite device.

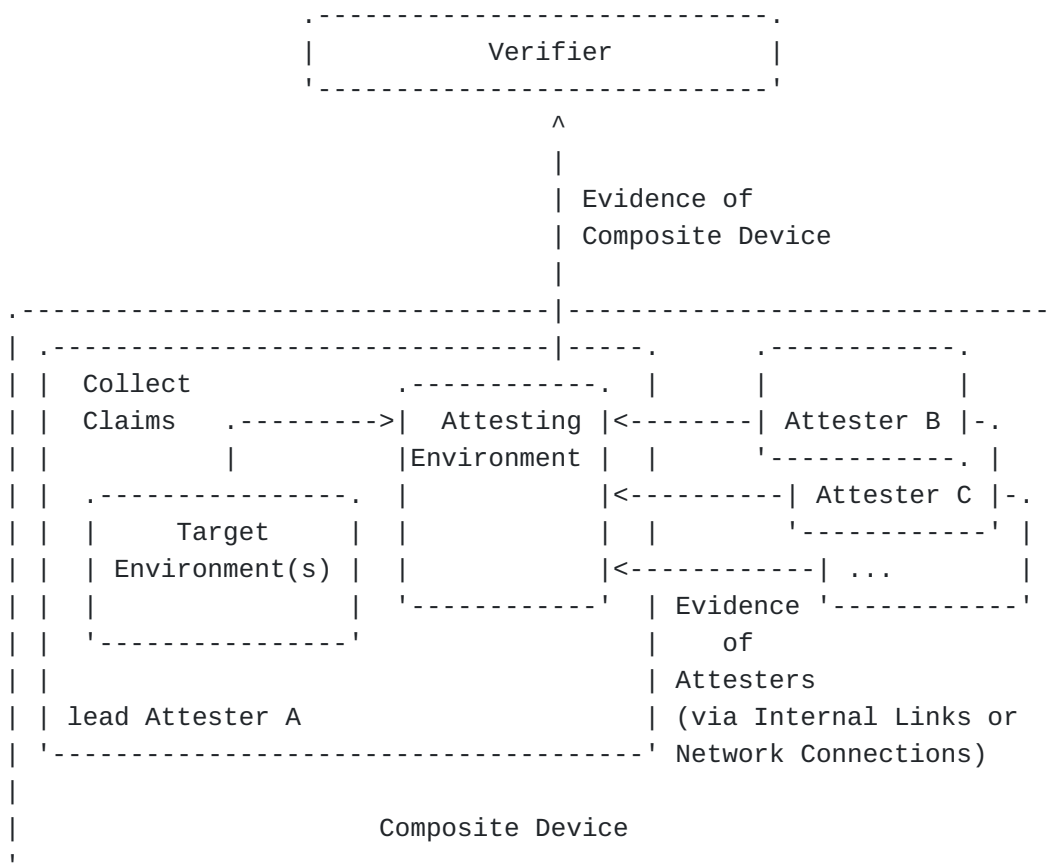


Figure 4: Composite Device

In a composite device, each Attester generates its own Evidence by its Attesting Environment(s) collecting the Claims from its Target Environment(s). The lead Attester collects Evidence from other Attesters and conveys it to a Verifier. Collection of Evidence from sub-entities may itself be a form of Claims collection that results in Evidence asserted by the lead Attester. The lead Attester generates Evidence about the layout of the whole composite device, while sub-Attesters generate Evidence about their respective (sub-)modules.

In this scenario, the trust model described in [Section 7](#) can also be applied to an inside Verifier.

### 3.6. Implementation Considerations

An entity can take on multiple RATS roles (e.g., Attester, Verifier, Relying Party, etc.) at the same time. Multiple entities can cooperate to implement a single RATS role as well. In essence, the combination of roles and entities can be arbitrary. For example, in the composite device scenario, the entity inside the lead Attester can also take on the role of a Verifier, and the outer entity of Verifier can take on the role of a Relying Party. After collecting the Evidence of other Attesters, this inside Verifier uses Endorsements and appraisal policies (obtained the same way as by any other Verifier) as part of the appraisal procedures that generate Attestation Results. The inside Verifier then conveys the Attestation Results of other Attesters to the outside Verifier, whether in the same conveyance protocol as part of the Evidence or not.

## 4. Terminology

This document uses the following terms.

### 4.1. Roles

**Attester:** A role performed by an entity (typically a device) whose Evidence must be appraised in order to infer the extent to which the Attester is considered trustworthy, such as when deciding whether it is authorized to perform some operation.

Produces: Evidence

**Relying Party:** A role performed by an entity that depends on the validity of information about an Attester, for purposes of

reliably applying application specific actions. Compare /relying party/ in [[RFC4949](#)].

Consumes: Attestation Results

**Verifier:** A role performed by an entity that appraises the validity of Evidence about an Attester and produces Attestation Results to be used by a Relying Party.

Consumes: Evidence, Reference Values, Endorsements, Appraisal Policy for Evidence

Produces: Attestation Results

**Relying Party Owner:** A role performed by an entity (typically an administrator), that is authorized to configure Appraisal Policy for Attestation Results in a Relying Party.

Produces: Appraisal Policy for Attestation Results

**Verifier Owner:** A role performed by an entity (typically an administrator), that is authorized to configure Appraisal Policy for Evidence in a Verifier.

Produces: Appraisal Policy for Evidence

**Endorser:** A role performed by an entity (typically a manufacturer) whose Endorsements help Verifiers appraise the authenticity of Evidence.

Produces: Endorsements

**Reference Value Provider:** A role performed by an entity (typically a manufacturer) whose Reference Values help Verifiers appraise Evidence to determine if acceptable known Claims have been recorded by the Attester.

Produces: Reference Values



## 4.2. Artifacts

**Claim:** A piece of asserted information, often in the form of a name/value pair. Claims make up the usual structure of Evidence and other RATS artifacts. Compare /claim/ in [[RFC7519](#)].

**Endorsement:** A secure statement that an Endorser vouches for the integrity of an Attester's various capabilities such as Claims collection and Evidence signing.

Consumed By: Verifier

Produced By: Endorser

**Evidence:** A set of Claims generated by an Attester to be appraised by a Verifier. Evidence may include configuration data, measurements, telemetry, or inferences.

Consumed By: Verifier

Produced By: Attester

**Attestation Result:** The output generated by a Verifier, typically including information about an Attester, where the Verifier vouches for the validity of the results.

Consumed By: Relying Party

Produced By: Verifier

**Appraisal Policy for Evidence:** A set of rules that informs how a Verifier evaluates the validity of information about an Attester. Compare /security policy/ in [[RFC4949](#)].

Consumed By: Verifier

Produced By: Verifier Owner

**Appraisal Policy for Attestation Results:** A set of rules that direct how a Relying Party uses the Attestation Results regarding an Attester generated by the Verifiers. Compare /security policy/ in [[RFC4949](#)].

Consumed by: Relying Party

Produced by: Relying Party Owner

**Reference Values:** A set of values against which values of Claims can be compared as part of applying an Appraisal Policy for Evidence. Reference Values are sometimes referred to in other

documents as known-good values, golden measurements, or nominal values, although those terms typically assume comparison for equality, whereas here Reference Values might be more general and be used in any sort of comparison.

Consumed By: Verifier

Produced By: Reference Value Provider

## 5. Topological Patterns

[Figure 1](#) shows a data-flow diagram for communication between an Attester, a Verifier, and a Relying Party. The Attester conveys its Evidence to the Verifier for appraisal, and the Relying Party receives the Attestation Result from the Verifier. This section refines the data-flow diagram by describing two reference models, as well as one example composition thereof. The discussion that follows is for illustrative purposes only and does not constrain the interactions between RATS roles to the presented patterns.

### 5.1. Passport Model

The passport model is so named because of its resemblance to how nations issue passports to their citizens. The nature of the Evidence that an individual needs to provide to its local authority is specific to the country involved. The citizen retains control of the resulting passport document and presents it to other entities when it needs to assert a citizenship or identity Claim, such as an airport immigration desk. The passport is considered sufficient because it vouches for the citizenship and identity Claims, and it is issued by a trusted authority. Thus, in this immigration desk analogy, the passport issuing agency is a Verifier, the passport is an Attestation Result, and the immigration desk is a Relying Party.

In this model, an Attester conveys Evidence to a Verifier, which compares the Evidence against its appraisal policy. The Verifier then gives back an Attestation Result. If the Attestation Result was a successful one, the Attester can then present the Attestation Result (and possibly additional Claims) to a Relying Party, which then compares this information against its own appraisal policy.

Three ways in which the process may fail include:

- \*First, the Verifier may not issue a positive Attestation Result due to the Evidence not passing the Appraisal Policy for Evidence.

- \*The second way in which the process may fail is when the Attestation Result is examined by the Relying Party, and based

upon the Appraisal Policy for Attestation Results, the result does not pass the policy.

\*The third way is when the Verifier is unreachable or unavailable.

Since the resource access protocol between the Attester and Relying Party includes an Attestation Result, in this model the details of that protocol constrain the serialization format of the Attestation Result. The format of the Evidence on the other hand is only constrained by the Attester-Verifier remote attestation protocol. This implies that interoperability and standardization is more relevant for Attestation Results than it is for Evidence.

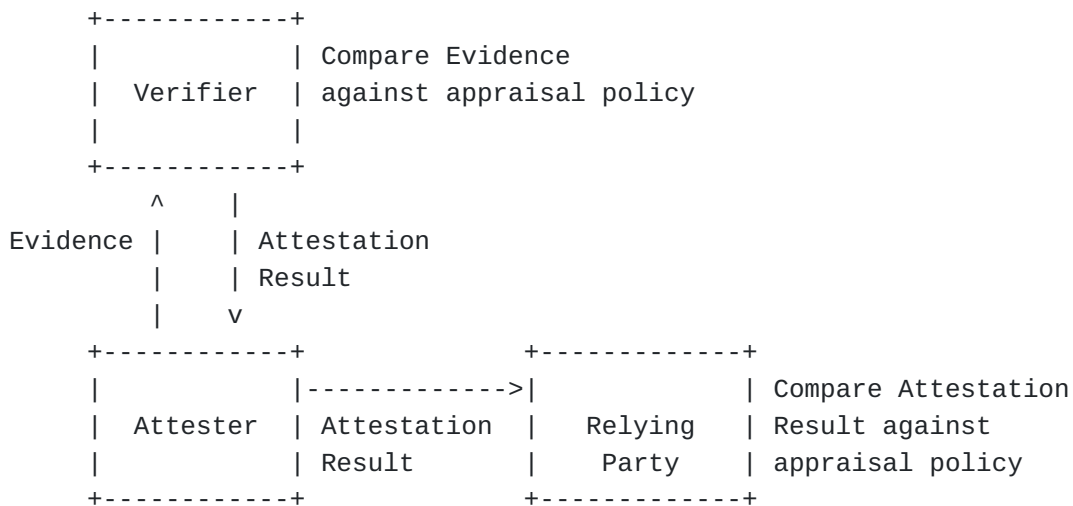


Figure 5: Passport Model

## 5.2. Background-Check Model

The background-check model is so named because of the resemblance of how employers and volunteer organizations perform background checks. When a prospective employee provides Claims about education or previous experience, the employer will contact the respective institutions or former employers to validate the Claim. Volunteer organizations often perform police background checks on volunteers in order to determine the volunteer's trustworthiness. Thus, in this analogy, a prospective volunteer is an Attester, the organization is the Relying Party, and the organization that issues a report is a Verifier.

In this model, an Attester conveys Evidence to a Relying Party, which simply passes it on to a Verifier. The Verifier then compares the Evidence against its appraisal policy, and returns an Attestation Result to the Relying Party. The Relying Party then compares the Attestation Result against its own appraisal policy.

The resource access protocol between the Attester and Relying Party includes Evidence rather than an Attestation Result, but that Evidence is not processed by the Relying Party. Since the Evidence is merely forwarded on to a trusted Verifier, any serialization format can be used for Evidence because the Relying Party does not need a parser for it. The only requirement is that the Evidence can be *encapsulated in* the format required by the resource access protocol between the Attester and Relying Party.

However, like in the Passport model, an Attestation Result is still consumed by the Relying Party. Code footprint and attack surface area can be minimized by using a serialization format for which the Relying Party already needs a parser to support the protocol between the Attester and Relying Party, which may be an existing standard or widely deployed resource access protocol. Such minimization is especially important if the Relying Party is a constrained node.

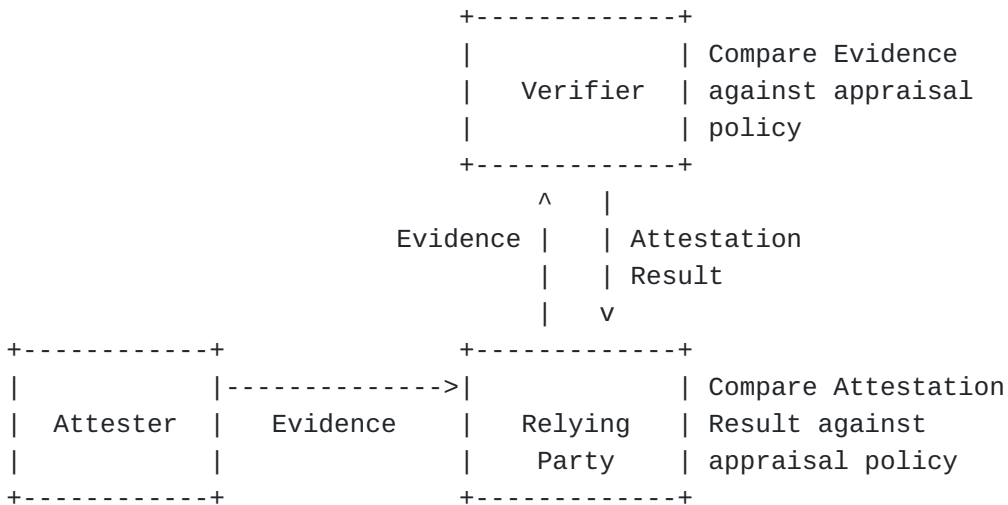


Figure 6: Background-Check Model

### 5.3. Combinations

One variation of the background-check model is where the Relying Party and the Verifier are on the same machine, performing both functions together. In this case, there is no need for a protocol between the two.

It is also worth pointing out that the choice of model depends on the use case, and that different Relying Parties may use different topological patterns.

The same device may need to create Evidence for different Relying Parties and/or different use cases. For instance, it would use one model to provide Evidence to a network infrastructure device to gain access to the network, and the other model to provide Evidence to a

server holding confidential data to gain access to that data. As such, both models may simultaneously be in use by the same device.

[Figure 7](#) shows another example of a combination where Relying Party 1 uses the passport model, whereas Relying Party 2 uses an extension of the background-check model. Specifically, in addition to the basic functionality shown in [Figure 6](#), Relying Party 2 actually provides the Attestation Result back to the Attester, allowing the Attester to use it with other Relying Parties. This is the model that the Trusted Application Manager plans to support in the TEEP architecture [[I-D.ietf-teep-architecture](#)].

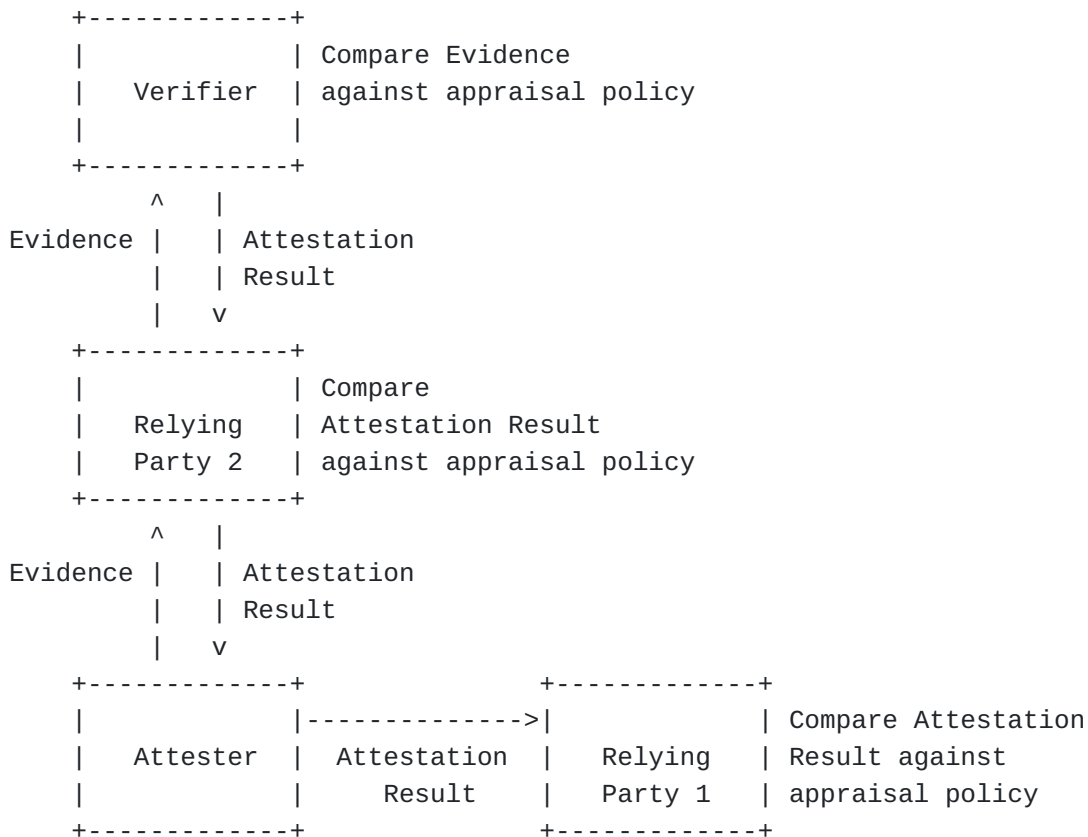


Figure 7: Example Combination

## 6. Roles and Entities

An entity in the RATS architecture includes at least one of the roles defined in this document.

An entity can aggregate more than one role into itself, such as being both a Verifier and a Relying Party, or being both a Reference Value Provider and an Endorser. As such, any conceptual messages (see [Section 8](#) for more discussion) originating from such roles might also be combined. For example, Reference Values might be

conveyed as part of an appraisal policy if the Verifier Owner and Reference Value Provider roles are combined. Similarly, Reference Values might be conveyed as part of an Endorsement if the Endorser and Reference Value Provider roles are combined.

Interactions between roles aggregated into the same entity do not necessarily use the Internet Protocol. Such interactions might use a loopback device or other IP-based communication between separate environments, but they do not have to. Alternative channels to convey conceptual messages include function calls, sockets, GPIO interfaces, local busses, or hypervisor calls. This type of conveyance is typically found in composite devices. Most importantly, these conveyance methods are out-of-scope of RATS, but they are presumed to exist in order to convey conceptual messages appropriately between roles.

For example, an entity that both connects to a wide-area network and to a system bus is taking on both the Attester and Verifier roles. As a system bus-connected entity, a Verifier consumes Evidence from other devices connected to the system bus that implement Attester roles. As a wide-area network connected entity, it may implement an Attester role.

In essence, an entity that combines more than one role creates and consumes the corresponding conceptual messages as defined in this document.

## **7. Trust Model**

### **7.1. Relying Party**

This document covers scenarios for which a Relying Party trusts a Verifier that can appraise the trustworthiness of information about an Attester. Such trust might come by the Relying Party trusting the Verifier (or its public key) directly, or might come by trusting an entity (e.g., a Certificate Authority) that is in the Verifier's certificate chain.

The Relying Party might implicitly trust a Verifier, such as in a Verifier/Relying Party combination where the Verifier and Relying Party roles are combined. Or, for a stronger level of security, the Relying Party might require that the Verifier first provide information about itself that the Relying Party can use to assess the trustworthiness of the Verifier before accepting its Attestation Results.

For example, one explicit way for a Relying Party "A" to establish such trust in a Verifier "B", would be for B to first act as an Attester where A acts as a combined Verifier/Relying Party. If A

then accepts B as trustworthy, it can choose to accept B as a Verifier for other Attesters.

As another example, the Relying Party can establish trust in the Verifier by out of band establishment of key material, combined with a protocol like TLS to communicate. There is an assumption that between the establishment of the trusted key material and the creation of the Evidence, that the Verifier has not been compromised.

Similarly, the Relying Party also needs to trust the Relying Party Owner for providing its Appraisal Policy for Attestation Results, and in some scenarios the Relying Party might even require that the Relying Party Owner go through a remote attestation procedure with it before the Relying Party will accept an updated policy. This can be done similarly to how a Relying Party could establish trust in a Verifier as discussed above.

## **7.2. Attester**

In some scenarios, Evidence might contain sensitive information such as Personally Identifiable Information (PII) or system identifiable information. Thus, an Attester must trust entities to which it conveys Evidence, to not reveal sensitive data to unauthorized parties. The Verifier might share this information with other authorized parties, according to a governing policy that address the handling of sensitive information (potentially included in Appraisal Policies for Evidence). In the background-check model, this Evidence may also be revealed to Relying Party(s).

When Evidence contains sensitive information, an Attester typically requires that a Verifier authenticates itself (e.g., at TLS session establishment) and might even request a remote attestation before the Attester sends the sensitive Evidence. This can be done by having the Attester first act as a Verifier/Relying Party, and the Verifier act as its own Attester, as discussed above.

## **7.3. Relying Party Owner**

The Relying Party Owner might also require that the Relying Party first act as an Attester, providing Evidence that the Owner can appraise, before the Owner would give the Relying Party an updated policy that might contain sensitive information. In such a case, authentication or attestation in both directions might be needed, in which case typically one side's Evidence must be considered safe to share with an untrusted entity, in order to bootstrap the sequence. See [Section 11](#) for more discussion.

#### 7.4. Verifier

The Verifier trusts (or more specifically, the Verifier's security policy is written in a way that configures the Verifier to trust) a manufacturer, or the manufacturer's hardware, so as to be able to appraise the trustworthiness of that manufacturer's devices. In a typical solution, a Verifier comes to trust an Attester indirectly by having an Endorser (such as a manufacturer) vouch for the Attester's ability to securely generate Evidence.

In some solutions, a Verifier might be configured to directly trust an Attester by having the Verifier have the Attester's key material (rather than the Endorser's) in its trust anchor store.

Such direct trust must first be established at the time of trust anchor store configuration either by checking with an Endorser at that time, or by conducting a security analysis of the specific device. Having the Attester directly in the trust anchor store narrows the Verifier's trust to only specific devices rather than all devices the Endorser might vouch for, such as all devices manufactured by the same manufacturer in the case that the Endorser is a manufacturer.

Such narrowing is often important since physical possession of a device can also be used to conduct a number of attacks, and so a device in a physically secure environment (such as one's own premises) may be considered trusted whereas devices owned by others would not be. This often results in a desire to either have the owner run their own Endorser that would only endorse devices one owns, or to use Attesters directly in the trust anchor store. When there are many Attesters owned, the use of an Endorser enables better scalability.

That is, a Verifier might appraise the trustworthiness of an application component, operating system component, or service under the assumption that information provided about it by the lower-layer firmware or software is true. A stronger level of assurance of security comes when information can be vouched for by hardware or by ROM code, especially if such hardware is physically resistant to hardware tampering. In most cases, components that have to be vouched for via Endorsements because no Evidence is generated about them are referred to as roots of trust.

The manufacturer having arranged for an Attesting Environment to be provisioned with key material with which to sign Evidence, the Verifier is then provided with some way of verifying the signature on the Evidence. This may be in the form of an appropriate trust anchor, or the Verifier may be provided with a database of public



keys (rather than certificates) or even carefully curated and secured lists of symmetric keys.

The nature of how the Verifier manages to validate the signatures produced by the Attester is critical to the secure operation of a remote attestation system, but is not the subject of standardization within this architecture.

A conveyance protocol that provides authentication and integrity protection can be used to convey Evidence that is otherwise unprotected (e.g., not signed). Appropriate conveyance of unprotected Evidence (e.g., [[I-D.birkholz-rats-uccs](#)]) relies on the following conveyance protocol's protection capabilities:

1. The key material used to authenticate and integrity protect the conveyance channel is trusted by the Verifier to speak for the Attesting Environment(s) that collected Claims about the Target Environment(s).
2. All unprotected Evidence that is conveyed is supplied exclusively by the Attesting Environment that has the key material that protects the conveyance channel
3. The root of trust protects both the conveyance channel key material and the Attesting Environment with equivalent strength protections.

See [Section 12](#) for discussion on security strength.

#### **7.5. Endorser, Reference Value Provider, and Verifier Owner**

In some scenarios, the Endorser, Reference Value Provider, and Verifier Owner may need to trust the Verifier before giving the Endorsement, Reference Values, or appraisal policy to it. This can be done similarly to how a Relying Party might establish trust in a Verifier.

As discussed in [Section 7.3](#), authentication or attestation in both directions might be needed, in which case typically one side's identity or Evidence must be considered safe to share with an untrusted entity, in order to bootstrap the sequence. See [Section 11](#) for more discussion.

### **8. Conceptual Messages**

#### **8.1. Evidence**

Evidence is a set of Claims about the target environment that reveal operational status, health, configuration or construction that have security relevance. Evidence is appraised by a Verifier to establish

its relevance, compliance, and timeliness. Claims need to be collected in a manner that is reliable. Evidence needs to be securely associated with the target environment so that the Verifier cannot be tricked into accepting Claims originating from a different environment (that may be more trustworthy). Evidence also must be protected from man-in-the-middle attackers who may observe, change or misdirect Evidence as it travels from Attester to Verifier. The timeliness of Evidence can be captured using Claims that pinpoint the time or interval when changes in operational status, health, and so forth occur.

## **8.2. Endorsements**

An Endorsement is a secure statement that some entity (e.g., a manufacturer) vouches for the integrity of the device's signing capability. For example, if the signing capability is in hardware, then an Endorsement might be a manufacturer certificate that signs a public key whose corresponding private key is only known inside the device's hardware. Thus, when Evidence and such an Endorsement are used together, an appraisal procedure can be conducted based on appraisal policies that may not be specific to the device instance, but merely specific to the manufacturer providing the Endorsement. For example, an appraisal policy might simply check that devices from a given manufacturer have information matching a set of Reference Values, or an appraisal policy might have a set of more complex logic on how to appraise the validity of information.

However, while an appraisal policy that treats all devices from a given manufacturer the same may be appropriate for some use cases, it would be inappropriate to use such an appraisal policy as the sole means of authorization for use cases that wish to constrain *which* compliant devices are considered authorized for some purpose. For example, an enterprise using remote attestation for Network Endpoint Assessment [[RFC5209](#)] may not wish to let every healthy laptop from the same manufacturer onto the network, but instead only want to let devices that it legally owns onto the network. Thus, an Endorsement may be helpful information in authenticating information about a device, but is not necessarily sufficient to authorize access to resources which may need device-specific information such as a public key for the device or component or user on the device.

## **8.3. Attestation Results**

Attestation Results are the input used by the Relying Party to decide the extent to which it will trust a particular Attester, and allow it to access some data or perform some operation.

Attestation Results may carry a boolean value indicating compliance or non-compliance with a Verifier's appraisal policy, or may carry a

richer set of Claims about the Attester, against which the Relying Party applies its Appraisal Policy for Attestation Results.

The quality of the Attestation Results depends upon the ability of the Verifier to evaluate the Attester. Different Attesters have a different *Strength of Function* [[strengthoffunction](#)], which results in the Attestation Results being qualitatively different in strength.

An Attestation Result that indicates non-compliance can be used by an Attester (in the passport model) or a Relying Party (in the background-check model) to indicate that the Attester should not be treated as authorized and may be in need of remediation. In some cases, it may even indicate that the Evidence itself cannot be authenticated as being correct.

By default, the Relying Party does not believe the Attester to be compliant. Upon receipt of an authentic Attestation Result and given the Appraisal Policy for Attestation Results is satisfied, the Attester is allowed to perform the prescribed actions or access. The simplest such appraisal policy might authorize granting the Attester full access or control over the resources guarded by the Relying Party. A more complex appraisal policy might involve using the information provided in the Attestation Result to compare against expected values, or to apply complex analysis of other information contained in the Attestation Result.

Thus, Attestation Results often need to include detailed information about the Attester, for use by Relying Parties, much like physical passports and drivers licenses include personal information such as name and date of birth. Unlike Evidence, which is often very device- and vendor-specific, Attestation Results can be vendor-neutral, if the Verifier has a way to generate vendor-agnostic information based on the appraisal of vendor-specific information in Evidence. This allows a Relying Party's appraisal policy to be simpler, potentially based on standard ways of expressing the information, while still allowing interoperability with heterogeneous devices.

Finally, whereas Evidence is signed by the device (or indirectly by a manufacturer, if Endorsements are used), Attestation Results are signed by a Verifier, allowing a Relying Party to only need a trust relationship with one entity, rather than a larger set of entities, for purposes of its appraisal policy.

## **9. Claims Encoding Formats**

The following diagram illustrates a relationship to which remote attestation is desired to be added:

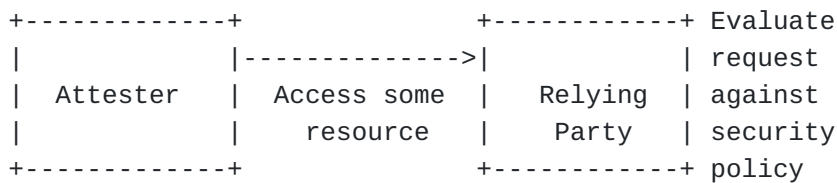


Figure 8: Typical Resource Access

In this diagram, the protocol between Attester and a Relying Party can be any new or existing protocol (e.g., HTTP(S), COAP(S), ROLIE [RFC8322], 802.1x, OPC UA [OPCUA], etc.), depending on the use case.

Typically, such protocols already have mechanisms for passing security information for authentication and authorization purposes. Common formats include JWTs [RFC7519], CWTs [RFC8392], and X.509 certificates.

Retrofitting already deployed protocols with remote attestation requires adding RATS conceptual messages to the existing data flows. This must be done in a way that does not degrade the security properties of the systems involved and should use native extension mechanisms provided by the underlying protocol. For example, if a TLS handshake is to be extended with remote attestation capabilities, attestation Evidence may be embedded in an ad-hoc X.509 certificate extension (e.g., [TCG-DICE]), or into a new TLS Certificate Type (e.g., [I-D.tschofenig-tls-cwt]).

Especially for constrained nodes there is a desire to minimize the amount of parsing code needed in a Relying Party, in order to both minimize footprint and to minimize the attack surface. While it would be possible to embed a CWT inside a JWT, or a JWT inside an X.509 extension, etc., there is a desire to encode the information natively in a format that is already supported by the Relying Party.

This motivates having a common "information model" that describes the set of remote attestation related information in an encoding-agnostic way, and allowing multiple encoding formats (CWT, JWT, X.509, etc.) that encode the same information into the Claims format needed by the Relying Party.

The following diagram illustrates that Evidence and Attestation Results might be expressed via multiple potential encoding formats, so that they can be conveyed by various existing protocols. It also motivates why the Verifier might also be responsible for accepting Evidence that encodes Claims in one format, while issuing Attestation Results that encode Claims in a different format.

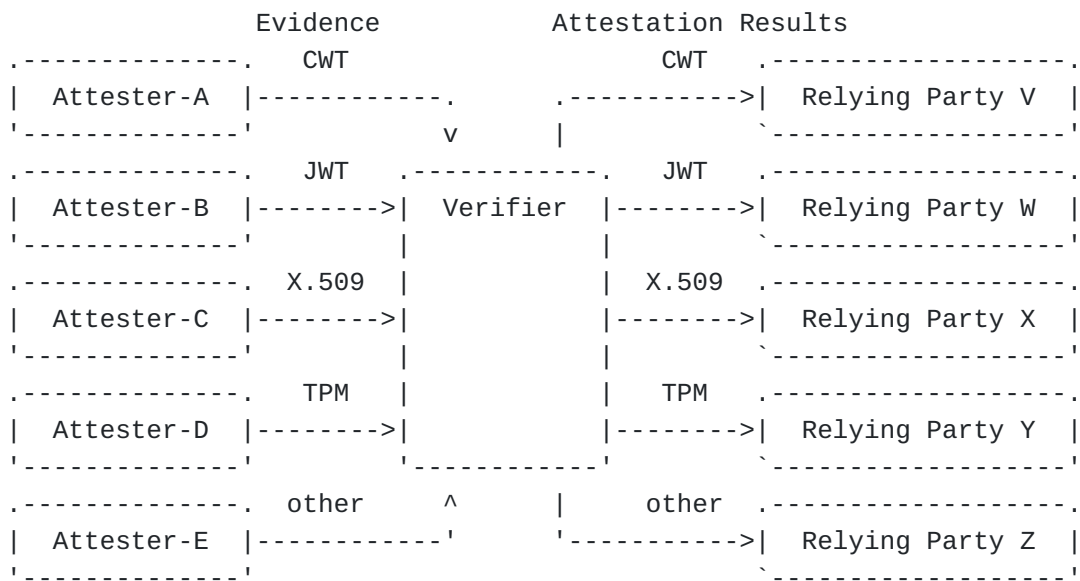


Figure 9: Multiple Attesters and Relying Parties with Different Formats

## 10. Freshness

A Verifier or Relying Party might need to learn the point in time (i.e., the "epoch") an Evidence or Attestation Result has been produced. This is essential in deciding whether the included Claims and their values can be considered fresh, meaning they still reflect the latest state of the Attester, and that any Attestation Result was generated using the latest Appraisal Policy for Evidence.

Freshness is assessed based on the Appraisal Policy for Evidence or Attestation Results that compares the estimated epoch against an "expiry" threshold defined locally to that policy. There is, however, always a race condition possible in that the state of the Attester, and the appraisal policies might change immediately after the Evidence or Attestation Result was generated. The goal is merely to narrow their recentness to something the Verifier (for Evidence) or Relying Party (for Attestation Result) is willing to accept. Some flexibility on the freshness requirement is a key component for enabling caching and reuse of both Evidence and Attestation Results, which is especially valuable in cases where their computation uses a substantial part of the resource budget (e.g., energy in constrained devices).

There are three common approaches for determining the epoch of Evidence or an Attestation Result.

### 10.1. Explicit Timekeeping using Synchronized Clocks

The first approach is to rely on synchronized and trustworthy clocks, and include a signed timestamp (see [[I-D.birkholz-rats-](#)

[tuda](#)]) along with the Claims in the Evidence or Attestation Result. Timestamps can also be added on a per-Claim basis to distinguish the time of generation of Evidence or Attestation Result from the time that a specific Claim was generated. The clock's trustworthiness can generally be established via Endorsements and typically requires additional Claims about the signer's time synchronization mechanism.

In some use cases, however, a trustworthy clock might not be available. For example, in many Trusted Execution Environments (TEEs) today, a clock is only available outside the TEE and so cannot be trusted by the TEE.

## **10.2. Implicit Timekeeping using Nonces**

A second approach places the onus of timekeeping solely on the Verifier (for Evidence) or the Relying Party (for Attestation Results), and might be suitable, for example, in case the Attester does not have a trustworthy clock or time synchronization is otherwise impaired. In this approach, a non-predictable nonce is sent by the appraising entity, and the nonce is then signed and included along with the Claims in the Evidence or Attestation Result. After checking that the sent and received nonces are the same, the appraising entity knows that the Claims were signed after the nonce was generated. This allows associating a "rough" epoch to the Evidence or Attestation Result. In this case the epoch is said to be rough because:

- \*The epoch applies to the entire Claim set instead of a more granular association, and

- \*The time between the creation of Claims and the collection of Claims is indistinguishable.

## **10.3. Implicit Timekeeping using Epoch IDs**

A third approach relies on having epoch identifiers (or "IDs") periodically sent to both the sender and receiver of Evidence or Attestation Results by some "Epoch ID Distributor".

Epoch IDs are different from nonces as they can be used more than once and can even be used by more than one entity at the same time. Epoch IDs are different from timestamps as they do not have to convey information about a point in time, i.e., they are not necessarily monotonically increasing integers.

Like the nonce approach, this allows associating a "rough" epoch without requiring a trustworthy clock or time synchronization in order to generate or appraise the freshness of Evidence or Attestation Results. Only the Epoch ID Distributor requires access to a clock so it can periodically send new epoch IDs.

The most recent epoch ID is included in the produced Evidence or Attestation Results, and the appraising entity can compare the epoch ID in received Evidence or Attestation Results against the latest epoch ID it received from the Epoch ID Distributor to determine if it is within the current epoch. An actual solution also needs to take into account race conditions when transitioning to a new epoch, such as by using a counter signed by the Epoch ID Distributor as the epoch ID, or by including both the current and previous epoch IDs in messages and/or checks, by requiring retries in case of mismatching epoch IDs, or by buffering incoming messages that might be associated with a epoch ID that the receiver has not yet obtained.

More generally, in order to prevent an appraising entity from generating false negatives (e.g., discarding Evidence that is deemed stale even if it is not), the appraising entity should keep an "epoch window" consisting of the most recently received epoch IDs. The depth of such epoch window is directly proportional to the maximum network propagation delay between the first to receive the epoch ID and the last to receive the epoch ID, and it is inversely proportional to the epoch duration. The appraising entity shall compare the epoch ID carried in the received Evidence or Attestation Result with the epoch IDs in its epoch window to find a suitable match.

Whereas the nonce approach typically requires the appraising entity to keep state for each nonce generated, the epoch ID approach minimizes the state kept to be independent of the number of Attesters or Verifiers from which it expects to receive Evidence or Attestation Results, as long as all use the same Epoch ID Distributor.

#### **10.4. Discussion**

Implicit and explicit timekeeping can be combined into hybrid mechanisms. For example, if clocks exist and are considered trustworthy but are not synchronized, a nonce-based exchange may be used to determine the (relative) time offset between the involved peers, followed by any number of timestamp based exchanges.

It is important to note that the actual values in Claims might have been generated long before the Claims are signed. If so, it is the signer's responsibility to ensure that the values are still correct when they are signed. For example, values generated at boot time might have been saved to secure storage until network connectivity is established to the remote Verifier and a nonce is obtained.

A more detailed discussion with examples appears in [Section 16](#).

For a discussion on the security of epoch IDs see [Section 12.3](#).

## 11. Privacy Considerations

The conveyance of Evidence and the resulting Attestation Results reveal a great deal of information about the internal state of a device as well as potentially any users of the device. In many cases, the whole point of attestation procedures is to provide reliable information about the type of the device and the firmware/software that the device is running. This information might be particularly interesting to many attackers. For example, knowing that a device is running a weak version of firmware provides a way to aim attacks better.

Many Claims in Evidence and Attestation Results are potentially Personally Identifying Information (PII) depending on the end-to-end use case of the remote attestation procedure. Remote attestation that goes up to include containers and applications, e.g., a blood pressure monitor, may further reveal details about specific systems or users.

In some cases, an attacker may be able to make inferences about the contents of Evidence from the resulting effects or timing of the processing. For example, an attacker might be able to infer the value of specific Claims if it knew that only certain values were accepted by the Relying Party.

Evidence and Attestation Results are expected to be integrity protected (i.e., either via signing or a secure channel) and optionally might be confidentiality protected via encryption. If confidentiality protection via signing the conceptual messages is omitted or unavailable, the protecting protocols that convey Evidence or Attestation Results are responsible for detailing what kinds of information are disclosed, and to whom they are exposed.

As Evidence might contain sensitive or confidential information, Attesters are responsible for only sending such Evidence to trusted Verifiers. Some Attesters might want a stronger level of assurance of the trustworthiness of a Verifier before sending Evidence to it. In such cases, an Attester can first act as a Relying Party and ask for the Verifier's own Attestation Result, and appraising it just as a Relying Party would appraise an Attestation Result for any other purpose.

Another approach to deal with Evidence is to remove PII from the Evidence while still being able to verify that the Attester is one of a large set. This approach is often called "Direct Anonymous Attestation". See [[CCC-DeepDive](#)] section 6.2 for more discussion.



## **12. Security Considerations**

### **12.1. Attester and Attestation Key Protection**

Implementers need to pay close attention to the protection of the Attester and the manufacturing processes for provisioning attestation key material. If either of these are compromised, intended levels of assurance for RATS are compromised because attackers can forge Evidence or manipulate the Attesting Environment. For example, a Target Environment should not be able to tamper with the Attesting Environment that measures it, by isolating the two environments from each other in some way.

Remote attestation applies to use cases with a range of security requirements, so the protections discussed here range from low to high security where low security may be limited to application or process isolation by the device's operating system, and high security may involve specialized hardware to defend against physical attacks on a chip.

#### **12.1.1. On-Device Attester and Key Protection**

It is assumed that an Attesting Environment is sufficiently isolated from the Target Environment it collects Claims about and that it signs the resulting Claims set with an attestation key, so that the Target Environment cannot forge Evidence about itself. Such an isolated environment might be provided by a process, a dedicated chip, a TEE, a virtual machine, or another secure mode of operation. The Attesting Environment must be protected from unauthorized modification to ensure it behaves correctly. Confidentiality protection of the Attesting Environment's signing key is vital so it cannot be misused to forge Evidence.

In many cases the user or owner of a device that takes on the role of Attester must not be able to modify or extract keys from its Attesting Environments. For example, the owner or user of a mobile phone or FIDO authenticator might not be trusted to use the keys to report Evidence about the environment that protects the keys. An essential value-add provided by RATS is for the Relying Party to be able to trust the Attester even if the user or owner is not trusted.

Measures for a minimally protected system might include process or application isolation provided by a high-level operating system, and restricted access to root or system privileges. In contrast, For really simple single-use devices that don't use a protected mode operating system, like a Bluetooth speaker, the only factual isolation might be the sturdy housing of the device.

Measures for a moderately protected system could include a special restricted operating environment, such as a TEE. In this case, only

security-oriented software has access to the Attester and key material.

Measures for a highly protected system could include specialized hardware that is used to provide protection against chip decapping attacks, power supply and clock glitching, faulting injection and RF and power side channel attacks.

#### **12.1.2. Attestation Key Provisioning Processes**

Attestation key provisioning is the process that occurs in the factory or elsewhere to establish signing key material on the device and the validation key material off the device. Sometimes this is procedure is referred to as personalization or customization.

One way to provision key material is to first generate it external to the device and then copy the key onto the device. In this case, confidentiality protection of the generator, as well as for the path over which the key is provisioned, is necessary. The manufacturer needs to take care to protect corresponding key material with measures appropriate for its value.

Confidentiality protection can be realized via physical provisioning facility security involving no encryption at all. For low-security use cases, this might be simply locking doors and limiting personnel that can enter the facility. For high-security use cases, this might involve a special area of the facility accessible only to select security-trained personnel.

Typically, cryptography is used to enable confidentiality protection. This can result in recursive problems, as the key material used to provision attestation keys must again somehow have been provisioned securely beforehand (requiring an additional level of protection, and so on).

In general, a combination of some physical security measures and some cryptographic measures is used to establish confidentiality protection.

Another way to provision key material is to generate it on the device and export the validation key. If public-key cryptography is being used, then only integrity is necessary. Confidentiality of public keys is not necessary.

In all cases, attestation key provisioning must ensure that only attestation key material that is generated by a valid Endorser is established in Attesters. For many use cases, this will involve physical security at the facility, to prevent unauthorized devices from being manufactured that may be counterfeit or incorrectly configured.

## 12.2. Integrity Protection

Any solution that conveys information used for security purposes, whether such information is in the form of Evidence, Attestation Results, Endorsements, or appraisal policy must support end-to-end integrity protection and replay attack prevention, and often also needs to support additional security properties, including:

- \*end-to-end encryption,
- \*denial of service protection,
- \*authentication,
- \*auditing,
- \*fine grained access controls, and
- \*logging.

[Section 10](#) discusses ways in which freshness can be used in this architecture to protect against replay attacks.

To assess the security provided by a particular appraisal policy, it is important to understand the strength of the root of trust, e.g., whether it is mutable software, or firmware that is read-only after boot, or immutable hardware/ROM.

It is also important that the appraisal policy was itself obtained securely. If an attacker can configure appraisal policies for a Relying Party or for a Verifier, then integrity of the process is compromised.

Security protections in RATS may be applied at different layers, whether by a conveyance protocol, or an information encoding format. This architecture expects conceptual messages (see [Section 8](#)) to be end-to-end protected based on the role interaction context. For example, if an Attester produces Evidence that is relayed through some other entity that doesn't implement the Attester or the intended Verifier roles, then the relaying entity should not expect to have access to the Evidence.

## 12.3. Epoch ID-based Attestation

Epoch IDs, described in [Section 10.3](#), can be tampered with, replayed, dropped, delayed, and reordered by an attacker.

An attacker could be either external or belong to the distribution group, for example, if one of the Attester entities have been compromised.

An attacker who is able to tamper with epoch IDs can potentially lock all the participants in a certain epoch of choice for ever, effectively freezing time. This is problematic since it destroys the ability to ascertain freshness of Evidence and Attestation Results.

To mitigate this threat, the transport should be at least integrity protected and provide origin authentication.

Selective dropping of epoch IDs is equivalent to pinning the victim node to a past epoch. An attacker could drop epoch IDs to only some entities and not others, which will typically result in a denial of service due to the permanent staleness of the Attestation Result or Evidence.

Delaying or reordering epoch IDs is equivalent to manipulating the victim's timeline at will. This ability could be used by a malicious actor (e.g., a compromised router) to mount a confusion attack where, for example, a Verifier is tricked into accepting Evidence coming from a past epoch as fresh, while in the meantime the Attester has been compromised.

Reordering and dropping attacks are mitigated if the transport provides the ability to detect reordering and drop. However, the delay attack described above can't be thwarted in this manner.

### **13. IANA Considerations**

This document does not require any actions by IANA.

### **14. Acknowledgments**

Special thanks go to Joerg Borchert, Nancy Cam-Winget, Jessica Fitzgerald-McKay, Diego Lopez, Laurence Lundblade, Paul Rowe, Hannes Tschofenig, Frank Xia, and David Wooten.

### **15. Notable Contributions**

Thomas Hardjono created initial versions of the terminology section in collaboration with Ned Smith. Eric Voit provided the conceptual separation between Attestation Provision Flows and Attestation Evidence Flows. Monty Wisemen created the content structure of the first three architecture drafts. Carsten Bormann provided many of the motivational building blocks with respect to the Internet Threat Model.

### **16. Appendix A: Time Considerations**

The table below defines a number of relevant events, with an ID that is used in subsequent diagrams. The times of said events might be defined in terms of an absolute clock time, such as the Coordinated

Universal Time timescale, or might be defined relative to some other timestamp or timeticks counter, such as a clock resetting its epoch each time it is powered on.

ID	Event	Explanation of event
VG	Value generated	A value to appear in a Claim was created. In some cases, a value may have technically existed before an Attester became aware of it but the Attester might have no idea how long it has had that value. In such a case, the Value created time is the time at which the Claim containing the copy of the value was created.
NS	Nonce sent	A nonce not predictable to an Attester (recentness & uniqueness) is sent to an Attester.
NR	Nonce relayed	A nonce is relayed to an Attester by another entity.
IR	Epoch ID received	An epoch ID is successfully received and processed by an entity.
EG	Evidence generation	An Attester creates Evidence from collected Claims.
ER	Evidence relayed	A Relying Party relays Evidence to a Verifier.
RG	Result generation	A Verifier appraises Evidence and generates an Attestation Result.
RR	Result relayed	A Relying Party relays an Attestation Result to a Relying Party.
RA	Result appraised	The Relying Party appraises Attestation Results.
OP	Operation performed	The Relying Party performs some operation requested by the Attester via a resource access protocol as depicted in <a href="#">Figure 8</a> , e.g., across a session created earlier at time(RA).
RX	Result expiry	An Attestation Result should no longer be accepted, according to the Verifier that generated it.

Table 1

Using the table above, a number of hypothetical examples of how a solution might be built are illustrated below. This list is not intended to be complete, but is just representative enough to highlight various timing considerations.

All times are relative to the local clocks, indicated by an "\_a" (Attester), "\_v" (Verifier), or "\_r" (Relying Party) suffix.

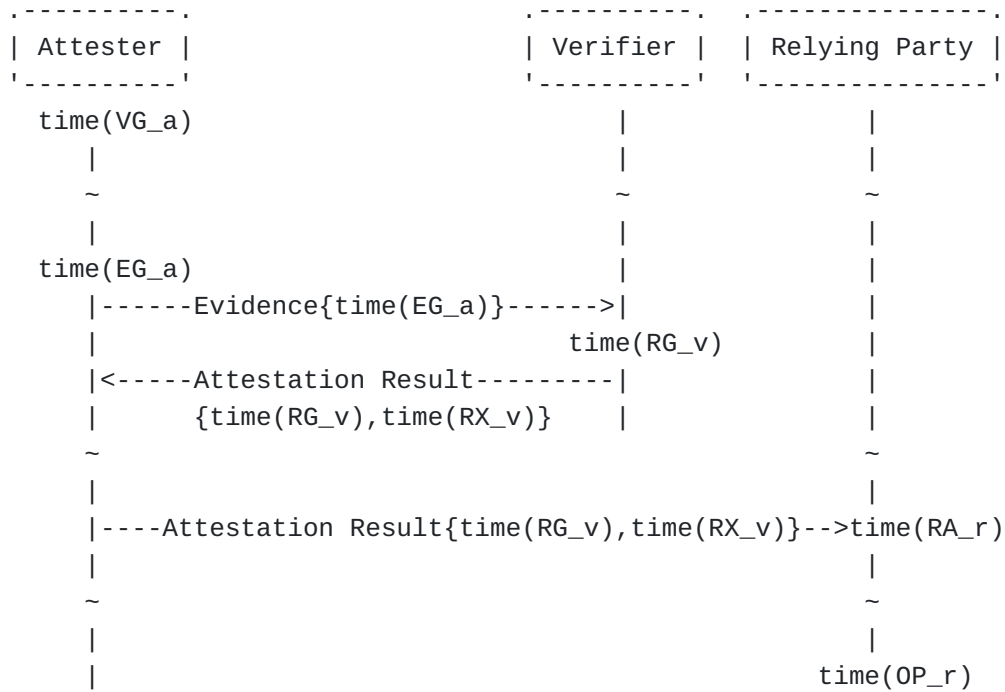
Times with an appended Prime (') indicate a second instance of the same event.

How and if clocks are synchronized depends upon the model.

In the figures below, curly braces indicate containment. For example, the notation Evidence{foo} indicates that 'foo' is contained in the Evidence and is thus covered by its signature.

### 16.1. Example 1: Timestamp-based Passport Model Example

The following example illustrates a hypothetical Passport Model solution that uses timestamps and requires roughly synchronized clocks between the Attester, Verifier, and Relying Party, which depends on using a secure clock synchronization mechanism. As a result, the receiver of a conceptual message containing a timestamp can directly compare it to its own clock and timestamps.



The Verifier can check whether the Evidence is fresh when appraising it at time(RG\_v) by checking  $\text{time(RG\_v)} - \text{time(EG\_a)} < \text{Threshold}$ , where the Verifier's threshold is large enough to account for the maximum permitted clock skew between the Verifier and the Attester.

If time(VG\_a) is also included in the Evidence along with the Claim value generated at that time, and the Verifier decides that it can trust the time(VG\_a) value, the Verifier can also determine whether the Claim value is recent by checking  $\text{time(RG\_v)} - \text{time(VG\_a)} < \text{Threshold}$ . The threshold is decided by the Appraisal Policy for Evidence, and again needs to take into account the maximum permitted clock skew between the Verifier and the Attester.

The Relying Party can check whether the Attestation Result is fresh when appraising it at time(RA\_r) by checking  $\text{time(RA\_r)} - \text{time(RG\_v)} < \text{Threshold}$ , where the Relying Party's threshold is large enough to

account for the maximum permitted clock skew between the Relying Party and the Verifier. The result might then be used for some time (e.g., throughout the lifetime of a connection established at  $\text{time}(\text{RA}_r)$ ). The Relying Party must be careful, however, to not allow continued use beyond the period for which it deems the Attestation Result to remain fresh enough. Thus, it might allow use (at  $\text{time}(\text{OP}_r)$ ) as long as  $\text{time}(\text{OP}_r) - \text{time}(\text{RG}_v) < \text{Threshold}$ . However, if the Attestation Result contains an expiry time  $\text{time}(\text{RX}_v)$  then it could explicitly check  $\text{time}(\text{OP}_r) < \text{time}(\text{RX}_v)$ .

## **16.2. Example 2: Nonce-based Passport Model Example**

The following example illustrates a hypothetical Passport Model solution that uses nonces instead of timestamps. Compared to the timestamp-based example, it requires an extra round trip to retrieve a nonce, and requires that the Verifier and Relying Party track state to remember the nonce for some period of time.

The advantage is that it does not require that any clocks are synchronized. As a result, the receiver of a conceptual message containing a timestamp cannot directly compare it to its own clock or timestamps. Thus we use a suffix ("a" for Attester, "v" for Verifier, and "r" for Relying Party) on the IDs below indicating which clock generated them, since times from different clocks cannot be compared. Only the delta between two events from the sender can be used by the receiver.

Attester	Verifier	Relying Party
time(VG_a)		
~	~	~
<--Nonce1-----time(NS_v)		
time(EG_a)		
---Evidence----->		
{Nonce1, time(EG_a)-time(VG_a)}		
	time(RG_v)	
<--Attestation Result-----		
{time(RX_v)-time(RG_v)}		
~	~	~
<--Nonce2-----time(NS_r)		
time(RR_a)		
--[Attestation Result{time(RX_v)-time(RG_v)}, --> time(RA_r)		
Nonce2, time(RR_a)-time(EG_a)]		
~	~	~
		time(OP_r)

In this example solution, the Verifier can check whether the Evidence is fresh at time(RG\_v) by verifying that  $\text{time(RG\_v)} - \text{time(NS\_v)} < \text{Threshold}$ .

The Verifier cannot, however, simply rely on a Nonce to determine whether the value of a Claim is recent, since the Claim value might have been generated long before the nonce was sent by the Verifier. However, if the Verifier decides that the Attester can be trusted to correctly provide the delta  $\text{time(EG\_a)} - \text{time(VG\_a)}$ , then it can determine recency by checking  $\text{time(RG\_v)} - \text{time(NS\_v)} + \text{time(EG\_a)} - \text{time(VG\_a)} < \text{Threshold}$ .

Similarly if, based on an Attestation Result from a Verifier it trusts, the Relying Party decides that the Attester can be trusted to correctly provide time deltas, then it can determine whether the Attestation Result is fresh by checking  $\text{time(OP\_r)} - \text{time(NS\_r)} + \text{time(RR\_a)} - \text{time(EG\_a)} < \text{Threshold}$ . Although the Nonce2 and  $\text{time(RR\_a)} - \text{time(EG\_a)}$  values cannot be inside the Attestation Result, they might be signed by the Attester such that the Attestation Result vouches for the Attester's signing capability.

The Relying Party must still be careful, however, to not allow continued use beyond the period for which it deems the Attestation Result to remain valid. Thus, if the Attestation Result sends a validity lifetime in terms of  $\text{time(RX\_v)} - \text{time(RG\_v)}$ , then the



Relying Party can check  $\text{time}(\text{OP}_r) - \text{time}(\text{NS}_r) < \text{time}(\text{RX}_v) - \text{time}(\text{RG}_v)$ .

### 16.3. Example 3: Epoch ID-based Passport Model Example

The example in [Figure 10](#) illustrates a hypothetical Passport Model solution that uses epoch IDs instead of nonces or timestamps.

The Epoch ID Distributor broadcasts epoch ID I which starts a new epoch E for a protocol participant upon reception at  $\text{time}(\text{IR})$ .

The Attester generates Evidence incorporating epoch ID I and conveys it to the Verifier.

The Verifier appraises that the received epoch ID I is "fresh" according to the definition provided in [Section 10.3](#) whereby retries are required in the case of mismatching epoch IDs, and generates an Attestation Result. The Attestation Result is conveyed to the Attester.

After the transmission of epoch ID I' a new epoch E' is established when I' is received by each protocol participant. The Attester relays the Attestation Result obtained during epoch E (associated with epoch ID I) to the Relying Party using the epoch ID for the current epoch I'. If the Relying Party had not yet received I', then the Attestation Result would be rejected, but in this example, it is received.

In the illustrated scenario, the epoch ID for relaying an Attestation Result to the Relying Party is current, while a previous epoch ID was used to generate Verifier evaluated evidence. This indicates that at least one epoch transition has occurred, and the Attestation Results may only be as fresh as the previous epoch. If the Relying Party remembers the previous epoch ID I during an epoch window as discussed in [Section 10.3](#), and the message is received during that window, the Attestation Result is accepted as fresh, and otherwise it is rejected as stale.

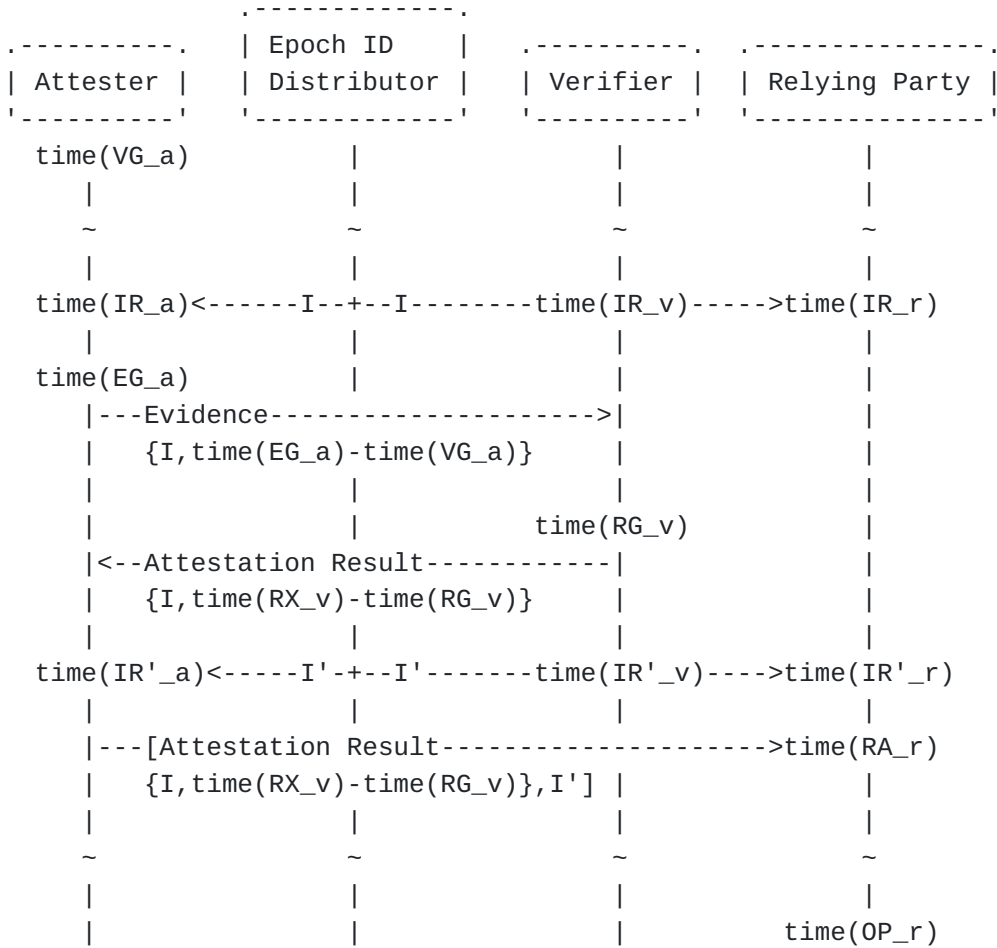
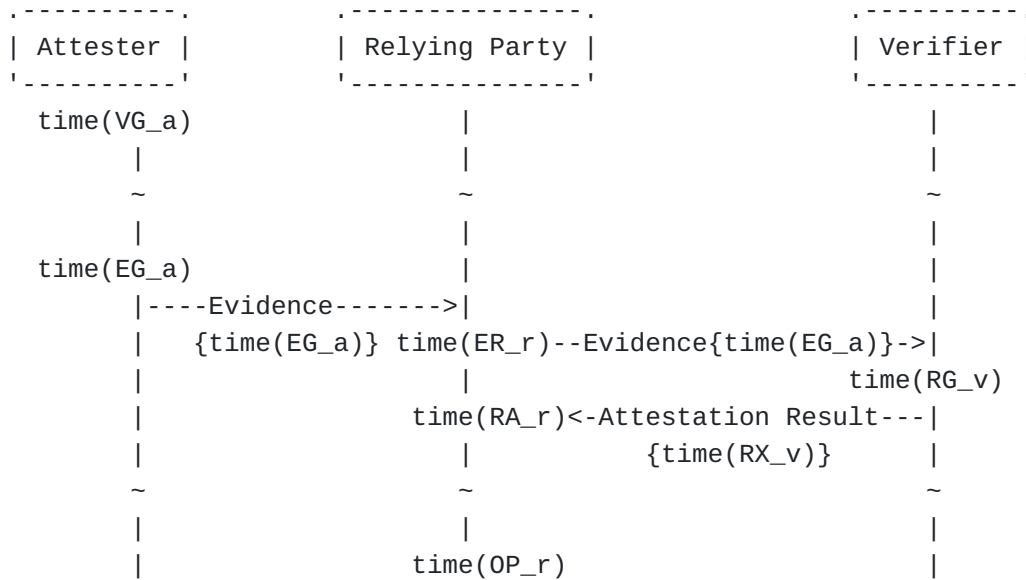


Figure 10: Epoch ID-based Passport Model

#### 16.4. Example 4: Timestamp-based Background-Check Model Example

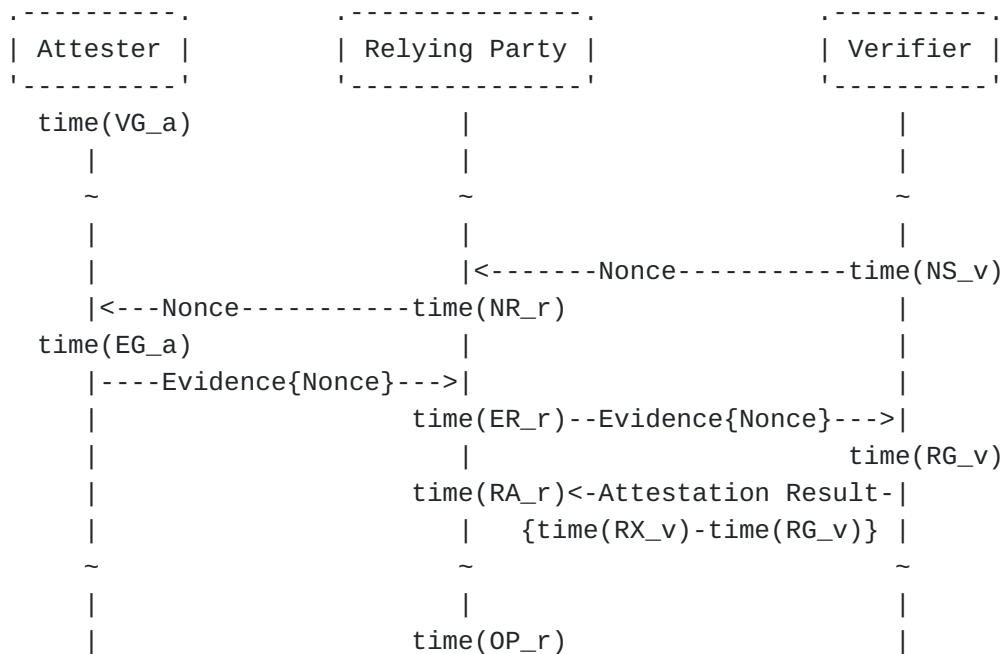
The following example illustrates a hypothetical Background-Check Model solution that uses timestamps and requires roughly synchronized clocks between the Attester, Verifier, and Relying Party.



The time considerations in this example are equivalent to those discussed under Example 1 above.

#### 16.5. Example 5: Nonce-based Background-Check Model Example

The following example illustrates a hypothetical Background-Check Model solution that uses nonces and thus does not require that any clocks are synchronized. In this example solution, a nonce is generated by a Verifier at the request of a Relying Party, when the Relying Party needs to send one to an Attester.



The Verifier can check whether the Evidence is fresh, and whether a Claim value is recent, the same as in Example 2 above.

However, unlike in Example 2, the Relying Party can use the Nonce to determine whether the Attestation Result is fresh, by verifying that  $\text{time}(\text{OP}_r) - \text{time}(\text{NR}_r) < \text{Threshold}$ .

The Relying Party must still be careful, however, to not allow continued use beyond the period for which it deems the Attestation Result to remain valid. Thus, if the Attestation Result sends a validity lifetime in terms of  $\text{time}(\text{RX}_v) - \text{time}(\text{RG}_v)$ , then the Relying Party can check  $\text{time}(\text{OP}_r) - \text{time}(\text{ER}_r) < \text{time}(\text{RX}_v) - \text{time}(\text{RG}_v)$ .

## 17. References

### 17.1. Normative References

- [RFC7519] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, DOI 10.17487/RFC7519, May 2015, <<https://www.rfc-editor.org/info/rfc7519>>.
- [RFC8392] Jones, M., Wahlstroem, E., Erdtman, S., and H. Tschofenig, "CBOR Web Token (CWT)", RFC 8392, DOI 10.17487/RFC8392, May 2018, <<https://www.rfc-editor.org/info/rfc8392>>.

### 17.2. Informative References

- [CCC-DeepDive] Confidential Computing Consortium, "Confidential Computing Deep Dive", n.d., <<https://confidentialcomputing.io/whitepaper-02-latest>>.
- [CTAP] FIDO Alliance, "Client to Authenticator Protocol", n.d., <<https://fidoalliance.org/specs/fido-v2.0-id-20180227/fido-client-to-authenticator-protocol-v2.0-id-20180227.html>>.
- [I-D.birkholz-rats-tuda] Fuchs, A., Birkholz, H., McDonald, I., and C. Bormann, "Time-Based Uni-Directional Attestation", Work in Progress, Internet-Draft, draft-birkholz-rats-tuda-04, 13 January 2021, <<http://www.ietf.org/internet-drafts/draft-birkholz-rats-tuda-04.txt>>.
- [I-D.birkholz-rats-uccs] Birkholz, H., O'Donoghue, J., Cam-Winget, N., and C. Bormann, "A CBOR Tag for Unprotected CWT Claims Sets", Work in Progress, Internet-Draft, draft-birkholz-rats-uccs-02, 2 December 2020, <<http://>>.

[www.ietf.org/internet-drafts/draft-birkholz-rats-uccs-02.txt](http://www.ietf.org/internet-drafts/draft-birkholz-rats-uccs-02.txt)>.

**[I-D.ietf-teep-architecture]**

Pei, M., Tschofenig, H., Thaler, D., and D. Wheeler, "Trusted Execution Environment Provisioning (TEEP) Architecture", Work in Progress, Internet-Draft, draft-ietf-teep-architecture-13, 2 November 2020, <<http://www.ietf.org/internet-drafts/draft-ietf-teep-architecture-13.txt>>.

**[I-D.tschofenig-tls-cwt]**

Tschofenig, H. and M. Brossard, "Using CBOR Web Tokens (CWTs) in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", Work in Progress, Internet-Draft, draft-tschofenig-tls-cwt-02, 13 July 2020, <<http://www.ietf.org/internet-drafts/draft-tschofenig-tls-cwt-02.txt>>.

**[OPCUA]**

OPC Foundation, "OPC Unified Architecture Specification, Part 2: Security Model, Release 1.03", OPC 10000-2 , 25 November 2015, <<https://opcfoundation.org/developer-tools/specifications-unified-architecture/part-2-security-model/>>.

**[RFC4949]**

Shirey, R., "Internet Security Glossary, Version 2", FYI 36, RFC 4949, DOI 10.17487/RFC4949, August 2007, <<https://www.rfc-editor.org/info/rfc4949>>.

**[RFC5209]**

Sangster, P., Khosravi, H., Mani, M., Narayan, K., and J. Tardo, "Network Endpoint Assessment (NEA): Overview and Requirements", RFC 5209, DOI 10.17487/RFC5209, June 2008, <<https://www.rfc-editor.org/info/rfc5209>>.

**[RFC8322]**

Field, J., Banghart, S., and D. Waltermire, "Resource-Oriented Lightweight Information Exchange (ROLIE)", RFC 8322, DOI 10.17487/RFC8322, February 2018, <<https://www.rfc-editor.org/info/rfc8322>>.

**[strengthoffunction]** NISC, "Strength of Function", n.d., <[https://csrc.nist.gov/glossary/term/strength\\_of\\_function](https://csrc.nist.gov/glossary/term/strength_of_function)>.

**[TCG-DICE]**

Trusted Computing Group, "DICE Certificate Profiles", n.d., <[https://trustedcomputinggroup.org/wp-content/uploads/DICE-Certificate-Profiles-r01\\_3june2020-1.pdf](https://trustedcomputinggroup.org/wp-content/uploads/DICE-Certificate-Profiles-r01_3june2020-1.pdf)>.

**[TCGarch]**

Trusted Computing Group, "Trusted Platform Module Library - Part 1: Architecture", 8 November 2019, <[https://trustedcomputinggroup.org/wp-content/uploads/TCG\\_TPM2\\_r1p59\\_Part1\\_Architecture\\_pub.pdf](https://trustedcomputinggroup.org/wp-content/uploads/TCG_TPM2_r1p59_Part1_Architecture_pub.pdf)>.

## **[WebAuthN]**

W3C, "Web Authentication: An API for accessing Public Key Credentials", n.d., <<https://www.w3.org/TR/webauthn-1/>>.

## **Contributors**

Monty Wiseman

Email: [montywiseman32@gmail.com](mailto:montywiseman32@gmail.com)

Liang Xia

Email: [frank.xialiang@huawei.com](mailto:frank.xialiang@huawei.com)

Laurence Lundblade

Email: [lgl@island-resort.com](mailto:lgl@island-resort.com)

Eliot Lear

Email: [ellear@cisco.com](mailto:ellear@cisco.com)

Jessica Fitzgerald-McKay

Sarah C. Helbe

Andrew Guinn

Peter Loscocco

Email: [pete.loscocco@gmail.com](mailto:pete.loscocco@gmail.com)

Eric Voit

Thomas Fossati

Email: [thomas.fossati@arm.com](mailto:thomas.fossati@arm.com)

Paul Rowe

Carsten Bormann

Email: [cabo@tzi.org](mailto:cabo@tzi.org)

Giri Mandyam

Email: [mandyam@qti.qualcomm.com](mailto:mandyam@qti.qualcomm.com)

Kathleen Moriarty

Email: [kathleen.moriarty.ietf@gmail.com](mailto:kathleen.moriarty.ietf@gmail.com)

Guy Fedorkow

Email: [gfedorkow@juniper.net](mailto:gfedorkow@juniper.net)

Simon Frost

Email: [Simon.Frost@arm.com](mailto:Simon.Frost@arm.com)

#### **Authors' Addresses**

Henk Birkholz  
Fraunhofer SIT  
Rheinstrasse 75  
64295 Darmstadt  
Germany

Email: [henk.birkholz@sit.fraunhofer.de](mailto:henk.birkholz@sit.fraunhofer.de)

Dave Thaler  
Microsoft  
United States of America

Email: [dthaler@microsoft.com](mailto:dthaler@microsoft.com)

Michael Richardson  
Sandelman Software Works  
Canada

Email: [mcr+ietf@sandelman.ca](mailto:mcr+ietf@sandelman.ca)

Ned Smith  
Intel Corporation  
United States of America

Email: [ned.smith@intel.com](mailto:ned.smith@intel.com)

Wei Pan  
Huawei Technologies

Email: [william.panwei@huawei.com](mailto:william.panwei@huawei.com)