RATS Working Group
Internet-Draft
Intended status: Standards Track
Expires: 4 October 2026

H. Birkholz
M. Eckel
Fraunhofer SIT
W. Pan
Huawei Technologies
E. Voit
Cisco

2 April 2026

Reference Interaction Models for Remote Attestation Procedures
draft-ietf-rats-reference-interaction-models-17

Abstract

This document describes interaction models for remote attestation procedures (RATS) [RFC9334]. Three conveying mechanisms -- Challenge/Response, Uni-Directional, and Streaming Remote Attestation -- are illustrated and defined. Analogously, a general overview about the information elements typically used by corresponding conveyance protocols are highlighted.

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 4 October 2026.

Copyright Notice

Table of Contents

1. Introduction

Remote ATtestation procedureS (RATS, [RFC9334]) are workflows composed of roles and interactions, in which Verifiers create Attestation Results about the trustworthiness of an Attester's system component characteristics. The Verifier's assessment in the form of Attestation Results is produced based on Endorsements, Reference Values, Appraisal Policies, and Evidence -- trustable and tamper-evident Claims Sets about an Attester's system component characteristics -- generated by an Attester. The roles _Attester_ and _Verifier_, as well as the Conceptual Messages _Evidence_ and _Attestation Results_ are concepts defined by the RATS Architecture [RFC9334]. This document illustrates three main interaction models between various RATS roles, namely Attesters, Verifiers, and Relying Parties that can be used in specific RATS-related specifications. Using Evidence as a prominent example, these three interaction models are:

1. _Challenge/Response Remote Attestation_: A Verifier actively challenges an Attester and receives time-fresh Evidence in response.

2. _Uni-Directional Remote Attestation_: An Attester sends Evidence proactively to a Verifier, often using externally generated freshness indicators.

3. _Streaming Remote Attestation_: A persistent subscription-based model where Evidence is pushed continuously to interested Verifiers, optionally via a broker.

As a basis to describe the interaction models is this document, the example of attestation Evidence conveyance is used to illustrate the usage scenarios. This document aims to:

1. prevent inconsistencies in descriptions of interaction models in other documents, which may occur due to text cloning and evolution over time, and to

2. highlight the exact delta/divergence between the core characteristics captured in this document and variants of these interaction models used in other specifications or solutions.

In summary, this document enables the specification and design of trustworthy and privacy-preserving (see Section Section 9.1) conveyance methods for RATS Conceptual Messages; specifically attestation Evidence conveyed from an Attester to a Verifier. While the exact details for conveyance of other Conceptual Messages is out of scope, the models described in this document may be adapted to apply to the conveyance of other Conceptual Messages, such as Endorsements or Attestation Results, or supplemental messages, such as Epoch Markers [I-D.ietf-rats-epoch-markers] or stand-alone event logs.

2. Terminology

This document uses the following terms defined in Section 4 of [RFC9334]: Attester, Verifier, Relying Party, Conceptual Message, Evidence, Endorsement, Attestation Result, Appraisal Policy, Attesting Environment, Target Environment.

A PKIX Certificate is an X.509v3 certificate as specified by [RFC5280].

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2.1. Disambiguation

"Remote Attestation" is a common expression often associated or connoted with certain properties. In the context of this document, the term "Remote" does not necessarily refer to a remote entity in the scope of network topologies or the Internet. It rather refers to decoupled systems or entities that exchange the Conceptual Message type called Evidence [RFC9334]. This conveyance can also be "Local", if the Verifier role is part of the same entity as the Attester role, e.g., separate system components of the same Composite Device (a single RATS entity), or the Verifier and Relying Party roles are hosted by the same entity, for example in a cryptographic key broker system (see Section 6 of [RFC9334] for more details). If an entity takes on two or more different roles, the functions they provide typically reside in isolated environments that are components of the same entity. Examples of such isolated environments include a Trusted Execution Environment (TEE), Baseboard Management Controllers

(BMCs), as well as other physical or logical protected/isolated/ shielded Computing Environments (e.g., embedded Secure Elements (eSE) or Trusted Platform Modules (TPM)).

2.2. Nonce Disambiguation

The term "nonce" can be used in different security disciplines with different intended security properties. In order to avoid confusion, this document distinguishes the following contexts:

Attestation nonce (freshness handle): A nonce used as a Handle in a remote attestation interaction model to provide replay protection and/or evidence freshness. It is conveyed to an Attesting Environment and is cryptographically bound into Evidence. Depending on deployment, an attestation nonce MAY also serve as a claims-collection control signal (e.g., to invalidate cached claims and trigger re-collection of fresh claims), and it may be provisioned before a reboot and then included with Evidence collected after boot.

* TLS nonce: Random values used within TLS handshakes and key derivation. These are not Handles and are unrelated to Evidence freshness semantics in this document.

* Signature nonce (e.g., ECDSA nonce): Ephemeral randomness used internally by some signature algorithms. These are not protocol-visible Handles and have different security requirements (e.g., uniqueness and secrecy) than attestation nonces.

Unless explicitly stated otherwise, when this document uses the term "nonce" in the context of Handles, it refers to an _attestation nonce_ with the purpose to demonstrate freshness.

2.3. Boot Time Integrity

Boot time integrity refers to the trustworthiness of the platform during its boot sequence, typically covering firmware, BIOS/UEFI, initial bootloaders, and core operating system components up until a stable runtime environment is reached. This may apply equally to physical devices and virtual machines.

2.4. Runtime Integrity

Runtime integrity refers to the ongoing trustworthiness of a platform during normal operation, after the boot sequence completes. It encompasses the integrity of dynamic system state, including loaded applications, kernel modules, and active configurations.

2.5. Boot-to-Runtime Boundary

The exact boundary between boot time and runtime may vary between systems. Generally, it is marked by the handoff from the final bootloader or initial OS kernel to a fully operational environment capable of executing arbitrary applications or services.

3. Scope and Intent

This document:

* outlines common interaction models between RATS roles;

* illustrates interaction models using the conveyance of Evidence about boot-time integrity as an example throughout this document;

* does not exclude the application of those interaction models to runtime integrity or the conveyance of other RATS Conceptual Messages;

* does not cover every detail about Evidence conveyance.

While details regarding Evidence of runtime integrity are not explicitly highlighted, the provided model descriptions serve as a foundation for developing corresponding model extensions. While the interaction models described in this document, including their variants, cover many relevant conveyance models for Conceptual Messages implemented on the Internet, they do not represent an exhaustive list of all possible models.

Procedures, functions, and services needed for a complete semantic binding of the concepts defined in [RFC9334] are not covered in this document. Examples of such procedures include: identity establishment, key distribution and enrollment, time synchronization, and certificate revocation.

Furthermore, any processes and duties beyond conducting remote attestation procedures are out of scope. For example, utilizing the results of a remote attestation procedure generated by the Verifier, such as triggering remediation actions or recovery processes, as well as the remediation actions and recovery processes themselves, are also out of scope.

The interaction models described in this document are meant to serve as a solid foundation and reference for other solution documents within or outside the IETF. Solution documents of any kind can refer to these interaction models to prevent duplicating text and to avoid the risk of subtle discrepancies. Similarly, deviations from the generic model described in this document can be illustrated in solution documents to highlight distinct contributions.

4. Essential Requirements

In order to ensure appropriate conveyance of Evidence, the following requirements MUST be fulfilled:

Integrity: Information provided by an Attester MUST NOT have been altered since it was created. This may be achieved via symmetric cryptography, e.g., using COSE Mac0 as in PSA TF-M profile (Section 5.2 of [RFC9783]), or asymmetric, such as a COSE Sign algorithm like ECDSA.

Authentication: The information provided by the Attester MUST be authentic. To do this, the Attester should authenticate itself to the Verifier.

This can be achieved by digitally signing the Attestation Evidence (i.e., explicit authentication) and conveying that signed Evidence to the Verifier, without requiring additional authentication handshakes beyond those needed for conveyance.

Alternatively, Evidence can be conveyed over a secure channel that provides authentication (see Sections 3 and 4 of [RFC9781]).

5. Normative Prerequisites

In order to ensure Evidence is appropriately conveyed through the interaction models described in this document, the following prerequisites MUST be in place to support their implementation:

Authentication Secret: An Authentication Secret MUST be established before any RATS interaction takes place, and it must be made available exclusively to an Attesting Environment of the Attester.

The Attester MUST protect Claims with this Authentication Secret to prove the authenticity of the Claims included in Evidence. The Authentication Secret MUST be established before RATS take place.

Attester Identity: A statement made by an Endorser about an Attester that affirms the Attester’s distinguishability.

In essence, an Attester Identity can either be explicit (e.g., via a Claim in Evidence or Endorsement) or implicit (e.g., via a signature that matches a trust anchor). Note that distinguishability does not imply uniqueness; for example, a group of Attesters can be identified by an Attester Identity.

The provenance of Evidence SHOULD be distinguishable with respect to the Attesting Environment and MUST be unambiguous with respect to the Attester Identity.

An Attester Identity MAY be an Authentication Secret which is available exclusively to one of the Attesting Environments of the Attester. It could be a unique identity, it could be included in a zero-knowledge proof (ZKP), it could be part of a group signature, or it could be a randomized DAA credential [DAA].

Attestation Evidence Authenticity: Attestation Evidence MUST be authentic.

In order to provide a proof of authenticity, Attestation Evidence can be cryptographically associated with an identity document (e.g., a PKIX certificate or trusted key material, or a randomized DAA credential [DAA]), or could include a correct, unambiguous, and stable reference to an accessible identity document.

For signature validation and appraisal, the Verifier MUST be able to obtain the corresponding verification key material (e.g., a public key or certificate) and the trust anchor(s) needed to establish trust in that material. This may be achieved via provisioning/enrollment, by conveying the identity document together with Evidence, or by providing a stable reference that enables retrieval of the identity document. The concrete distribution and enrollment mechanisms are out of scope of this document.

Authenticity includes the protection of Evidence in a tamper-evident manner (e.g., either by signatures or by protection mechanisms implemented at both ends of a Secure Channel; see Authentication above).

Evidence Freshness: Evidence MUST include an indicator about its freshness that can be understood by a Verifier (See also Section 10 of [RFC9334]). This enables interaction models to support the conveyance of proofs of freshness in a way that is useful to Verifiers and their appraisal procedures.

6. Generic Information Elements

This section describes the essential information elements for the interaction models described in Section 7. These generic information elements may be Conceptual Messages included in the protocol messages or may be added as protocol parameters, depending on the specific solution.

The information elements below are grouped with mandatory elements first, followed by optional elements.

Handle (handle): _mandatory_

An information element provided to the Attester from an external source included in Evidence (or other RATS Conceptual Messages) to determine recentness, freshness, or to protect against replay attacks.

The term Handle encompasses various data types that can be utilized to determine recentness, freshness, or provide replay protection. Examples include attestation nonces (see Section 2.2), which are used to demonstrate freshness (and also protect against replay attacks), and Epoch Markers, which identify distinct periods (Epochs) of freshness [I-D.ietf-rats-epoch-markers]. Handles can also indicate authenticity or attestation Evidence provenance, as only specific RATS roles (e.g., an Attester and a Verifier in a challenge-response interaction) are meant to know a certain handle. In contexts where the concrete Handle type matters, this document uses the more specific term (e.g., "attestation nonce" or "Epoch Marker") in addition to the generic term Handle.

Claims (claims): _mandatory_

Claims are assertions that represent characteristics of an Attester’s Target Environment.

Claims are part of a Conceptual Message and are used, for example, to appraise the integrity of Attesters by Verifiers. The other information elements in this section can be presented as Claims in any type of Conceptual Message.

Collected Claims (collectedClaims): _mandatory_

Collected Claims represent a (sub-)set of Claims created by an Attester.

Collected Claims are gathered based on the Claim Selection. If a

Verifier does not provide a Claim Selection, all available Claims on the Attester are part of the Collected Claims.

Evidence (evidence): _mandatory_

A set of Claims that can include: (a) a list of Attestation Environment IDs, each identifying an Authentication Secret in a single Attesting Environment, (b) the Attester Identity, (c) Claims about the Attester's Target Environment, and (d) a Handle. Attestation Evidence MUST cryptographically bind all of these information elements. Evidence MUST be protected via an Authentication Secret. The Authentication Secret MUST be trusted by the Verifier as authoritatively "speaking for" [lampson06] the Attester.

Attestation Result (attestationResult): _mandatory_

Attestation Results are assertions about the integrity or other characteristics of the appraised Attester.

Attestation Results are created by Verifiers based on appraised Evidence and other input information. Attestation Results are typically conveyed to Relying Parties and can be used to inform access control decisions.

Verifier Inputs ('verInputs') _mandatory_

Appraisal procedures implemented by Verifiers require certain inputs as defined in [RFC9334]: Reference Values, Endorsements, and Appraisal Policy for Evidence. These Conceptual Messages can take various forms. For example, Reference Values that can be expressed via Reference Integrity Measurements (RIM) or Endorsements that can range from trust anchors to assertions cryptographically bound to the public key associated with an Attesting Environment.

Attesting Environment IDs ('attEnvIDs'): _optional_

A statement representing one or more identifiers that MUST be associated with a corresponding Attestation Environment's keys used to protect Claims in Evidence produced by an Attester. Exemplary identifiers include attestation key material (e.g., the public key associated with the private key that is used to produce Evidence), key identifiers, environment names, or individual hardware-based immutable identifiers.

A Verifier MAY use such identifiers (or other key identifiers used

by a given interaction model) to locate the verification key material and related trust anchors required to validate the Evidence signature.

While a Verifier does not necessarily have knowledge about an Attesting Environment's identifiers, each distinguishable Attesting Environment typically has access to a protected capability that includes an Attesting Environment ID. If no Attesting Environment IDs are provided, a local default applies based on the Attester. For example, all Attesting Environments will provide Evidence.

Event Logs (eventLogs): _optional_

Event Logs accompany Claims by providing event trails of security-critical events in a system. The primary purpose of Event Logs is to ensure Claim reproducibility by providing information on how Claims originated.

Claim Selection (claimSelection): _optional_

A (sub-)set of Claims that can be collected and incorporated in Evidence by the Attesting Environments of an Attester.

Claim Selections act as optional filters to specify the exact set of Claims to be included in Evidence. For example, a Verifier could send a Claim Selection, among other elements, to an Attester. An Attester MAY decide whether or not to provide all requested Claims from a Claim Selection to the Verifier. If there is no way to convey a Claim Selection in a remote attestation protocol, a default Claim Selection (e.g., "all") MUST be defined by the Attester and SHOULD be known by the Verifier. In order to select Claims, Claims that can be potentially included in Evidence by an Attesting Environment have to be known by the Verifier.

7. Interaction Models

This document describes three interaction models for Remote Attestation:

1. Challenge/Response (Section 7.1),

2. Unidirectional (Section 7.2), and

3. Streaming (Section 7.3).

Each section starts with a sequence diagram illustrating the interactions between the involved roles: Attester, Verifier and, optionally, a Relying Party. The presented interaction models focus on the conveyance of Evidence and Attestation Results. The same interaction models may apply to the conveyance of other Conceptual Messages (Endorsements, Reference Values, or Appraisal Policies) with other roles involved. However, that is out of scope for the present document.

All interaction models have a strong focus on the use of a Handle to incorporate a proof of freshness and to prevent replay attacks. The way the Handle is processed is the most prominent difference between the three interaction models.

7.1. Challenge/Response Remote Attestation

Note: In the following diagrams, a leading ? indicates that an information element is optional.

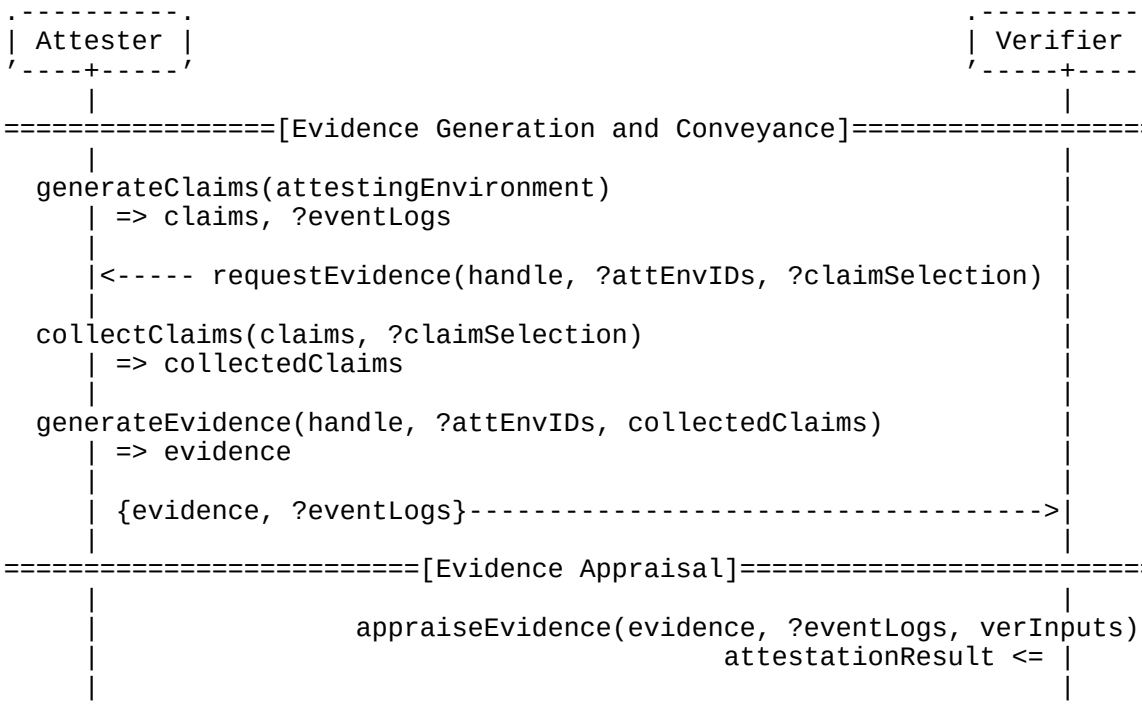


Figure 1: Challenge/Response Remote Attestation

The Attester boots up and thereby produces Claims about its boot state and its operational state during runtime (cf. Section 2), e.g., loaded applications, configurations, and environment variables. Event Logs may accompany the produced Claims and provide an event trail of security-critical events in the system. Claims are produced by all Attesting Environments of an Attester system.

The Challenge/Response remote attestation procedure is typically initiated by the Verifier by sending a remote attestation request to the Attester. Alternative initiation flows, e.g., via an intermediary or through pre-configured requests (e.g., Call-Home procedures or trusted trigger events from Relying Parties), are out of scope for this document. A request includes a Handle, an optional list of Attestation Key IDs, and an optional Claim Selection.

In the Challenge/Response model, the Handle is composed of qualifying data in the form of a practically infeasible-to-guess nonce [RFC4949], such as a cryptographically strong random number. This nonce is typically generated by the Verifier to guarantee Evidence freshness and prevent replay attacks; however, depending on deployment context, it may also be generated by another trusted role, such as a Relying Party. In this interaction model, that nonce is an attestation nonce (freshness handle) as described in Section 2.2.

The list of Attestation Key IDs selects the attestation keys with which the Attester is requested to sign the attestation Evidence. Each selected key is uniquely associated with an Attesting Environment of the Attester. As a result, a single Attestation Key ID identifies a single Attesting Environment. Correspondingly, a particular set of Evidence originating from a particular Attesting Environment in a composite device can be requested via multiple Attestation Key IDs. Methods to acquire Attestation Key IDs or mappings between Attesting Environments to Attestation Key IDs are out of scope of this document.

Nevertheless, in order to validate the Evidence signature, the Verifier needs access to the verification key material corresponding to the selected Attestation Key ID(s), as well as the trust anchors required to establish trust in that key material. This trust material may be provisioned/enrolled out of band, conveyed alongside Evidence, or retrieved via a stable reference to an identity document.

The Attester selects Claims based on the specified Claim Selection, which is defined by the Verifier. The Claim Selection determines the Collected Claims, which may be a subset of all the available Claims. If the Claim Selection is omitted, then all available Claims on the Attester MUST be used to create corresponding Evidence. For example,

when performing a boot integrity evaluation, a Verifier may only request specific claims about the Attester, such as Evidence about the BIOS/UEFI and firmware that the Attester booted up, without including information about all currently running software.

With the Handle, the Attestation Key IDs, and the Collected Claims, the Attester produces signed Evidence. That is, it digitally signs the Handle and the Collected Claims with a cryptographic secret identified by the Attestation Key ID. This is done once per Attesting Environment which is identified by the particular Attestation Key ID. The Attester communicates the signed Evidence as well as all accompanying Event Logs back to the Verifier.

The Claims, the Handle, and the Attester Identity information (i.e., the Authentication Secret) MUST be cryptographically bound to the signature of Evidence. These MAY be presented obfuscated, encrypted, or selectively disclosed (e.g., via hashing) as discussed in Section Section 9.1 or, for example, via the use of [I-D.ietf-spice-sd-cwt] in [RFC9711]. For further reference see, Section Section 9.

Upon receiving the Evidence and Event Logs, the Verifier validates the signature, Attester Identity, and Handle, and then appraises the Claims. Claim appraisal is driven by Policy and takes Reference Values and Endorsements as input. The Verifier outputs Attestation Results. Attestation Results create new Claim Sets about the properties and characteristics of an Attester, which enable Relying Parties to assess an Attester’s trustworthiness.

Note: This diagram illustrates the canonical Challenge/Response interaction between Attester and Verifier. Variants that include a Relying Party (e.g., Passport or Background-Check models) are shown in subsequent sections.

7.1.1. Models and Example Sequences of Challenge/Response Remote Attestation

According to the RATS Architecture, two reference models for Challenge/Response Attestation have been proposed. This section highlights the information flows between the Attester, Verifier, and Relying Party undergoing Remote Attestation Procedure, using these models.

7.1.1.1. Passport Model

The passport model is so named because of its resemblance to how nations issue passports to their citizens. In this model, the attestation sequence is a two-step procedure. In the first step, an Attester conveys Evidence to a Verifier, which appraises the Evidence according to its Appraisal Policy. The Verifier then gives back an Attestation Result to the Attester, which caches it. In the second step, the Attester presents the Attestation Result (and possibly additional Claims/Evidence) to a Relying Party, which appraises this information according to its own Appraisal Policy to establish the trustworthiness of the Attester.

In Challenge/Response interaction models, the Handle (e.g., an attestation nonce) can be generated by different trusted roles depending on deployment context. In the Passport model, the Handle MAY be generated by the Relying Party and conveyed as part of the requestEvidence(...) call; alternatively, it MAY be generated by the Verifier and conveyed to the Attester (e.g., via the Relying Party or other protocol mechanisms).

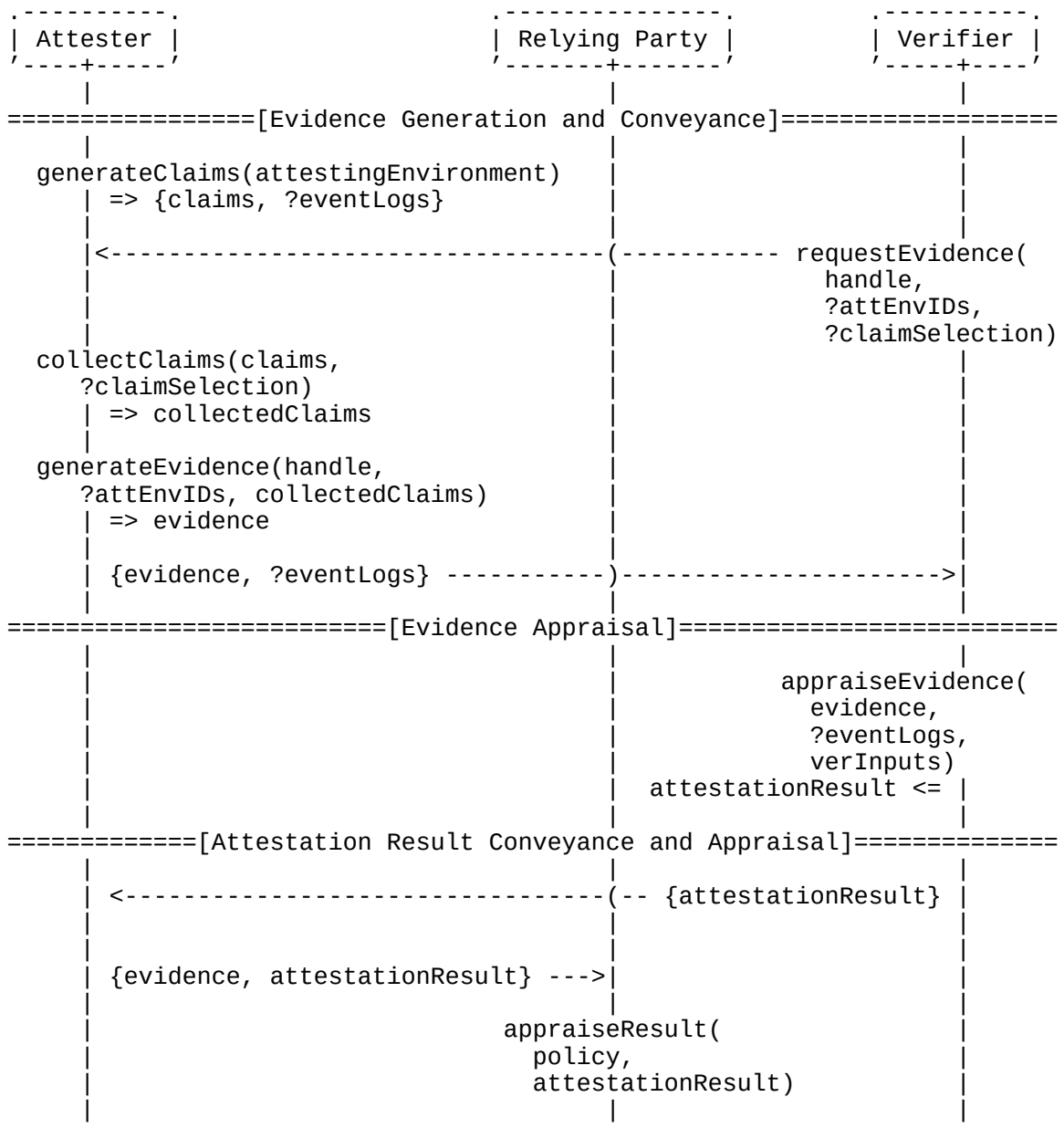


Figure 2: Passport Model for Challenge/Response Remote Attestation

7.1.1.2. Background-Check Model

The background-check model is so named because of the resemblance of how employers and volunteer organizations perform background checks. In this model, the attestation sequence is initiated by a Relying Party. The Attester conveys Evidence to the Relying Party, which does not process its payload, but relays the message and optionally checks its signature against a policed trust anchor store. Upon receiving the Evidence, the Relying Party initiates a session with the Verifier. Once the session is established, it forwards the received Evidence to the Verifier. The Verifier appraises the received Evidence according to its appraisal policy for Evidence and returns a corresponding Attestation Result to the Relying Party. The Relying Party then checks the Attestation Result against its own appraisal policy to conclude attestation.

As in the Passport model, the Handle MAY be generated by the Relying Party (as depicted), or it MAY originate from the Verifier and be conveyed to the Attester by protocol-specific means.

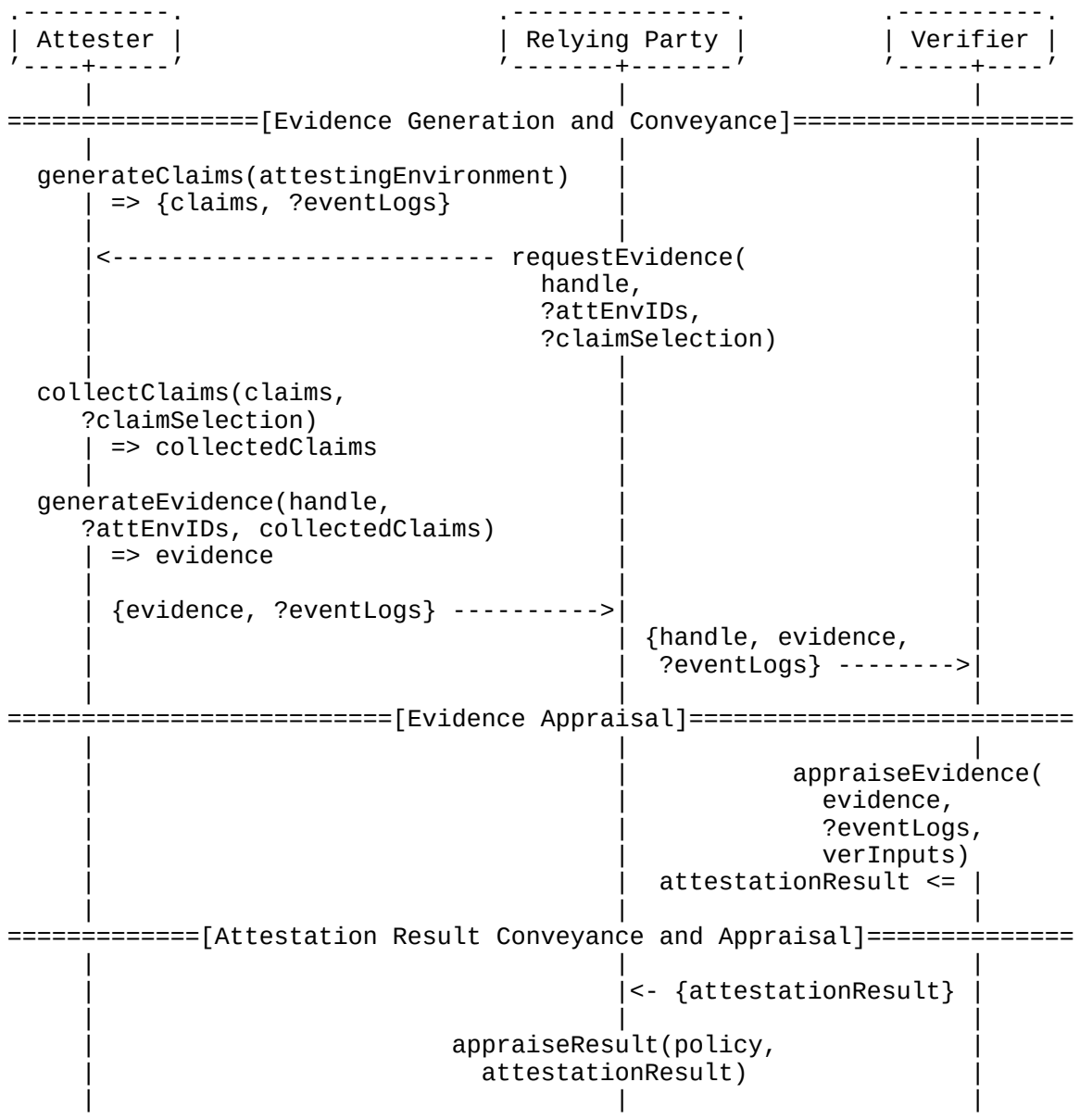
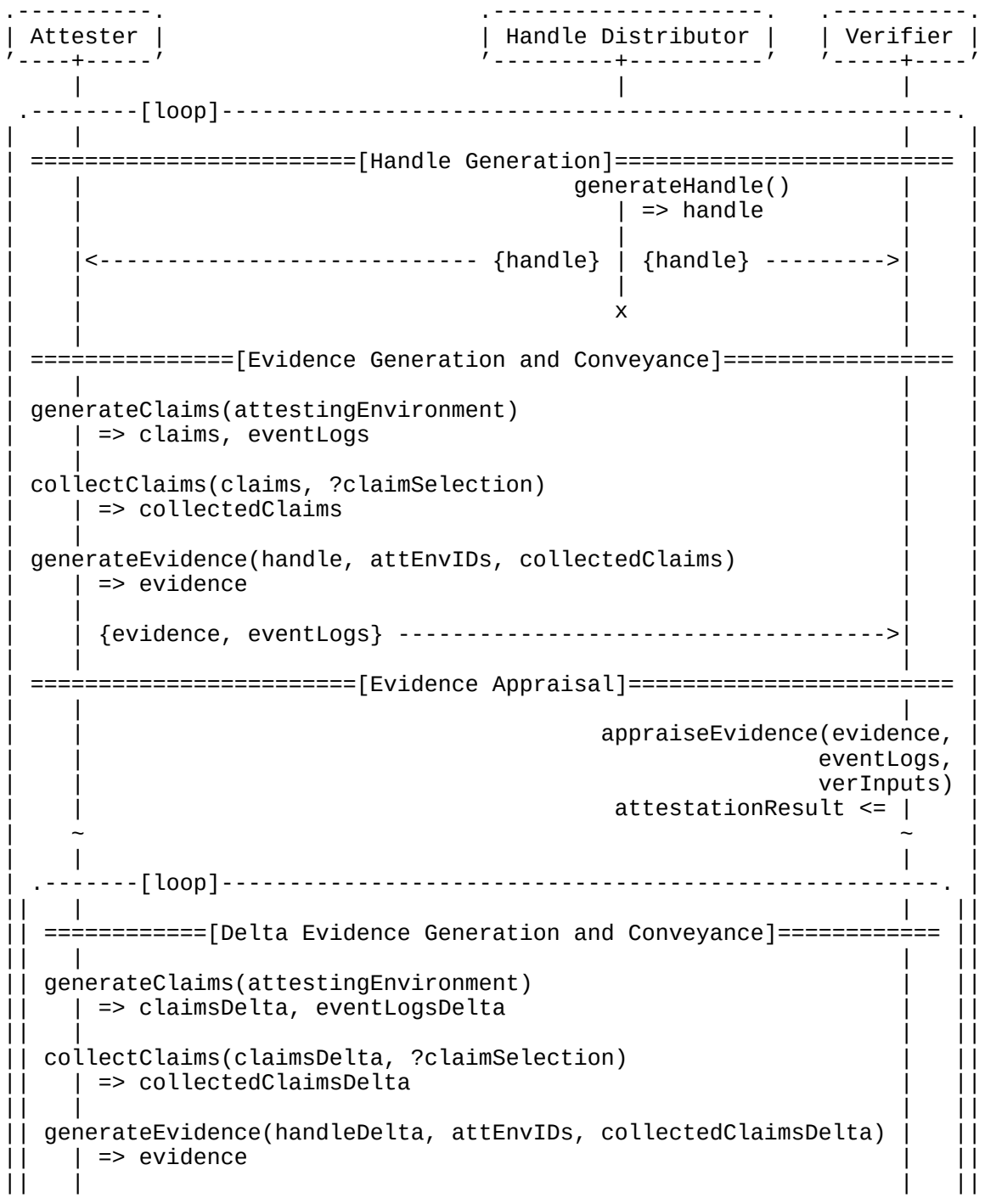


Figure 3: Background-Check Model for Challenge/Response Remote Attestation

7.2. Uni-Directional Remote Attestation


Attester
Handle Distributor
Verifier
loop
Handle Generation
generateHandle()
=> handle
{handle}
{handle}
Evidence Generation and Conveyance
generateClaims(attestingEnvironment)
=> claims, eventLogs
collectClaims(claims, ?claimSelection)
=> collectedClaims
generateEvidence(handle, attEnvIDs, collectedClaims)
=> evidence
{evidence, eventLogs}
Evidence Appraisal
appraiseEvidence(evidence,
eventLogs,
verInputs)
attestationResult <=
loop
Delta Evidence Generation and Conveyance
generateClaims(attestingEnvironment)
=> claimsDelta, eventLogsDelta
collectClaims(claimsDelta, ?claimSelection)
=> collectedClaimsDelta
generateEvidence(handleDelta, attEnvIDs, collectedClaimsDelta)
=> evidence

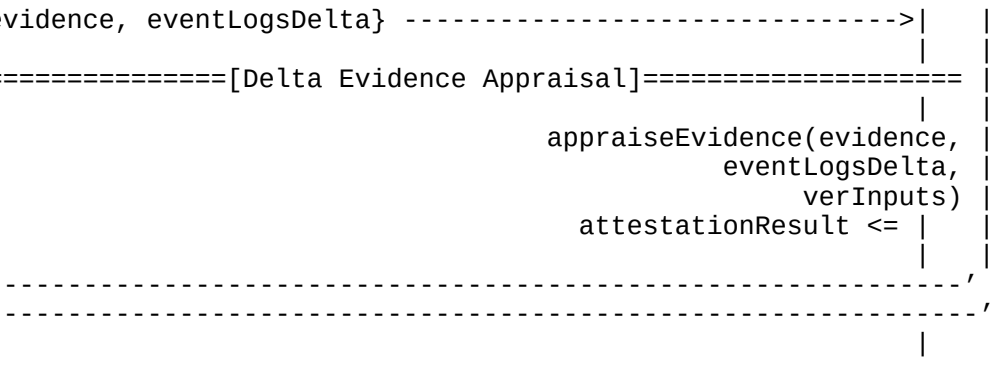


Figure 4: Uni-Directional Remote Attestation

Uni-Directional Remote Attestation procedures can be initiated both by the Attester and by the Verifier. Initiation by the Attester can result in unsolicited pushes of Evidence to the Verifier. Initiation by the Verifier always results in solicited pushes to the Verifier.

The Uni-Directional model uses the same information elements as the Challenge/Response model. In the sequence diagram above, the Attester initiates the conveyance of Evidence (comparable with a RESTful POST operation). While a request of Evidence from the Verifier would result in a sequence diagram more similar to the Challenge/Response model (comparable with a RESTful GET operation). The specific manner how Handles are generated is not in scope of this document. One example of a specific handle representation is [I-D.ietf-rats-epoch-markers].

In the Uni-Directional model, Evidence may be conveyed as full Evidence or as delta Evidence. Regardless of whether delta Evidence is used, each Evidence conveyance is bound to a current Handle. In the diagrams, handleDelta denotes the Handle associated with the delta Evidence conveyance, which may be the same as the previous Handle or a newly issued Handle (e.g., for a new Epoch or freshness window). The concrete rules for when a Handle is rotated (resulting in a new handleDelta) are deployment- and protocol-specific and are out of scope of this document.

Note: If a Verifier does not receive the current Handle (e.g., due to propagation loss or delay), it may be unable to determine Evidence freshness for the corresponding Epoch. In such cases, Evidence appraisal cannot be completed until re-synchronization occurs. Error signaling and re-synchronization mechanisms are protocol-specific and out of scope of this document.

In the Uni-Directional model, handles are composed of cryptographically signed trusted timestamps as shown in [I-D.birkholz-rats-tuda], potentially including other qualifying data. The Handles are created by an external trusted third party (TTP) -- the Handle Distributor -- which includes a trustworthy source of time, and takes on the role of a Time Stamping Authority (TSA, as initially defined in [RFC3161]). Timestamps created from local clocks (absolute clocks using a global timescale, as well as relative clocks, such as tick-counters) of Attesters and Verifiers MUST be cryptographically bound to fresh Handles received from the Handle Distributor. This binding provides a proof of synchronization that MUST be included in all produced Evidence. This model provides proof that Evidence generation happened after the Handle generation phase. The Verifier can always determine whether the received Evidence includes a fresh Handle, i.e., one corresponding to the current Epoch as identified by an Epoch Marker handle.

7.2.1. Handle Lifecycle and Propagation Delays

The term "uni-directional" refers to individual conveyance channels: one from the Handle Distributor to the Attester, and one from the Attester to the Verifier. Together, they establish an attestation loop without requiring request/response exchanges. This model does not assume that Verifiers broadcast Handles, as such a setup would require Verifiers to take on the Handle Distributor role and undermine the separation of duties between these roles.

The lifecycle of a handle is a critical aspect of ensuring the freshness and validity of attestation Evidence. When a new handle is generated by the Handle Distributor, it effectively supersedes the previous handle. However, due to network latencies and propagation delays, there may be a period during which both the old and new handles are in circulation. This "grey zone" can potentially lead to situations where Evidence may be associated with an outdated handle yet still appear to be valid.

To manage this complexity, it is essential to define a clear policy for handle validity and expiration:

* _Handle Expiry_: Each handle should have a well-defined expiration time, after which it is considered invalid. This expiry must account for expected propagation delays and be clearly communicated to all entities in the attestation process.

* _Synchronization Checks_: Implement periodic synchronization checks between the Handle Distributor and both Attesters and Verifiers to ensure that handles are updated consistently across all participants. For example, in TUDA [I-D.birkholz-rats-tuda],

synchronization checks can be realized by cryptographically binding local timestamps from Attesters and Verifiers to fresh Epoch Handles issued by a trusted Time Stamp Authority (TSA), thereby proving that both entities share a consistent notion of time.

* _Grace Periods_: Define grace periods during which a newly issued handle starts being accepted, and the old handle stops being valid. This period should be long enough to account for the maximum expected propagation delay across the network.

Implementing these measures will help mitigate the risks associated with the handle lifecycle, particularly in environments where propagation delays are significant. This careful management ensures that the integrity and trustworthiness of the attestation process are maintained.

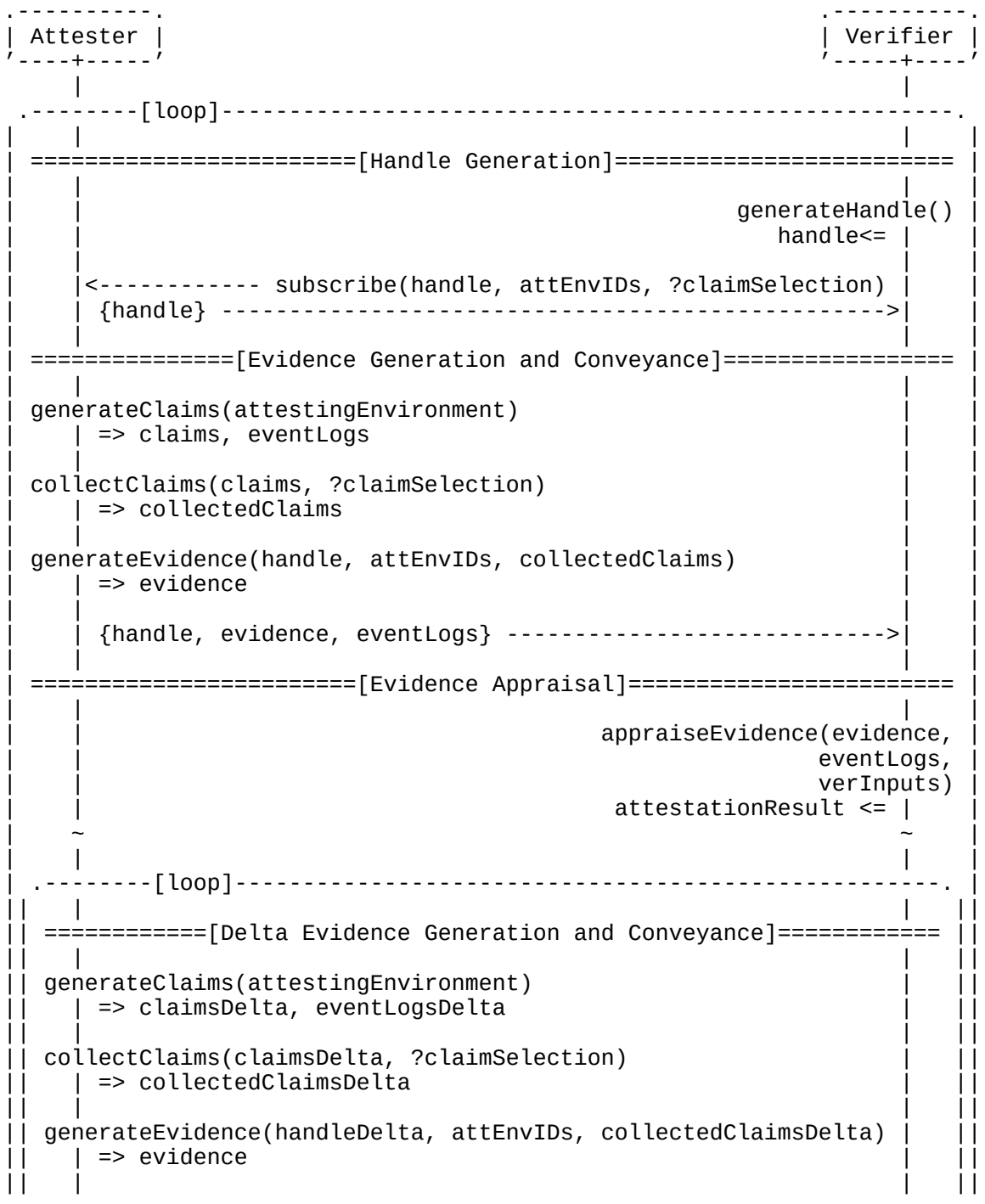
While periodically pushing Evidence to the Verifier, the Attester only needs to generate and convey updates since the previous conveyance. These updates, referred to as "delta" in the sequence diagrams, are not limited to net changes of Claim values. They MUST include all state changes detected since the previous conveyance, even if values later revert to their prior state. For example, if an Attester goes through a sleep or hibernation cycle and a Claim value changes and then reverts, both transitions MUST be reported to the Verifier as soon as possible after resuming operation.

Effectively, the Uni-Directional model allows for a series of Evidence to be pushed to multiple Verifiers simultaneously. Methods to detect excessive time drift that would mandate a fresh Handle to be received by the Handle Distributor as well as timing of Handle distribution are out-of-scope of this document.

7.3. Streaming Remote Attestation

Streaming Remote Attestation serves as the foundational concept for both the publish-subscribe pattern [DesignPatterns] and the observer pattern [ISIS]. It entails establishing subscription state to enable continuous remote attestation. In the publish-subscribe pattern, a central broker is used to distribute messages between publishers and subscribers. The broker receives messages from publishers and forwards them to the appropriate subscribers. In the observer pattern, however, observers are connected directly to target resources, without a broker (oub/sub only).

7.3.1. Streaming Remote Attestation without a Broker


Attester
Verifier
loop
Handle Generation
generateHandle()
handle<=
subscribe(handle, attEnvIDs, ?claimSelection)
{handle}
Evidence Generation and Conveyance
generateClaims(attestingEnvironment)
=> claims, eventLogs
collectClaims(claims, ?claimSelection)
=> collectedClaims
generateEvidence(handle, attEnvIDs, collectedClaims)
=> evidence
{handle, evidence, eventLogs}
Evidence Appraisal
appraiseEvidence(evidence,
eventLogs,
verInputs)
attestationResult <=
loop
Delta Evidence Generation and Conveyance
generateClaims(attestingEnvironment)
=> claimsDelta, eventLogsDelta
collectClaims(claimsDelta, ?claimSelection)
=> collectedClaimsDelta
generateEvidence(handleDelta, attEnvIDs, collectedClaimsDelta)
=> evidence

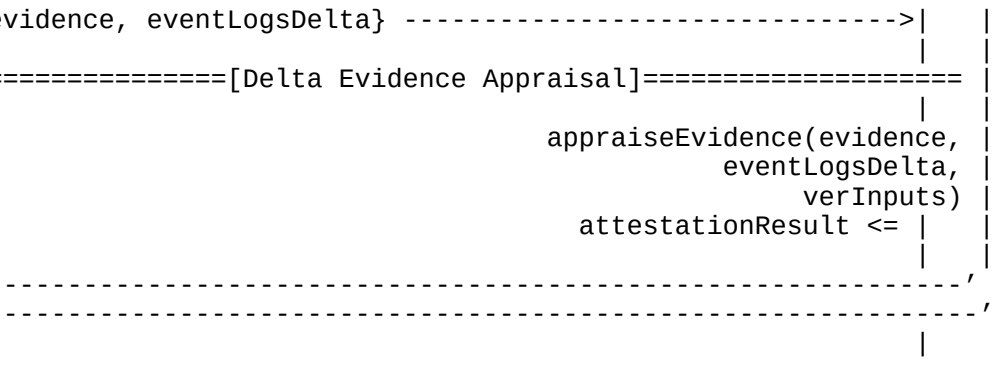


Figure 5: Streaming Remote Attestation without a Broker

In the observer pattern, an observer establishes a direct connection to the observed resources through a subscription mechanism, which is designed specifically for conveying conceptual messages for remote attestation purposes. This mechanism not only facilitates the initial subscription request but also actively maintains the state of the subscription, ensuring that any changes in the observed resources are consistently communicated to the observer. It handles the complexities of managing these connections, including the maintenance of pertinent information about the observer's preferences and security requirements, ensuring that the transmission of attestation data remains both secure and relevant to the observer' s specific context.

Setting up subscription state between a Verifier and an Attester is conducted via a subscribe operation. The subscribe operation is used to convey Handles required for Evidence generation. Effectively, this allows for a series of Evidence to be pushed to a Verifier, similar to the Uni-Directional model. In the observer pattern, the Handle Distributor role is optional. While the model is typically used for direct, bi-lateral subscription relationships where the Verifier generates and provides Handles directly, it is also possible to include the trusted third party that is a Handle Distributor. A Handle Distributor independently manages the generation and distribution of Handles to other RATS roles. As a result, scenarios involving more than bi-lateral interactions are enabled. However, in its basic form, the model assumes direct interaction between an Attester and a Verifier, where the Handle generation is a responsibility taken on by the Verifier roles.

Handles provided by a specific subscribing Verifier MUST be used in Evidence generation for that specific Verifier. The streaming model without a Broker uses the same information elements as the Challenge/ Response and the Uni-Directional model. Methods to detect excessive

time drift that would render Handles stale and mandate a fresh Handles to be conveyed via another subscribe operation are out-of-scope of this document.

Delta Evidence is produced with respect to the previous conveyance, but it is still bound to a (potentially updated) current Handle (e.g., handleDelta). If a new Handle is required, it is conveyed by a subsequent subscribe operation.

If Evidence or delta Evidence repeatedly fails to verify, a Verifier may terminate the subscription. The detailed mechanisms for unsubscribe and re-subscribe are protocol-specific and out of scope for this document; for example, subscription lifecycle management is defined in [I-D.ietf-rats-network-device-subscription].

7.3.2. Streaming Remote Attestation with a Broker

This model includes a Broker to facilitate the distribution of messages between RATS roles, such as Attesters and Verifiers. The Broker is a trusted third party and acts as an intermediary that ensures messages are securely and reliably conveyed between involved RATS roles. The publish-subscribe messaging pattern is widely used for communication in different areas. An example for a publish-subscribe model with a Broker is the Message Queuing Telemetry Transport [MQTT]. Unlike the _Streaming Remote Attestation without a Broker_ interaction model, Attesters are not required to be aware of corresponding Verifiers. In scenarios with large numbers of Attesters and Verifiers, the publish-subscribe pattern may reduce interdependencies and improve scalability.

With publish-subscribe, clients typically _connect_ to (or _register_ with) a publish-subscribe server (PubSub server or Broker). Clients may _publish_ data in the form of a _message_ under a certain _topic_. _Subscribers_ to that topic get _notified_ whenever a message arrives under a topic, and the appropriate message is forwarded to them. Depending on particular publish-subscribe models and implementations, involved roles can be publishers, subscribers or both.

The Broker and Handle Distributor are considered to be trusted third parties (TTPs) for all other participating roles, including Attesters and Verifiers (see also Section 9). All roles must establish a trust relationship with the Broker and Handle Distributor, as those are responsible for the secure and reliable dissemination of critical protocol information, such as Handles and Attestation Results.

Ensuring the security of these trusted third parties is vital, as any compromise could undermine the entire remote attestation procedure. Therefore, the deployment of Brokers and Handle Distributors requires stringent security measures to protect against unauthorized access and to ensure that they operate as trustworthy facilitators within the remote attestation framework.

In the following sections, the interaction models _Challenge/Response Remote Attestation over Publish-Subscribe_ and _Uni-Directional Remote Attestation over Publish-Subscribe_ are described. There are different phases that both models go through:

1. Handle Generation

2. Evidence Generation and Conveyance

3. Evidence Appraisal

4. Attestation Result Generation

The models only differ in the handle generation phase. From a remote attestations procedure's point of view Evidence Generation, Conveyance, and Appraisal, as well as Attestation Result Generation are identical in both models.

7.3.2.1. Handle Generation for Challenge/Response Remote Attestation over Publish-Subscribe

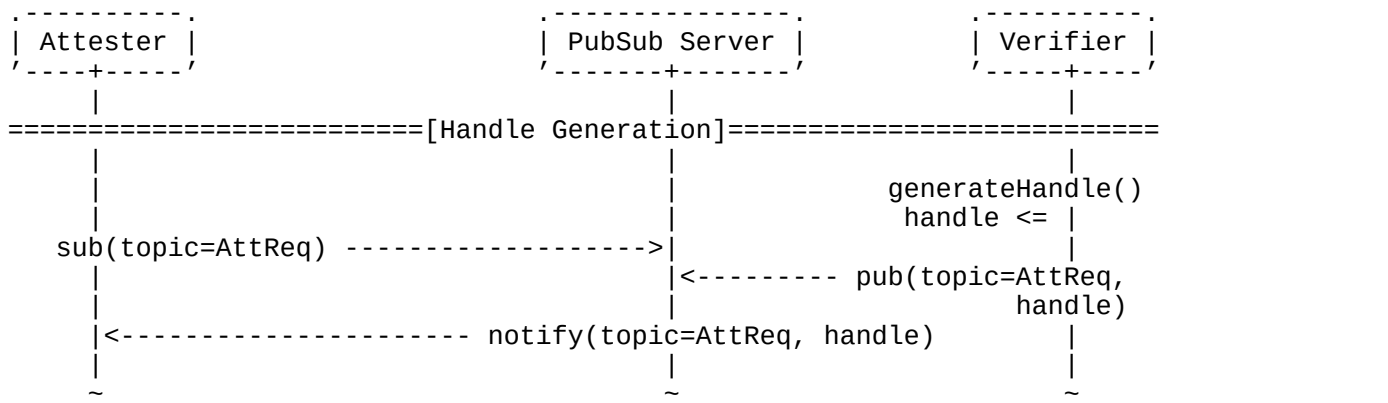


Figure 6: Handle Generation for Challenge/Response Remote Attestation over Publish-Subscribe

The _Challenge/Response Remote Attestation over Publish-Subscribe_ interaction model uses the same information elements as the _Challenge/Response Remote Attestation_ interaction model. Handles are generated by the Verifier on a per-request basis. In the sequence diagram above, the Verifier initiates an attestation by generating a new handle and publishing it to the "AttReq" (= Attestation Request) topic on the PubSub server. The PubSub server then forwards this handle to the Attester by notifying it. This mechanism ensures that each handle is uniquely associated with a specific attestation request, thereby enhancing security by preventing replay attacks.

7.3.2.2. Handle Generation for Uni-Directional Remote Attestation over Publish-Subscribe

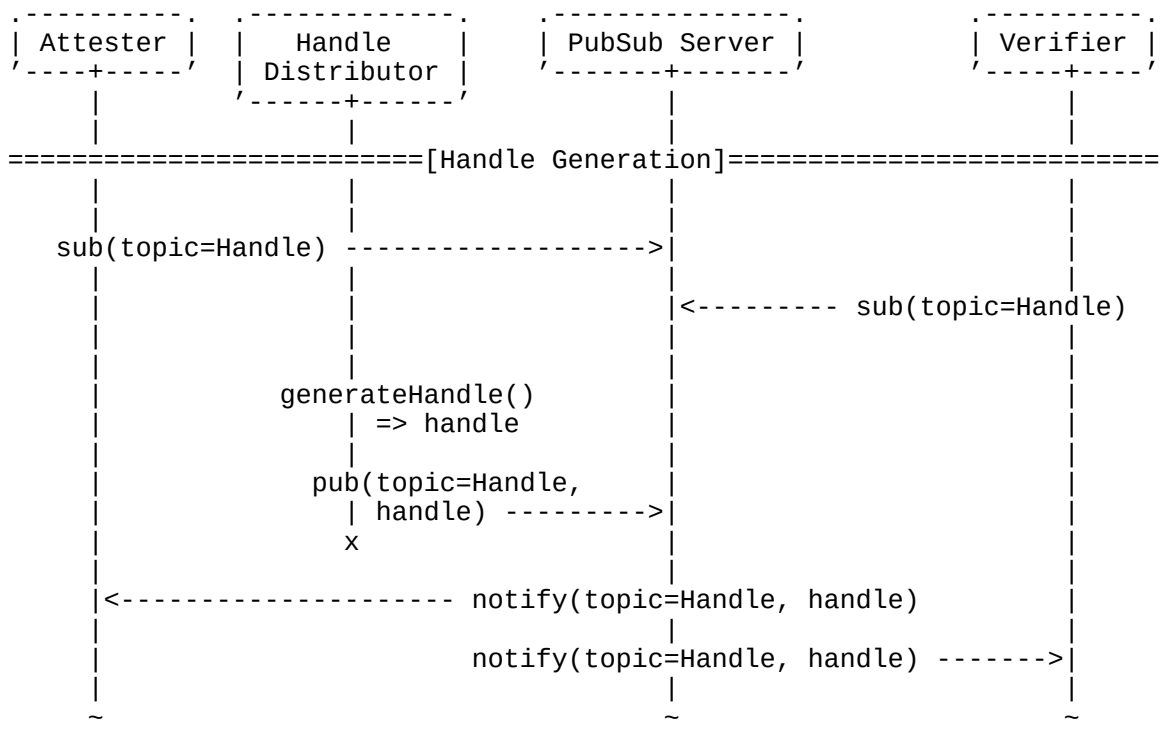


Figure 7: Handle Generation for Uni-Directional Remote Attestation over Publish-Subscribe

Handles are created by a trusted third party, the Handle Distributor (see Section 9). In the sequence diagram above, both an Attester and a Verifier subscribe to the topic "Handle" on the PubSub server. When the Handle Distributor generates and publishes a Handle to the

"Handle" topic on the PubSub server, the PubSub server notifies the subscribers, Attester and Verifier, and forwards ("notify") the Handle to them during Handle Generation.

7.3.2.3. Evidence Generation and Appraisal

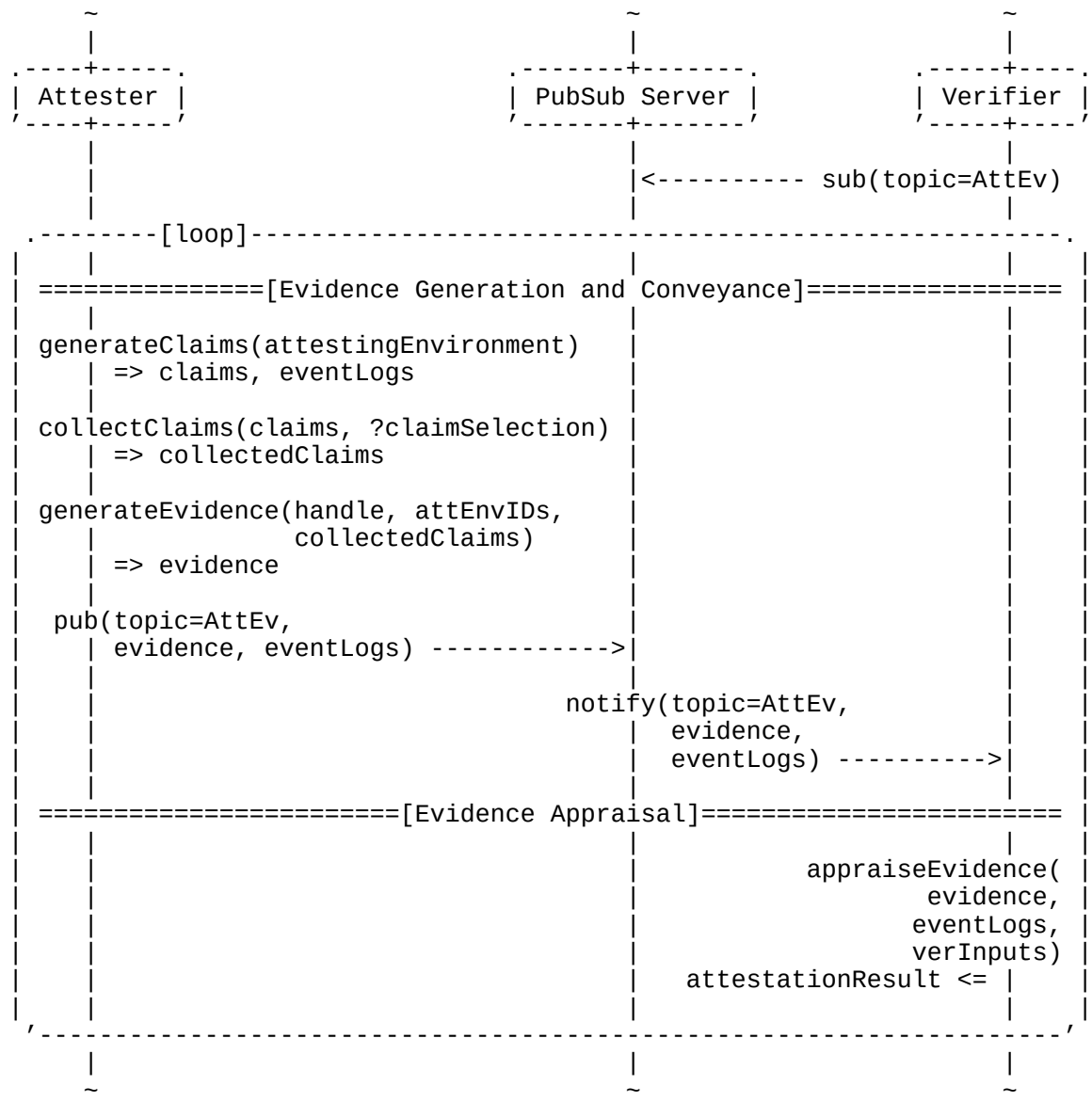


Figure 8: Evidence Generation and Appraisal for Remote Attestation over Publish-Subscribe

Exactly as in the Challenge/Response and Uni-Directional interaction models, there is an Evidence Generation-Appraisal loop, in which the Attester generates Evidence and the Verifier appraises it. In the Publish-Subscribe model above, the Attester publishes Evidence to the topic "AttEv" (= Attestation Evidence) on the PubSub server, to which a Verifier subscribed before. The PubSub server notifies Verifiers, accordingly, by forwarding the attestation Evidence. Although the above diagram depicts only full attestation Evidence and Event Logs, later attestations may use "deltas' for Evidence and Event Logs. The definition of delta Evidence is provided in Section 7.2.1.

Verifiers appraise the Evidence and publish the Attestation Result to topic "AttRes" (= Attestation Result) on the PubSub server.

7.3.2.4. Attestation Result Generation

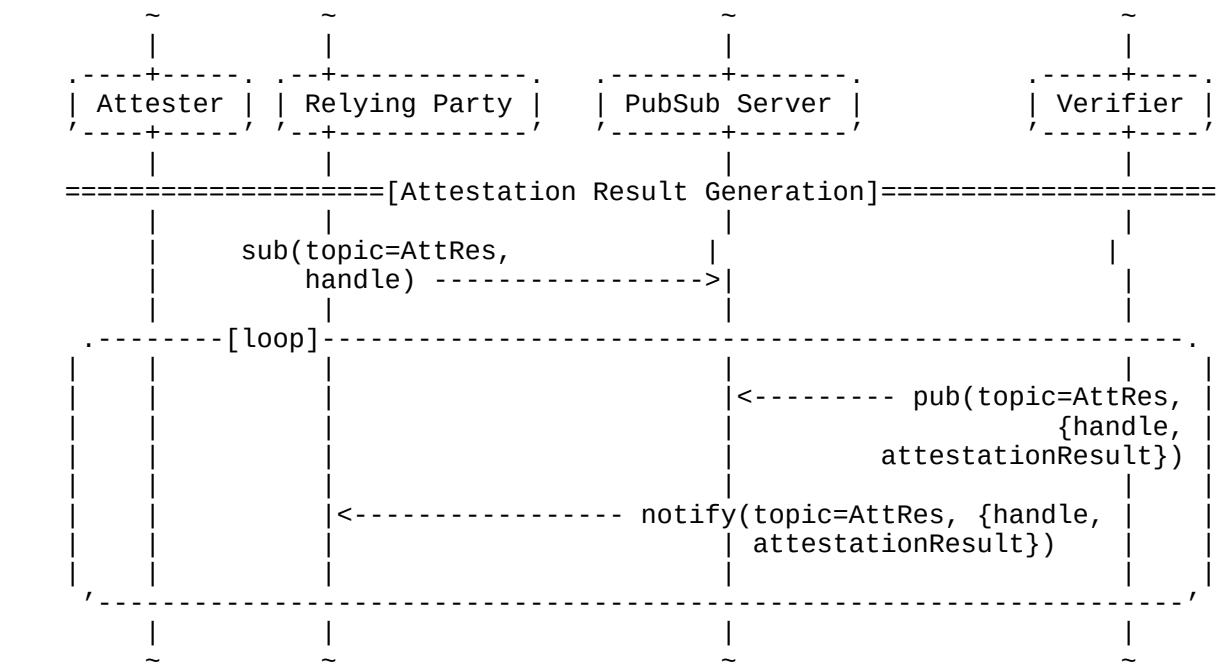


Figure 9: Attestation Result Generation for Remote Attestation over Publish- Subscribe

Attestation Result Generation is the same for both publish-subscribe models,_Challenge/Response Remote Attestation over Publish-Subscribe_ and _Uni-Directional Remote Attestation over Publish-Subscribe_. Relying Parties subscribe to topic AttRes (= Attestation Result) on the PubSub server. The PubSub server forwards Attestation Results to the Relying Parties as soon as they are published to topic AttRes.

Attestation Results conveyed to Relying Parties MUST be bound to the Handle used for the corresponding Evidence appraisal (to prevent mix-up and replay across Handles/Epochs, and to enable correlation). This binding can be achieved by including the Handle in the Attestation Result itself, or by cryptographically binding the Handle to the published message (e.g., signing a structure that includes both Handle and Attestation Result).

8. Implementation Status

Note to RFC Editor: Please remove this section as well as references to [BCP205] before AUTH48.

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [BCP205]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to [BCP205], "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

8.1. Implementer

The open-source implementation was initiated and is maintained by the Fraunhofer Institute for Secure Information Technology SIT.

8.2. Implementation Name

The open-source implementation is named "CHAllenge-Response based Remote Attestation" or in short: CHARRA.

8.3. Implementation URL

The open-source implementation project resource can be located via: https://github.com/fraunhofer-sit/charra (https://github.com/fraunhofer-sit/charra)

8.4. Maturity

The code’s level of maturity is considered to be "prototype".

8.5. Coverage and Version Compatibility

The current version (’6194b3b’) implements a challenge/response interaction model and is aligned with the exemplary specification of the CoAP FETCH bodies defined in Section Appendix A of this document.

8.6. License

The CHARRA project and all corresponding code and data maintained on GitHub are provided under the BSD 3-Clause "New" or "Revised" license.

8.7. Implementation Dependencies

The implementation requires the use of the Trusted Computing Group (TCG) Trusted Software Stack (TSS), and an HSM interoperable with the Trusted Platform Module Library specifications, e.g., a Trusted Platform Module (TPM) 2.0 or equivalent implementation. The corresponding project resources (code and data) for Linux-based operating systems are maintained on GitHub at https://github.com/tpm2-software/tpm2-tss/ (https://github.com/tpm2-software/tpm2-tss/).

The implementation uses the Constrained Application Protocol [RFC7252] (http://coap.technology/) and the Concise Binary Object Representation [RFC7049] (https://cbor.io/).

8.8. Contact

Michael Eckel (michael.eckel@sit.fraunhofer.de)

9. Security and Privacy Considerations

This document outlines three interaction models for remote attestation procedures (RATS) [RFC9334]. While the subsequent sections address additional security and privacy considerations, the security considerations from Section 12 of [RFC9334] must also be adhered to. Additionally, for TPM-based remote attestation, the security considerations outlined in Section 5 of [RFC9683] should be taken into account.

9.1. Selective Disclosure and Obfuscation

This section uses the term "blinding" in the broad sense of privacy-preserving treatment of attributes prior to disclosure (e.g., selective disclosure, obfuscation, or encryption), and not in the more specific cryptographic sense used for blind signatures.

In a remote attestation procedure, the Verifier and the Attester may choose to reduce the disclosure of sensitive attributes while still enabling appraisal. For example, an attribute can be conveyed only as a digest that is included in, and protected by, the Evidence signature; or it can be encrypted for an authorized recipient.

9.2. Trust Assumptions on the Handle Distributor

The handle distributor, as a third party in remote attestation scenarios, holds a critical position similar to that of a Trusted Third Party (TTP). Given its role in generating handles, it has the potential to influence the attestation process significantly. The integrity and reliability of the handles it produces are pivotal for ensuring that the attestation evidence remains trustworthy and that the attestation process is not susceptible to manipulation or interference.

9.2.1. Security Assumptions

* _Integrity and Authenticity_: It is assumed that the handle distributor operates with high levels of integrity and authenticity. Handles generated by the distributor must be secure, unique, and verifiable. This ensures that they cannot be forged or reused maliciously.

* _Isolation and Protection_: The handle distributor must be isolated and protected from unauthorized access and attacks. This includes physical security measures for hardware that might house the handle distributor’s operations, as well as cybersecurity measures to protect against online threats.

* _Auditability_: The operations of the handle distributor should be auditable. This allows for the verification of its compliance with security policies and the integrity of its operations. Regular audits help to ensure that the handle distributor is functioning as expected and has not been compromised.

* _Transparency_: While maintaining security and confidentiality, the processes by which handles are generated and distributed should be as transparent as possible to authorized parties. This helps in building trust and verifying that handles are distributed in a fair and unbiased manner.

9.2.2. Mitigating Risks

To mitigate risks associated with the handle distributor being a central point of potential failure or attack, several measures should be implemented:

* _Redundancy_: Deploying multiple, geographically dispersed handle distributors can ensure continuity of service even if one distributor is compromised or fails.

* _Cryptographic Security_: Using strong cryptographic techniques to protect the generation and transmission of handles ensures that they cannot be tampered with during distribution.

* _Certification and Compliance_: The handle distributor should comply with relevant security standards and undergo regular security certifications. This ensures that they meet industry-wide security benchmarks and maintain high levels of trust.

By defining and adhering to these security assumptions, the role of the handle distributor in remote attestation procedures can be securely managed, minimizing risks and enhancing the overall trust in the attestation process.

9.3. Security Considerations for Brokers in Remote Attestation

The role of the Broker in the "Streaming Remote Attestation with a Broker model" introduces potential security vulnerabilities, including the ability to perform cross-application attacks by manipulating handles and topics. To mitigate these risks, it is essential to implement robust security measures:

* _End-to-End Authentication:_ Establishing end-to-end authenticated channels between Attesters and Verifiers ensures that data integrity and authenticity are preserved across the communication process. This measure prevents the Broker from altering the content of the messages, including Handles and other sensitive data.

* _Strong Isolation of Topics:_ Implementing strong isolation mechanisms for topics can help prevent the Broker from inadvertently or maliciously routing notifications to unauthorized parties. This includes using secure naming conventions and access controls that restrict the Broker's ability to manipulate topic subscriptions.

* _Trusted Association Verification:_ To further safeguard against confusion attacks where the Broker might misroute notifications, mechanisms should be in place to verify the trust association between senders and receivers continuously. This can be facilitated by cryptographic assurances, such as digital signatures and trusted certificates that validate the sender's identity and the integrity of the message content.

* _Audit and Monitoring:_ Regular audits and real-time monitoring of Broker activities can detect and respond to anomalous behavior that might indicate security breaches or manipulation attempts. Logging all actions performed by the Broker provides an audit trail that can be critical for forensic analysis.

* _Broker as a Trusted Third Party (TTP):_ Recognizing the Broker as a TTP necessitates stringent security certifications and compliance with security standards to ensure that they operate under strict governance and security protocols. This includes regular security assessments and certifications that validate the Broker's security practices.

By addressing these vulnerabilities proactively, the integrity and confidentiality of the attestation process can be maintained, reducing the risks associated with Broker-mediated communication in remote attestation scenarios. It is crucial for solution architects to incorporate these security measures during the design and deployment phases to ensure that the attestation process remains secure and trustworthy.

9.4. Additional Application-Specific Security Considerations

The security and privacy requirements for remote attestation can vary significantly based on the deployment environment, the nature of the attestation mechanisms used, and the specific threats each scenario faces. This section details additional security considerations that are pertinent to the interaction models discussed in this document.

9.4.1. Confidentiality

The need for confidentiality in the transmission of attestation information is critical, particularly when exchanges occur over public or untrusted networks, such as the public Internet. For instance, in the _Streaming Remote Attestation with a Broker_ model (cf.Section 7.3.2), where data might traverse multiple nodes, employing TLS can provide necessary confidentiality protections. Similarly, for scenarios involving sensitive environments like carrier management networks, evaluating the confidentiality of the transport medium is crucial to ensure that attestation data remains secure against interception or eavesdropping.

9.4.2. Mutual Authentication

Mutual authentication is particularly relevant in models such as the _Challenge/Response Remote Attestation_ (cf. Section 7.1) where both the Attester and the Verifier engage in bidirectional exchanges of sensitive information. Ensuring that both parties can authenticate each other prevents impersonation attacks, enhancing the trustworthiness of the attestation results.

Practical example mechanisms to achieve mutual authentication include:

Mutual TLS (mTLS): TLS 1.3 [RFC8446] supports mutual authentication via the ertificateRequest message. The client presents its X.509 certificate and proves possession of the private key, establishing bidirectional trust before attestation data is exchanged.

Server-Authenticated TLS with HTTP Authentication: Server-only TLS authentication can be combined with application-layer client authentication. The server authenticates via its TLS certificate, while the client uses HTTP authentication [RFC9110] such as Basic [RFC7617] or Digest credentials. This approach suits deployments where client certificates are impractical.

9.4.3. Hardware Enforcement/Support

In environments where the integrity and security of attestation evidence are paramount, hardware-based security features play a critical role. Technologies like Hardware Security Modules (HSMs), Physically Unclonable Functions (PUFs), and Trusted Execution Environments (TEEs) provide robust protections against tampering and unauthorized access. These are especially important in high-security settings such as financial services or military applications, where attestation processes must rely on physically secure and tamper-resistant components to meet stringent regulatory and security standards.

By addressing these application-specific security requirements within the context of defined interaction models, security strategies can be tailored to fit the unique challenges and operational contexts of different attestation scenarios.

10. Acknowledgments

Olaf Bergmann, Michael Richardson, and Ned Smith

11. References

11.1. Normative References

[BCP205] Best Current Practice 205, <https://www.rfc-editor.org/info/bcp205>. At the time of writing, this BCP comprises the following:

Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <https://www.rfc-editor.org/info/rfc7942>.

[I-D.ietf-rats-epoch-markers] Birkholz, H., Fossati, T., Pan, W., Mihalcea, I., and C. Bormann, "Epoch Markers", Work in Progress, Internet-Draft, draft-ietf-rats-epoch-markers-03, 2 March 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-rats-epoch-markers-03>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC3161] Adams, C., Cain, P., Pinkas, D., and R. Zuccherato, "Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP)", RFC 3161, DOI 10.17487/RFC3161, August 2001, <https://www.rfc-editor.org/rfc/rfc3161>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/rfc/rfc5280>.

[RFC7049] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", RFC 7049, DOI 10.17487/RFC7049, October 2013, <https://www.rfc-editor.org/rfc/rfc7049>.

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <https://www.rfc-editor.org/rfc/rfc7252>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[RFC9334] Birkholz, H., Thaler, D., Richardson, M., Smith, N., and W. Pan, "Remote ATtestation procedureS (RATS) Architecture", RFC 9334, DOI 10.17487/RFC9334, January 2023, <https://www.rfc-editor.org/rfc/rfc9334>.

[RFC9711] Lundblade, L., Mandyam, G., O'Donoghue, J., and C. Wallace, "The Entity Attestation Token (EAT)", RFC 9711, DOI 10.17487/RFC9711, April 2025, <https://www.rfc-editor.org/rfc/rfc9711>.

11.2. Informative References

[DAA] Brickell, E., Camenisch, J., and L. Chen, "Direct Anonymous Attestation", ACM Proceedings of the 11th ACM conference on Computer and Communications Security, page 132-145, 2004.

[DesignPatterns] Gamma, E., Helm, R., Johnson, R., and J. Vlissides, "Design Patterns - Elements of Reusable Object-Oriented Software", Publisher Addison-Wesley, 1994.

[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-07, 10 July 2022, <https://datatracker.ietf.org/doc/html/draft-birkholz-rats-tuda-07>.

[I-D.ietf-rats-endorsements]
Thaler, D., Birkholz, H., and T. Fossati, "RATS Endorsements", Work in Progress, Internet-Draft, draft-ietf-rats-endorsements-09, 2 March 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-rats-endorsements-09>.

[I-D.ietf-rats-network-device-subscription]
Birkholz, H., Voit, E., and W. Pan, "Attestation Event Stream Subscription", Work in Progress, Internet-Draft, draft-ietf-rats-network-device-subscription-11, 30 March 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-rats-network-device-subscription-11>.

[I-D.ietf-spice-sd-cwt]
Prorock, M., Steele, O., Birkholz, H., and R. Mahy, "Selective Disclosure CBOR Web Tokens (SD-CWT)", Work in Progress, Internet-Draft, draft-ietf-spice-sd-cwt-07, 2 March 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-spice-sd-cwt-07>.

[ISIS] Birman, K. and T. Joseph, "Exploiting Virtual Synchrony in Distributed Systems", DOI 10.1145/41457.37515, 1987, <https://doi.org/10.1145/41457.37515>.

[lampson06]
Lampson, B., "Practical Principles for Computer Security", 2006.

[MQTT] OASIS, "Message Queuing Telemetry Transport (MQTT) Version 5.0 Committee Specification 02", Specification Version 5.0, 2018.

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

[RFC7617] Reschke, J., "The 'Basic' HTTP Authentication Scheme", RFC 7617, DOI 10.17487/RFC7617, September 2015, <https://www.rfc-editor.org/rfc/rfc7617>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/rfc/rfc8446>.

[RFC9110] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Semantics", STD 97, RFC 9110, DOI 10.17487/RFC9110, June 2022, <https://www.rfc-editor.org/rfc/rfc9110>.

[RFC9683] Fedorkow, G. C., Ed., Voit, E., and J. Fitzgerald-McKay, "Remote Integrity Verification of Network Devices Containing Trusted Platform Modules", RFC 9683, DOI 10.17487/RFC9683, December 2024, <https://www.rfc-editor.org/rfc/rfc9683>.

[RFC9781] Birkholz, H., O'Donoghue, J., Cam-Winget, N., and C. Bormann, "A Concise Binary Object Representation (CBOR) Tag for Unprotected CBOR Web Token Claims Sets (UCCS)", RFC 9781, DOI 10.17487/RFC9781, May 2025, <https://www.rfc-editor.org/rfc/rfc9781>.

[RFC9783] Tschofenig, H., Frost, S., Brossard, M., Shaw, A., and T. Fossati, "Arm's Platform Security Architecture (PSA) Attestation Token", RFC 9783, DOI 10.17487/RFC9783, June 2025, <https://www.rfc-editor.org/rfc/rfc9783>.

[turtles] Rudnicki, R., "Turtles All the Way Down: Foundation, Edifice, and Ruin in Faulkner and McCarthy", The Faulkner Journal 25.2, DOI 10.1353/fau.2010.0002, 2010, <https://doi.org/10.1353/fau.2010.0002>.

Appendix A. CDDL Specification for a simple CoAP Challenge/Response Interaction

The following CDDL specification is an exemplary proof-of-concept to illustrate a potential implementation of the Challenge/Response Interaction Model. The communication protocol used is CoAP. Both the request message and the response message are exchanged via the FETCH operation and corresponding FETCH request and FETCH response body.

In this example, Evidence is created via the root-of-trust for reporting primitive operation "quote" that is provided by a TPM 2.0.

```
   charra-bodies = charra-attestation-request / charra-attestation-response

   charra-attestation-request = [
       hello: bool,    ; if true, the TPM 2.0 AK Cert shall be conveyed
       key-id: bytes,  ; the key ID to use for signing
       nonce: bytes,   ; a (random) nonce, providing freshness and/or recentness
       pcr-selections: [ * pcr-selection ]
   ]

   pcr-selection = [
       tcg-hash-alg-id: uint .size 2,  ; TPM2_ALG_ID
       pcrs: [
           pcr: uint .size 2
       ]
   ]

   charra-attestation-response = [
       attestation-data: bytes,  ; TPMS_ATTEST.quoted
       tpm2-signature: bytes,
       ? ak-cert: bytes,         ; TPM2 attestation key certificate (AK Cert)
   ]
```

Authors' Addresses

Henk Birkholz
Fraunhofer SIT
Rheinstrasse 75
64295 Darmstadt
Germany
Email: henk.birkholz@ietf.contact

Michael Eckel
Fraunhofer SIT
Rheinstrasse 75
64295 Darmstadt
Germany
Email: michael.eckel@sit.fraunhofer.de

Wei Pan
Huawei Technologies
Email: william.panwei@huawei.com

Eric Voit
Cisco Systems
Email: evoit@cisco.com