TRUSTED® COMPUTING GROUP

SPECIFICATION

DICE Attestation Architecture

Version 1.00
Revision 0.22
September 18, 2020

Contact: admin@trustedcomputinggroup.org

PUBLIC REVIEW

Work in Progress

This document is an intermediate draft for comment only and is subject to change without notice. Readers should not design products based on this document.

DISCLAIMERS, NOTICES, AND LICENSE TERMS

CHANGE HISTORY

REVISION	DATE	DESCRIPTION
1.00/0.22	Sep 18, 2020	Initial Version

CONTENTS

FIGURES

1 SCOPE

This specification defines an attestation architecture for DICE layering architectures.

1.1 Key Words

The key words “MUST,” “MUST NOT,” “REQUIRED,” “SHALL,” “SHALL NOT,” “SHOULD,” “SHOULD NOT,” “RECOMMENDED,” “MAY,” and “OPTIONAL” in this document normative statements are to be interpreted as described in RFC-2119, Key words for use in RFCs to Indicate Requirement Levels.

1.2 Statement Type

Please note a very important distinction between different sections of text throughout this document. There are two distinctive kinds of text: informative comment and normative statements. Because most of the text in this specification will be of the kind normative statements, the authors have informally defined it as the default and, as such, have specifically called out text of the kind informative comment. They have done this by flagging the beginning and end of each informative comment and highlighting its text in gray. This means that unless text is specifically marked as of the kind informative comment, it can be considered a kind of normative statements.

EXAMPLE: Start of informative comment

This is the first paragraph of 1–n paragraphs containing text of the kind *informative comment* ...

This is the second paragraph of text of the kind *informative comment* ...

This is the nth paragraph of text of the kind *informative comment* ...

To understand the TCG specification the user must read the specification. (This use of MUST does not require any action).

End of informative comment

2 REFERENCES

[1] Trusted Computing Group, "TCG Glossary," 2017. [Online]. Available: https://www.trustedcomputinggoup.org.

[2] Internet Engineering Task Force, "Internet Security Glossary, Version 2," 2007. [Online]. Available: https://tools.ietf.org/html/rfc4949.

[3] IEEE, "802.1AR: Secure Device Identity," 2018. [Online]. Available: https://www.ieee.org/.

[4] Internet Engineering Task Force, "The Entity Attestation Token," 2019. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-rats-eat/.

[5] Trusted Computing Group, "DICE Layering Architecture," 2020. [Online]. Available: https://www.trustedcomputinggroup.org/.

[6] Trusted Computing Group, "TCG Trusted Attestation Protocol Information Model for TPM families 1.2 and 2.0 and DICE Family 1.0," 2019. [Online]. Available: https://www.trustedcomputinggroup.org.

[7] Internet Engineering Task Force, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile," 2008. [Online]. Available: https://tools.ietf.org/html/rfc5280.

[8] Internet Engineering Task Force, "Concise Binary Object Representation (CBOR)," 2013. [Online]. Available: https://tools.ietf.org/html/rfc7049.

[9] Internet Engineering Task Force, "Concise Software Identification Tags," 2019. [Online]. Available: https://www.ietf.org.

[10] Internet Engineering Task Force, "CBOR Web Token (CWT)," 2018. [Online]. Available: https://tools.ietf.org/html/rfc8392.

[11] W3C, "Extensible Markup Language (XML) 1.0 (Fifth Edition)," 2008. [Online]. Available: https://www.w3.org/TR/xml/.

[12] NIST, "Guidelines for the Creation of Interoperable Software Identification (SWID) Tags," 2016. [Online]. Available: https://www.nist.gov.

[13] Internet Engineering Task Force, "The JavaScript Object Notation (JSON) Data Interchange Format," 2017. [Online]. Available: https://tools.ietf.org/html/rfc8259.

[14] Internet Engineering Task Force, "Uniform Resource Identifier (URI): Generic Syntax," 2005. [Online]. Available: https://tools.ietf.org/html/rfc3986.

[15] Internet Engineering Task Force, "An Internet Attribute Certificate Profile for Authorization," 2010. [Online]. Available: https://tools.ietf.org/html/rfc5755.

[16] NIST, "Security Requirements for Cryptographic Modules," 2019. [Online]. Available: https://csrc.nist.gov/publications/detail/fips/140/3/final.

[17] Trusted Computing Group, "TCG PC Client Reference Integrity Manifest," 2020. [Online]. Available: https://www.trustedcomputinggroup.org.

[18] Trusted Computing Group, "DICE Certificate Profiles," 2020. [Online]. Available: https://www.trustedcomputinggroup.org.

3 TERMS AND DEFINITIONS

For the purposes of this document, the following terms and definitions apply. Some of these terms have related definitions in the Trusted Computing Group Glossary [1].

3.1 Glossary

TERM	DEFINITION
Actor	An entity, device or service that hosts or implements Attestation Roles.
Appraisal	The action of assessing the trustworthiness of an Attester based on the attestation Claims it provides.
Appraisal Policy for Evidence	A set of rules that direct how a Verifier evaluates the validity of information about an Attester. Typically, a policy is authorized by the Verifier Owner. Compare “security policy” in [2].
Appraisal Policy for Attestation Results	A set of rules that direct how a Relying Party evaluates the validity of information about an Attester. Typically, a policy is authorized by the Relying Party Owner. Compare “security policy” in [2].
Attestation	*See TCG Glossary - Attestation* [1].
Attestation Results	The evaluation results generated by a Verifier, typically including information about an Attester, where the Verifier vouches for the validity of the results.
Attestation Roles	Behaviors associated with attestation and the messages they exchange
Attestation Service Provider	A service provider entity that implements the Verifier role. Typically, the ASP is remote with respect to the Device / Attester. It may also be remote relative to a supply chain entity / Endorser and Resource Manager / Relying Party.
Attester	A role played by a device that provides attestation Evidence to a Verifier.
Attribute Certificate	A structure that contains signed Claims. *See Measurement.*
Assertion	An abstract expression (or information) that describes a trustworthiness property.
Claim	A formatted expression of an Assertion.
Composite Attester	The attester in a Composite Device.
Composite Device	A device with an integrated set of components.
Conveyance	A mechanism for transferring Evidence, Endorsements, Attestation Results, or policies.
Device	An environment that hosts an Attester. *See also TCG Glossary – Trusted Device* [1].
Device Identity	A value that identifies the device, TCB, or RoT.
Endorsements	A protected statement from an entity (typically in a supply chain) that vouches for the trustworthiness of an Attester device.
Endorser	An entity that creates Endorsements that can be used to help evaluate trustworthiness of Attesters

Evidence	A set of protected information about an Attester that is to be evaluated by a Verifier.
Manifest	A structure that contains Endorsements, Evidence, or Attestation Results.
Measurement	*See Evidence, Endorsements.*
Model	See Topology Model
Platform	*See Device. See also TCG Glossary – Platform, Trusted Platform* [1].
Relying Party	A role played by manager of a resource that accepts attestation Results from a Verifier.
Relying Party Owner	An entity, such as an administrator, that is authorized to configure Appraisal Policy for Attestation Results in a Relying Party.
Resource Manager	An entity that hosts the Relying Party.
Role	See Attestation Roles.
Root of Trust	The portion of Device necessary to determine whether a platform is trustworthy, and the misbehavior of which cannot be detected. *See TCG Glossary – Trust, Root of Trust* [1].
Supply Chain Entity	An entity that hosts the Endorser.
Topology Model	The organizational structure of a role composition. This specification provides a canonicalization of commonly used role compositions. These include Passport, Background Check, and Multi-Party Background Check.
Trusted Computing Base	An implementation of a Device having a specific set of attestable Assertions. *See also TCG Glossary – Trusted Component* [1].
Verifier	The role of an attestation service (typically) that accepts Endorsements and conveys attestation Results to a Relying Party. *See also TCG Glossary – Verifier* [1].
Verifier Owner	An entity, such as an administrator, that is authorized to configure Appraisal Policy for evaluating Evidence in a Verifier. *See also TCG Glossary – Owner* [1].
Verifier (Local)	See Verifier, a Local Verifier is Verifier that is implemented on the same platform as the Attesting Environment, e.g., access to a sealed key in a TPM.
Verifier Owner (Local)	The entity that provides the access policy to a Local Verifier, e.g., the entity that writes the TPM policy to accept particular measurements.

3.2 Abbreviations

ABBREVIATION	DESCRIPTION

ASP	Attestation Service Provider
CDI	Compound Device Identifier
CRL	Certificate Revocation List
ECA	Embedded Certificate Authority
IdevID	Initial Device ID. See [3].
RoT	Root of Trust
TCB	Trusted Computing Base
TCI	TCB Component Identifier
UEID	Universal Entity ID. See [4].

4 INTRODUCTION

Start of informative comment

Trustworthiness attributes are not a finite set of values. Attester environments can vary widely, ranging from those highly resistant to attack to those having little or no resistance. Configuration options, if set poorly, can result in a highly resistant environment being operationally less resistant. Computing environments are typically updatable, being constructed from reprogramable hardware, firmware, software, and memory. When a trustworthy environment changes, it is often necessary to determine whether the change transitioned the environment from a trustworthy state to an untrustworthy state. An attestation architecture provides a framework for anticipating when a trust relevant change occurs, what changed, and whether the change is relevant to device security. An attestation framework also creates a context for enabling appropriate responses by applications, system software, and protocol endpoints when trust relevant changes do occur. A trustworthiness assertion is information that describes the properties of a device that affects the Verifier or Relying Party perception of the device's integrity. The set of possible assertions is expected to be determined by the computing environments that support attestation. In many cases, there will be a set of assertions that is widely applicable across most, if not all, computing environments of a particular type. Conversely, there will be assertions that are unique to specific environments or devices. Therefore, this attestation architecture incorporates extensible mechanisms for representing assertions.

Computing environments can be structurally complex and consist of multiple components (memory, CPU, storage, networking, firmware, software). Components are often linked and composed to form computational pipelines, arrays, networks, etc. Not every component is expected to be capable of attestation and attestation capable components may not be capable of attesting to every computing element that interacts with the computing environment. This attestation architecture anticipates use of information modeling techniques that describe computing environment architectures so that verification operations may rely on the information model as an interoperable way to navigate structural complexity.

The attestation capability is itself a computing environment. The act of monitoring trustworthiness attributes, collecting them into an interoperable format, integrity protecting, authenticating, and conveying them employs a computing environment - one that is separate from the one being attested. The trustworthiness of the attestation capability is also a consideration for the attestation architecture. It should be possible for a verifier to understand the trustworthiness properties of the attestation capability for any set of assertions of an attestation flow. This attestation architecture anticipates recursive trust properties and the need for termination. Ultimately, a portion of the computing environment trustworthiness is established via non-automated means. For example, design reviews, manufacturing process audits, and physical security. For this reason, a trustworthy attestation mechanism depends on trustworthy manufacturing and supply chain practices.

End of informative comment

5 ATTESTATION ARCHITECTURE

Start of informative comment

This section defines an attestation architecture for DICE-based identities and layering architectures [5]. The Trusted Computing Group defines two forms of attestation, implicit and explicit [6]. The DICE attestation architecture comprehends both forms of remote attestation. Examples of both attestation forms are provided where the attester device has a DICE layered TCB. Implicit and explicit attestation forms are not mutually exclusive, attestation assertions pertaining to both forms can be asserted as part of the same attestation event.

This attestation architecture defines a set of roles that implement attestation flows. Roles are performed by actors. Actors are deployed entities. Different deployment models may combine or separate various roles in a single actor and may call for differing conveyance mechanisms. However, different deployment models do not fundamentally modify attestation roles, the responsibilities of each role, nor the information flows between them.

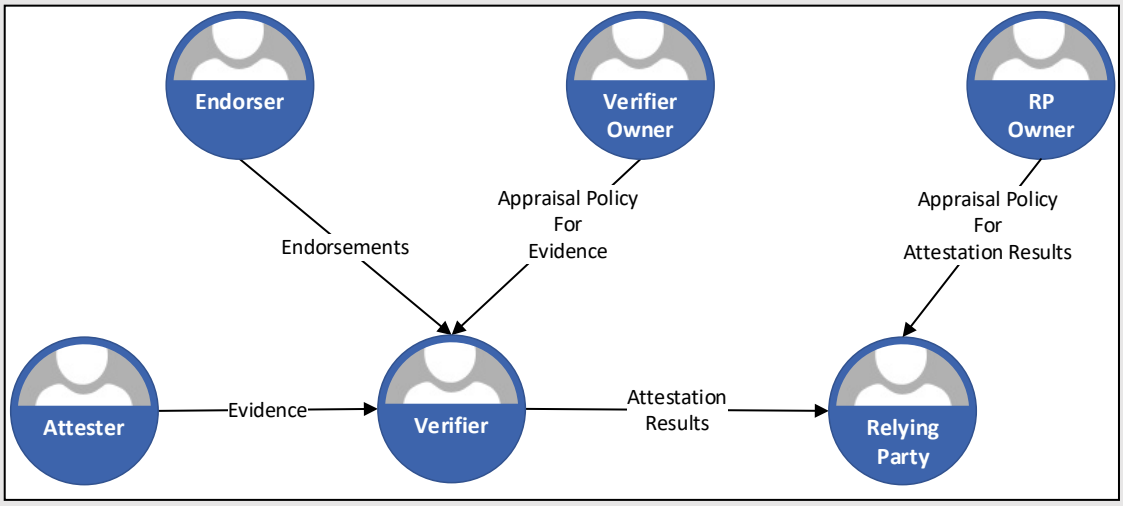

Figure 1: Attestation Roles and message flow

This attestation architecture defines certificate extensions that may be used to construct attestation evidence or reference attestation values.

The basic functions of this attestation architecture are the creation, conveyance, and appraisal of attestation evidence. The Attester creates attestation Evidence that is conveyed to a Verifier for appraisal. The appraisals compare Evidence with Endorsements. Endorsements are the possible values that the Verifier expects to find in Evidence. Endorsements are obtained from manufacturers, vendors, and other supply chain entities called Endorsers. There can be multiple forms of appraisal (e.g., software integrity verification, device composition and configuration verification, device identity and provenance verification). Attestation Results are the output of appraisals that are conveyed to Relying Parties. Attestation Results provide the basis by which the Relying Party may determine a level of confidence in subsequent operations.

This architecture defines attestation Roles (i.e., Attester, Verifier, Endorser, Relying Party, and Owner) and the messages they exchange. Message structure and the various ways in which Roles may be hosted, combined and divided are also part of the architecture. Messages are protected either by a data structure approach (e.g., X.509 certificates, RFC8392) and/or by a conveyance protocol (e.g., RFC5246).

5.1 Attestation Roles

The attestation roles architecture primarily focuses on the trust model elements of a system. There are five roles defined by the attestation roles architecture, as illustrated in Figure 1. Roles consume and/or produce attestation related information. There are a variety of possible configurations in which role interactions may occur. The

attestation roles architecture is a canonical model for a broad range of attestation scenarios. Different scenarios may require topological and/or deployment specific considerations. The primary objective of the attestation roles architecture is to define the functions pertaining to roles and the information exchanged between roles. Attestation roles may be combined and separated as needed to accommodate the requirements of a deployment or use case.

The roles' workflow produces and consumes attestation messages (see Section 5.2). There are a variety of possible configurations. The workflow shown in Figure 2 is the canonical interaction. The canonical interaction is preserved across topological models described in Section 5.3.

5.1.1 Attester Role

The Attester Role provides attestation Evidence to a Verifier. The Attester has an attestation identity that is used to authenticate Evidence. The attestation identity is often established as part of a manufacturing process that embeds identity credentials in the entity that implements an Attester.

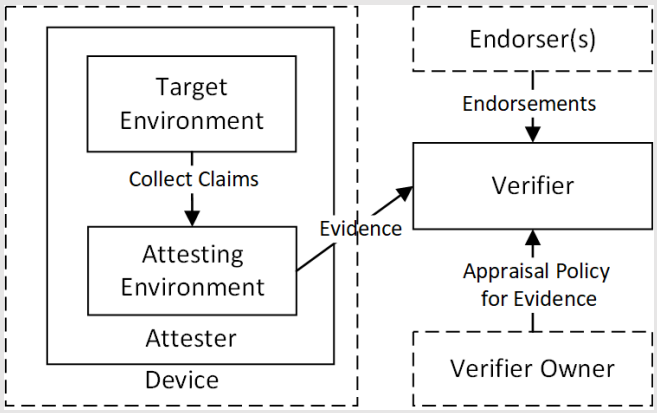

Figure 2: Device with Attesting Environment and Target Environment

The Attester consists of an Attesting Environment and a Target Environment. The Attesting Environment collects assertions, called Claims, about the trustworthiness properties of the Target Environment. Claims are packaged as Evidence by the Attesting Environment, integrity protected and authenticated. The Attesting Environment may also supply additional claims that attest the freshness and recentness of collected claims.

Each TCB in a layered device can be an Attesting Environment that may generate Evidence.

When a DICE layer L_N implements the Attester role, the DICE layer L_{N-1} attests the state of layer L_N, and so forth back to layer L_0.

In this scenario, prior to executing layer N, layer N is the Target Environment for the Attesting Environment executing in layer N-1 (assuming layer N-1 implements the Attester role). When layer N-1 transitions control to layer N, layer N acts as the Attesting Environment for Layer N+1 (assuming layer N also implements the Attester role).

The Attester may interact with the Endorser to communicate device identity and other information. Typically, this happens as part of a manufacturing process involving the construction of the device.

If the Attester and Endorser roles are implemented across DICE layers, the previous layer (L_{N-1}) may implement an Endorser role while the current layer (L_N) or higher layers (L_{N+y}) may implement the Attester role.

5.1.2 Endorser Role

An Endorser role is typically implemented by a supply chain entity that creates reference Endorsements (i.e., values or measurements that are known to be correct, e.g., a reference manifest). Endorsements contain assertions about the device's intrinsic trustworthiness properties. Endorsers implement manufacturing, productization, or other techniques that establish the trustworthiness properties of the Attesting Environments. DICE RoT and TCB layers

contain Attesting Environments. There may be multiple Endorsers for a given device or DICE layer. Endorsers typically authorize Endorsements using digital signatures. For example, certificates [7], manifests [8] [9], or packages.

5.1.3 Verifier Role

The Verifier role is implemented by an Actor that accepts Endorsements and Evidence, and then conveys Attestation Results to one or more Relying Parties. Typically, for remote attestation, a service provider entity implements this role. The Verifier has a trust relationship with its Owner(s) and obtains and applies Appraisal Policies for Evidence as part of Evidence appraisal. The Verifier needs to authenticate Owner policies and the Verifier is trusted to correctly apply supplied policies.

5.1.4 Verifier Owner Role

The Verifier Owner role provides the policy oversight for the Verifier. The Verifier Owner generates Appraisal Policy for Evidence and conveys the policy to the Verifier. The Verifier Owner sets policy for acceptable (or unacceptable) Evidence and Endorsements that may be supplied by Attesters and Endorsers. The policies determine the trustworthiness state of the Attester and how best to represent the state to Relying Parties in the form of Attestation Results.

The Verifier Owner manages Endorsements supplied by Endorsers and may maintain a database of acceptable and/or unacceptable Endorsements. The Verifier Owner authenticates Endorsements and maintains a list of trustworthy Endorsers.

Verifier Owner policies are conveyed to Verifiers. The Verifier works on behalf of the Verifier Owner.

A Verifier Owner is typically implemented by an Actor that deploys management consoles, network management equipment, security enforcement equipment, etc., or performs operational and system lifecycle management functions. The Verifier Owner and Verifier, or Verifier Owner and Relying Party, typically have an established legal or business relationship.

5.1.5 Relying Party Role

The Relying Party role is typically implemented by a resource manager that accepts Attestation Results from a Verifier**.** The Relying Party trusts the Verifier to correctly evaluate Attestation Evidence and Appraisal Policies, and to produce correct Attestation Results. The Relying Party evaluates Attestation Results according to Appraisal Policies for Attestation Results that it receives from the Relying Party Owner.

The Relying Party may take actions based on its evaluation and appraisals. For example, actions may include admitting or denying access, applying remediations, making entries in an audit log, or triggering a financial or other form of transaction. Actions taken by a Relying Party are out of scope of the Attestation Roles model with one exception; a Relying Party may return Results to the Attester for sharing with other Relying Parties.

5.1.6 Relying Party Owner Role

The Relying Party Owner Role (RP Owner) has policy oversight over the Relying Party. The RP Owner sets appraisal policy regarding acceptable (or unacceptable) attestation results about an Attester produced by the Verifier.

The RP Owner attestation policies are made available to the relevant services, management consoles, network equipment, etc., that enforce the policies set by the RP Owner. These are ancillary to this specification's definition of RP Owner and out of scope for this specification.

Note that, as with some other Attestation Roles, the Replying Party Owner Role and the Relying Party Role may be co-located. This means that a single Actor (e.g., a cloud service provider) may implement both the Relying Party Owner Role and the Relying Party Role directly.

5.2 Role Messages

Role messages consist of assertions about trustworthiness properties. Role messages flow between the various roles. The Actor exchanging a role message authenticates the message so that the Actor receiving the message can determine that the originator of the message is expected to perform the role, and so that message integrity is protected. The originator of the message ensures message veracity that the receiver verifies as part of the attestation trust model. Role messages consist of trustworthiness assertions, or Claims. Claims are explicitly realized in tag-value form or as an expression in a data definition language. Actors evaluate role message veracity according to the reputation or trust anchor of the entity asserting the claim.

5.2.1 Evidence

Evidence is a role message containing assertions, i.e., Claims, from the Attester. Evidence should contain freshness and recentness Claims that help establish Evidence relevance. For example, a Verifier supplies a nonce that can be included with the Evidence supplied by the Attester. Evidence typically describes the state of the device or entity. Normally, Evidence is collected in response to a request, i.e., challenge. Evidence may also describe historical device states, e.g., the state of the Attester during initial boot. It may also describe operational states that are dynamic and likely to change from one request to the next. Attestation protocols may be helpful in providing timing context for correct evaluation of Evidence that is highly dynamic.

A DICE layer (L_{N-1}) may supply Evidence about layer (L_N) in an identity credential that is issued by layer (L_{N-1}) if the attesting environment at layer N-1 collects claims about a target environment at layer N.

5.2.2 Appraisal Policy for Evidence

An Appraisal Policy for Evidence is an input to a Verifier that contains policies that reconcile trustworthiness Claims in Evidence with expected operational conditions involving the Attester.

5.2.3 Endorsements

Endorsement structures contain Assertions that are signed by an Actor performing the Endorser role. Endorsements are reference values that may be used by Owners to form Appraisal Policies.

A DICE layer (L_{N-1}) may supply Endorsements about layer (L_N) when Endorsement values are created by layer (L_{N-1}). For example, if layer N-1 randomizes layer N memory layout as part of loading an executable into memory and subsequently collects measurements by accessing layer N memory.

5.2.3.1 Intrinsic Endorsements

Static trustworthiness properties relating to an Actor (e.g., device, environment, TCB, layer, RoT, or entity) exist as part of the design, implementation, validation, and manufacture of that Actor. The Endorser may assert immutable and intrinsic claims in Endorsements allowing Verifier appraisal without requiring Attester reporting.

5.2.4 Attestation Results

Attestation Results are messages containing the results of attestation Evidence appraisals. Attestation Results may contain Claims or other application specific Assertions meaningful to the Relying Party. Attestation Results are authenticated and integrity and confidentiality protected by the Verifier. Attestation Results from a Verifier are presumed to comply with Verifier Owner policies. Consequently, Attestation Results are actionable values in the context of the Relying Party.

5.2.5 Appraisal Policy for Attestation Results

An Appraisal Policy for Attestation Results is an input to a Relying Party that contains policies that reconcile trustworthiness claims in Attestation Results with expected operational conditions involving the Attester.

5.2.6 Message Freshness

The freshness of Role messages affects trustworthiness. The efficacy of trustworthiness properties can deteriorate over time or change subsequent to the collection and reporting of Evidence. For example, when an operational mode changes, or a configuration setting is applied, or environmental conditions change, or physical damage or wear occurs.

Message freshness may be achieved in the following ways:

a) Requester supplied nonce
b) Timestamp
c) Validity period

It may be necessary to include freshness claims as part of Evidence or in conveyance protocols.

5.3 Topology Models

Attestation message exchanges may occur according to a variety of stereotypical patterns. This section identifies several popular message exchange patterns.

5.3.1 Passport Model

The passport model illustrated in Figure 3 defines a sequence of message exchanges that fits a pattern similar to, for example, government issued passports. The passport holder presents identity credentials to the passport issuer who constructs the passport document. The passport document contains markings or other factors that enables a third party to verify the authenticity of the passport document.

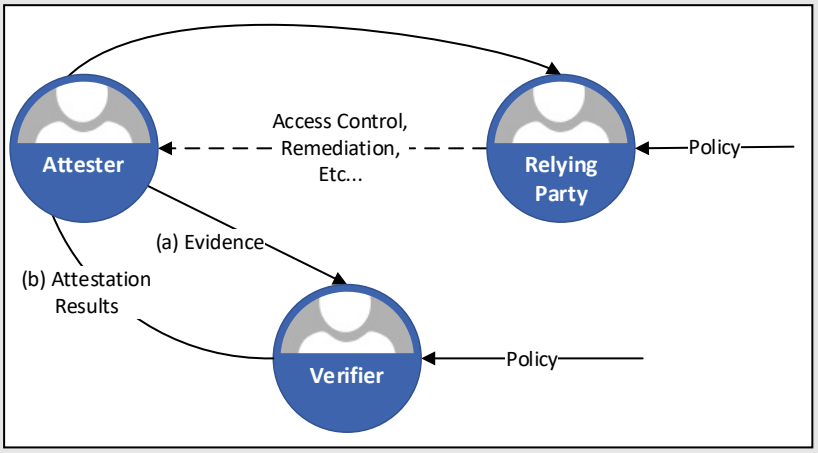

Figure 3: Passport Topology Model

The sequence of steps in the passport model for attestation consists of the following:

a) Attester presents the Evidence message to a Verifier. The Verifier checks message freshness, integrity, and origin. It may be necessary for the Verifier to first provide a nonce to the Attester to guarantee freshness. The Assertions within the message are evaluated against a Policy that identifies Assertions that are acceptable, unacceptable, or unspecified by the Owner. The Verifier creates a Results message containing Assertions or other expressions that represent the attestation evaluation result. The results message is signed by the Verifier or otherwise contains a credential allowing a Relying Party to authenticate the Results from the Verifier.

b) Attestation Results are delivered to the Attester and later forwarded to the Relying Party. The Relying Party authenticates Results, originating from the Verifier, that were provided by the Attester. The Relying Party may verify freshness from both Attester and Verifier. The Relying Party processes the Results according to application defined actions.

In the case of a failed attestation, the Relying Party may need to take one or more implementation-specific actions, such as access control alerts, logging, and/or remediation. Protocols implementing the passport model may need to anticipate ways to perform implementation-specific actions as the conclusion of the previous sequence of steps.

5.3.2 Background Check Model

The background check model illustrated in Figure 4 defines a sequence of message exchanges that fits a background check pattern where the entity receiving credentials is unable to directly process them. Instead, they are processed by a backend entity.

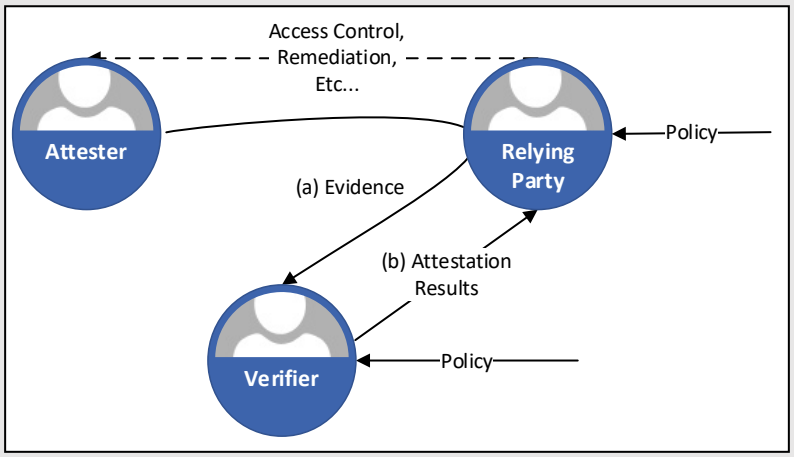

Figure 4: Role Interactions – Background Check Topology Model

The sequence of steps in the background check model for attestation consists of the following:

a) The Attester presents the Evidence message to a Relying Party. The Relying Party checks message freshness, integrity, and origin. It may be necessary for the Relying Party to first provide a nonce to the Attester to guarantee freshness. The Relying Party forwards Evidence to the Verifier.

   The Verifier checks message freshness, integrity, and origin. It may be necessary for the Verifier to first provide a nonce to the Relying Party (which the Relying Party, in turn, provides to the Attester prior to Evidence collection) to guarantee freshness. The Verifier performs appraisal of Evidence as defined in step (a) of the passport model.

b) The Verifier delivers Results to the Relying Party. The Relying Party evaluates Results as defined in step (b) of the passport model.

In the case of a failed attestation, the Relying Party may need to take one or more implementation-specific actions, such as access control alerts, logging, and/or remediation. Protocols implementing the passport model may need to anticipate ways to perform implementation-specific actions as the conclusion of the previous sequence of steps.

5.3.3 Multi-party Background Check Model

The multi-party background check model illustrated in Figure 5 defines a sequence of message exchanges similar to the background check model except that there may be multiple Relying Party entities involved.

The sequence of steps in the multi-party background model for attestation is the same as the steps for the background check model. The multi-party background check model differs in that the Attester forwards the Results to additional Relying Parties that each will evaluate the Results.

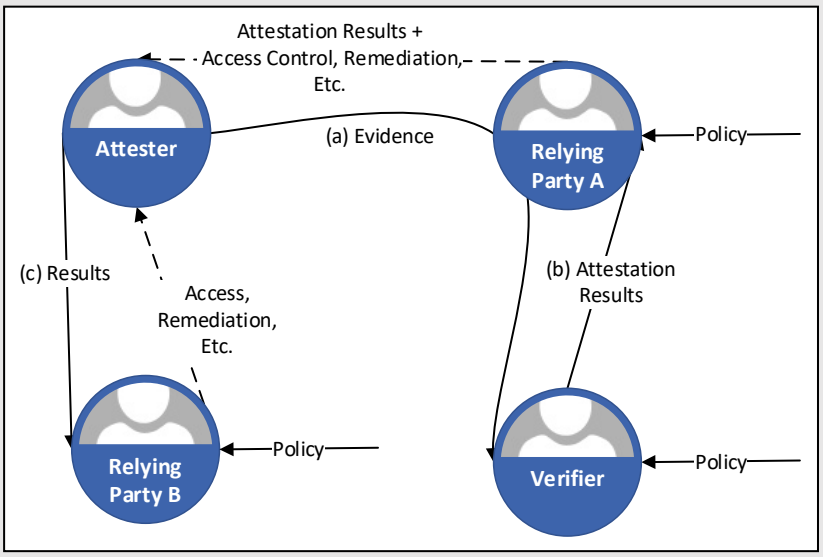

Figure 5: Role Interactions - Multi-Party Background Check Topology Model

The role interactions described here, as well as others not explicitly illustrated, all preserve the basic pattern described by the Attestation Roles Architecture diagram (See Section 5.1). The other patterns described are additions to the basic pattern that do not alter the basic role function.

The topological models presented in this section are for illustrative purposes and are not intended to imply a limitation on the number of role interactions, nor their organization or complexity.

5.4 Assignment of Roles to Actors

Entities that implement Attestation Roles are known as Actors. There are many possible ways to assign roles to Actors. This section identifies common patterns involving role-actor combinations. Actor entities are the deployment environments that host and implement attestation roles (e.g., users, organizations, execution environments, service providers, servers, networks, devices, TEEs, DICE layers, Roots of Trust, etc.).

Actors implement interfaces or protocols used to convey role messages. Conveyance mechanisms are either local or remote. Local conveyance exists when the same Actor is used to perform multiple roles where role message conveyance is internal to that actor. Local conveyance means the protocols for authenticating, protecting, and transmitting role messages are trusted and opaque from the perspective of the co-resident roles. See Figure 6.

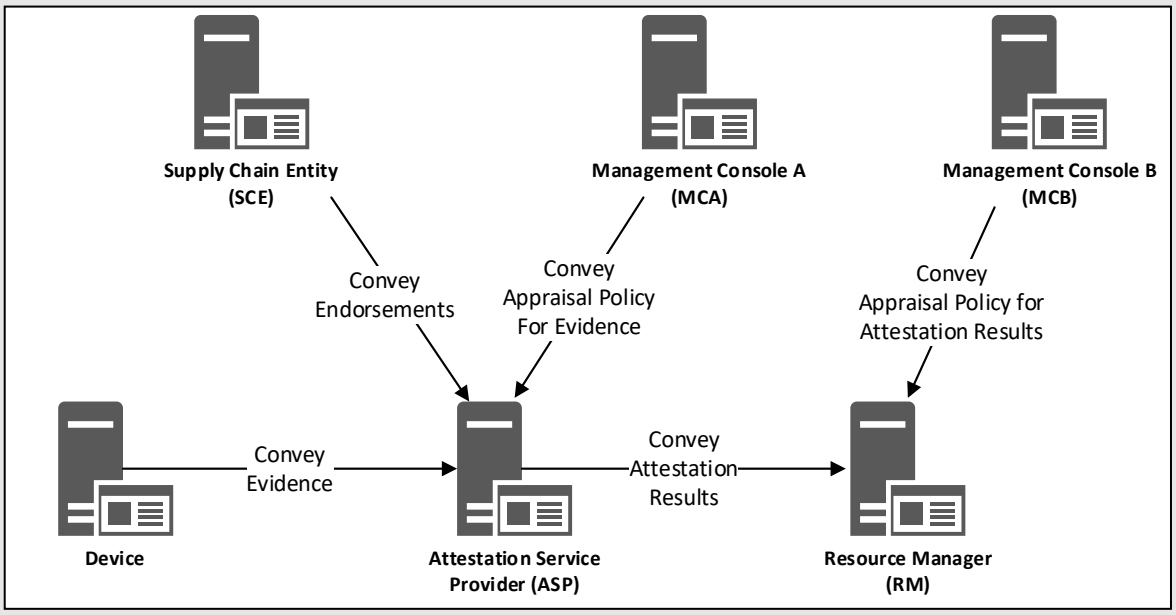

Figure 6: Attestation Actors

The Actor abstraction helps separate the operational elements of attestation from trust model elements.

Deployment architectures can differ significantly to address business, performance, geographic, or political and regulatory considerations. Actor names, credentials, and infrastructure often are reflective of the deployment architecture. For example, an Actor identified by organization name may be issued a certificate that is used to authenticate the Actor to another Actor. Their certifying infrastructures may be leveraged by attestation infrastructure to convey attestation information and to link roles to authentication credentials.

5.4.1 Role-Actor Composition

This section describes scenarios where two or more Actors are combined or co-located. The roles performed by discrete Actors are co-located but are not collectively considered a new hybrid role. Rather, they are recognized as separate roles being hosted by the same device, service, or entity. Actor composition semantics may also apply when virtual environments are dynamically instantiated. Both environments may exist on the same physical device yet have different actor contexts.

5.4.1.1 Co-located Verifier and Relying Party Example

The Resource Manager Actor composition illustrated in Figure 7 co-locates an Attestation Service Provider (ASP) that normally performs the Verifier role with a Resource Manager that normally performs the Relying Party role. The user may dedicate a single server, multiple servers inside a private network, or outsource to a cloud services provider for hosting both roles. Verifier and Relying Party role message interactions have local conveyance properties.

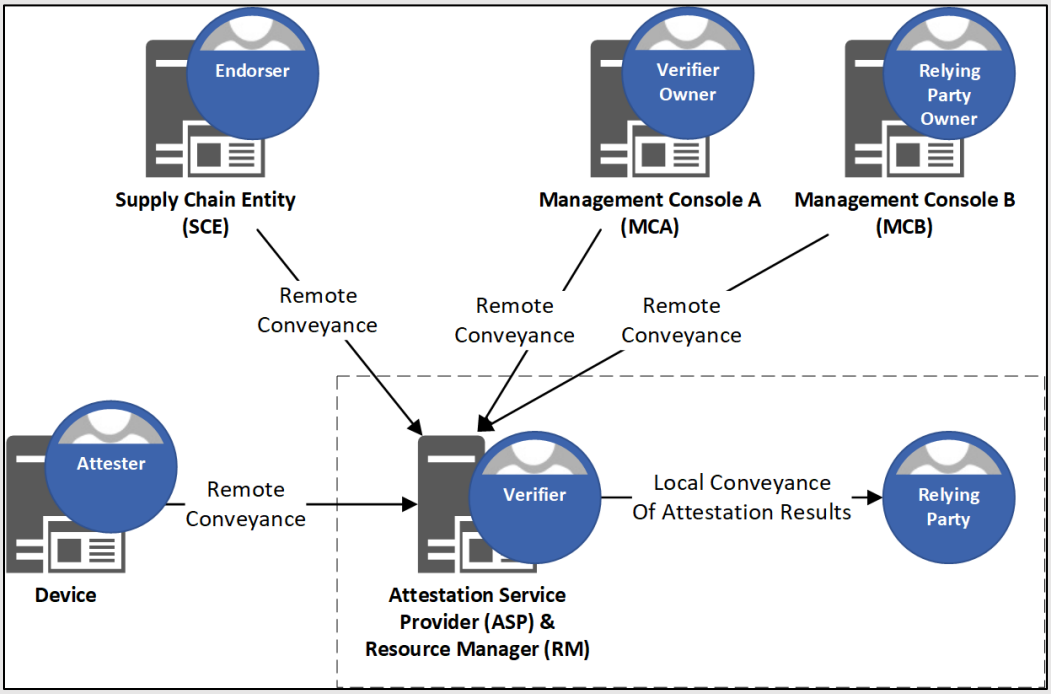

Figure 7: Role-Actor Composition – Combined Verifier and Relying Party Example

5.4.1.2 Composite Attestation Example

In a Composite Device attestation scenario, components have attestation capabilities that generate Evidence. Evidence is conveyed locally to a Composite Attester that assembles the various sets of Claims. The Evidence might also include Claims the Composite Attester directly collects or provides. The Composite Attester conveys Evidence to a remote service provider that hosts a Verifier. Figure 8 provides an illustration of this example.

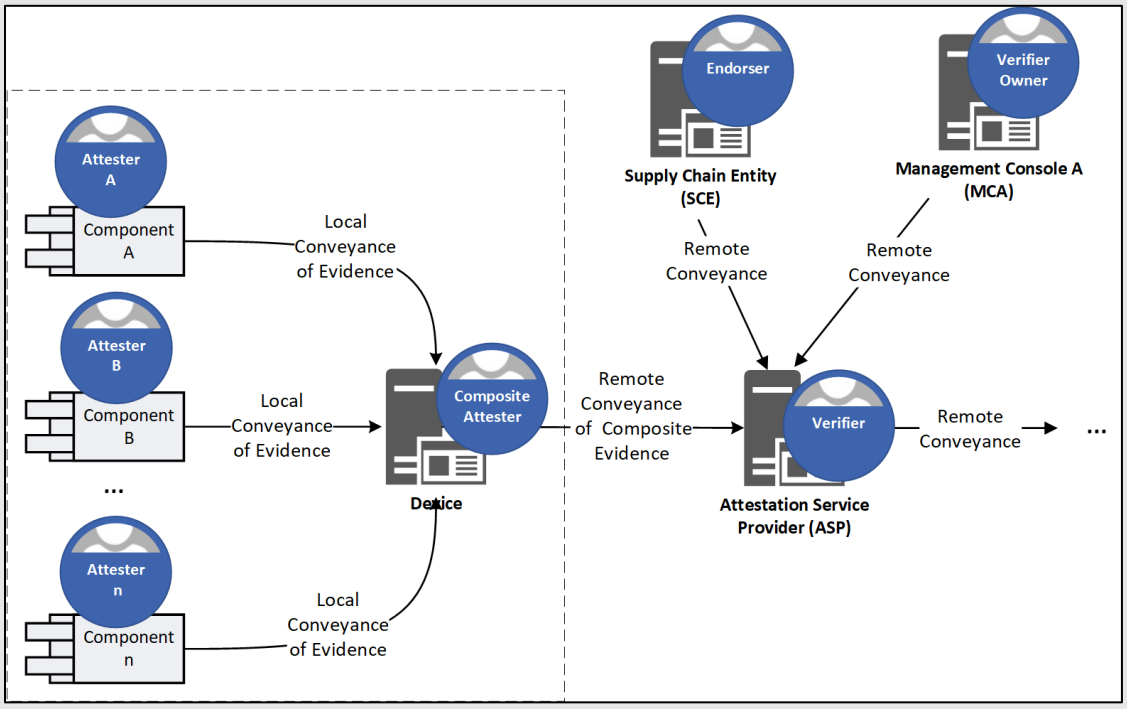

Figure 8: Role-Actor Composition – Composite Device Attestation Example

5.4.1.3 Local Verifier Example

In a local verifier scenario, local components are Attesters that use a local conveyance mechanism to deliver Evidence to the local Verifier for appraisal. The appraisal becomes Attestation Results that are conveyed to a remote Resource Manager that hosts the Relying Party. This example, illustrated in Figure 9, shows the local verifier having a Local Verifier Owner, so appraisal policies are locally conveyed. The local Verifier relies on Endorsements from a supply chain entity that are remotely conveyed.

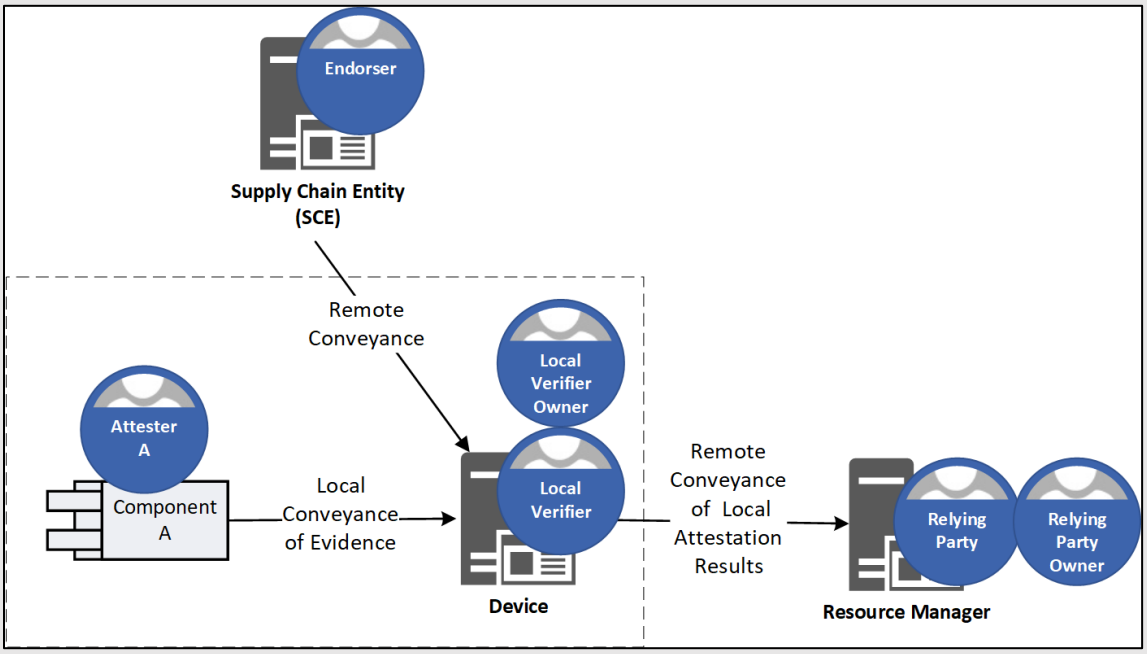

Figure 9: Role-Actor Composition – Local Verifier 777Example

5.4.1.4 Layered Device Attestation Example

In a layered device attestation scenario, a set of layered components each attest the state of the next component. Evidence from each layer is verified by a remote Verifier using an attestation service; therefore, most Evidence is protected for remote appraisals but is conveyed locally. A layered Attester identifies the next layer Attester designated to convey its Evidence. The designation becomes part of the Evidence it produces. This is illustrated in Figure 10.

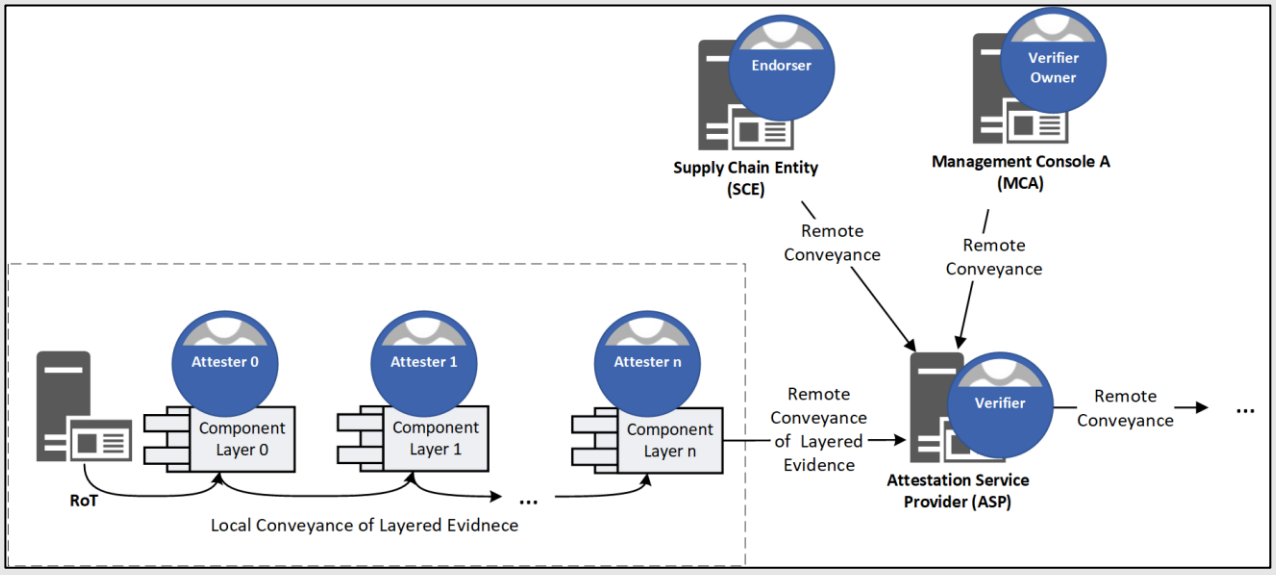

Figure 10: Role-Actor Composition – Layered Attester Example

5.4.2 Actor Composition Summary

Actor composition allows flexibility when determining which Roles an Actor may perform. When multiple Roles are performed by the same Actor, Roles do not interfere with each other.

Local conveyance of Role information must be trustworthy, but its definition is out-of-scope for this specification as it is implementation-specific. Remote conveyance expects that Role information will be communicated via untrusted transports and therefore needs to be protected. Protocol binding specifications are needed to address specific threats.

The examples presented in this specification are for illustrative purposes and are not intended to imply a limitation on the number, organization, or complexity of Role-Actor compositions.

End of informative comment

6 Layered Device Attestation

Layered attestation refers to the traversal of DICE layers where the current layer attests to the state of the next layer. Evidence about the next layer is signed by the current layer. Trust in a current DICE layer depends on the trustworthiness of all previous layers.

Start of informative comment

Consequently, a verifier of layered attestation evaluates attestation evidence of the dependent layers before it can reason about trust in the current layer.

Verifiers recognize when an Attester is performing layered attestation. Inclusion of attestation evidence in a certificate extension issued to a DICE layer by a previous DICE layer helps a Verifier to deduce the presence of layered attestation. The Verifier of a layered attestation always processes this certificate extension if it is supplied.

End of informative comment

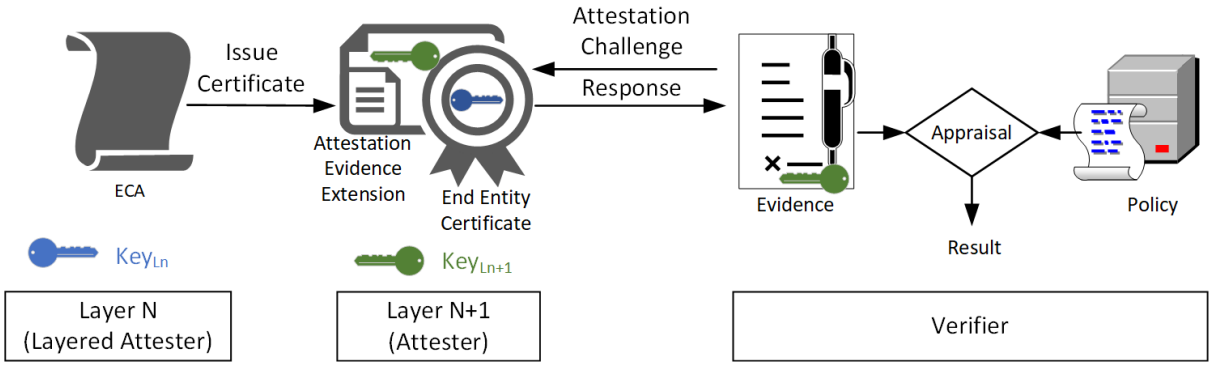

Figure 11: Layered Attestation

Attestation Verifiers require attestation evidence. There are several possible techniques for conveying evidence to a Verifier. This specification specifies the following approaches:

(i) X.509 identity certificates and certificate revocation lists (CRLs) with extensions that contain Evidence
(ii) X.509 attribute certificates containing Evidence
(iii) Manifests containing Evidence

6.1 Evidence as X.509 Certificate Extensions

This section defines X.509v3 certificate and CRL extensions. These extensions encode reference Endorsements about a DICE TCB environment. Certificates containing these extensions are RFC5280 [7] compliant.

Certificate revocation involving a DICE layer can benefit from the added context that attestation evidence provides. Certificate evidence extensions can be used with certificate revocation lists. Consequently, it may be appropriate to include Evidence extension in CRLs.

A certificate issuer uses an extension to assert the trustworthiness claims that apply to the DICE TCB that protects the subject private key that is identified by the certificate subject public key. When the certificate is presented to a Verifier, the reference Endorsements are available for trustworthiness evaluation.

A CRL issuer uses an extension to assert that these trustworthiness claims apply to the DICE TCB environment that protects the subject private key that is identified by the certificate serial number which identifies the certificate that identifies the subject public key. A Verifier, having the CRL, may use Endorsements contained in the extension to evaluate the TCB properties associated with the revocation request. Endorsements in a CRL describe claims that are no longer trustworthy.

The following extensions use an OID arc from the TCG namespace.

TCG Branch: `tcg OBJECT IDENTIFIER ::= {2 23 133}`

DICE Branch: `tcg-dice OBJECT IDENTIFIER ::= { tcg platformClass(5) 4 }`

Start of informative comment

It is recommended that Verifiers process all extensions defined in this specification if present in a certificate.

End of informative comment

6.1.1 TCB Info Evidence Extension

This extension defines attestation Evidence about the DICE layer that is associated with the Subject key. The certificate Subject and SubjectPublicKey identify the entity to which the DiceTcbInfo extension applies. When this extension is used, the measurements in Evidence usually describe the software/firmware which will execute within the TCB.

The AuthorityKeyIdentifier extension MUST be supplied when the DiceTcbInfo extension is supplied. This allows the Verifier to locate the signer's certificate.

Start of informative comment

Inclusion of the DiceTcbInfo extension is optional. However, if omitted an alternative method for conveying the DiceTcbInfo information to the Verifier needs to be provided.

End of informative comment

The DiceTcbInfo extension SHOULD be marked critical. The DiceTcbInfo extension SHOULD be included with CRL entries that revoke the certificate that originally included said DiceTcbInfo extension.

This space intentionally left blank.

The declaration of DiceTcbInfo is:

```
tcg-dice-TcbInfo OBJECT IDENTIFIER ::= {tcg-dice 1}
DiceTcbInfo ::== SEQUENCE {
      vendor      [0] IMPLICIT UTF8String          OPTIONAL,
      model       [1] IMPLICIT UTF8String          OPTIONAL,
      version     [2] IMPLICIT UTF8String          OPTIONAL,
      svn         [3] IMPLICIT INTEGER             OPTIONAL,
      layer       [4] IMPLICIT INTEGER             OPTIONAL,
      index       [5] IMPLICIT INTEGER             OPTIONAL,
      fwids       [6] IMPLICIT FWIDLIST            OPTIONAL,
      flags       [7] IMPLICIT OperationalFlags    OPTIONAL,
      vendorInfo  [8] IMPLICIT OCTET STRING        OPTIONAL,
      type        [9] IMPLICIT OCTET STRING        OPTIONAL
}
FWIDLIST ::== SEQUENCE SIZE (1..MAX) OF FWID
FWID ::== SEQUENCE {
      hashAlg          OBJECT IDENTIFIER,
      digest           OCTET STRING
}
OperationalFlags ::= BIT STRING {
      notConfigured (0),
      notSecure (1),
      recovery (2),
      debug (3)
 }
```

The DiceTcbInfo fields are:

- *vendor* – the entity that created the target TCB (e.g., a TCI value).
- *model* – the product name associated with the target TCB.
- *version* – the revision string associated with the target TCB.
- *svn* – the security version number associated with the target TCB.
- *layer* – the DICE layer associated with this measurement of the target TCB.
- *index* – a value that enumerates measurement of assets within the target TCB and DICE layer.
- *type* – a machine readable description of the measurement
- *fwids* – a list of FWID values resulting from applying the hashAlg function over the object being measured (e.g., target TCB elements used to compute TCI and CDI values). FWIDs are computed by the DICE layer that is the Attesting Environment and certificate Issuer.
- *hashAlg* – an algorithm identifier for the hash algorithm used to produce a digest value.
- *digest* – a digest of firmware, initialization values, or other settings of the target TCB.

- *flags* – a list of flags that enumerate potentially simultaneous operational states of the target TCB:
  - (i) notConfigured,
  - (ii) notSecure,
  - (iii) recovery,
  - (iv) debug.

  A value of 1 (TRUE) means the operational mode is active. A value of 0 (FALSE) means the operational state is not active. If the flags field is omitted, all flags are assumed to be 0 (FALSE).
- *vendorInfo* – vendor supplied values that encode vendor, model, or device specific state.

When filling in the DiceTcbInfo extension, the issuer (layer N) must ensure that any non-constant field contributes to the CDI that generated the subject key (such that a change in the value will cause a change in the CDI). For non-constant fields, the issuer must ensure that it derives the value by measuring the subject layer (layer N+1).

Start of informative comment

Non-constant fields include measurements and values that, if changed, would affect the trustworthiness properties of a device. This may include configuration parameters, measurements of code and/or data, etc.

End of informative comment

The Verifier queries a database containing Endorsements that correspond to this Evidence using a combination of any fields from the DiceTcbInfo. For example, it could query by digest or by vendor, model, and version values. Endorsements should describe how *digest* is computed.

Start of informative comment

Note that both Verifier and Attesting Environments need to consistently apply the digest computation method.

End of informative comment

6.1.1.1 Operational States

Operational states are Evidence claims that, when collected, describe environmental attributes affecting trustworthy operation. For each state, the Attesting Environment (e.g., layer n-1) determines whether the Target Environment (layer n) is currently, or will be, operating in one or more of the specified states:

(i) Not-configured – the target TCB requires additional information to operate.
(ii) Not-secure – the target TCB is not secure.
(iii) Recovery – the target TCB is recovering from a failure.
(iv) Debug – the target TCB can be debugged. The specific debug resources that may be accessed is TCB specific. The TCB environment and software determine which debug resources are accessible.

Operational states can be incorporated into an attestation in several ways:

a) An operational state MAY use different firmware for the different possible states. The digest values differ according to the state used. Consequently, a different CDI value is generated for each operational state used. However, there could be many different firmware images given four operational states. The reference manifest for the target TCB image MUST specify which operational states are inferred by its use.
b) The operational state MAY be explicitly declared in the DiceTcbInfo.flags extension. A different CDI results for each of the possible operational states.
c) The operational states MAY be attested explicitly, not using DiceTcbInfo.
d) The operational states MAY be something other than normal operation and the Attesting Environment (layer n-1) withholds computation of the CDI for the Target Environment (layer n). If the Target Environment attempts to read the CDI, a fault MUST be generated to indicate abnormal operation; and that none of options (a), (b) or (c) are in use.

e) The operational state MAY be explicitly declared in the DiceTcbInfo.vendorInfo extension. A different CDI results for each of the possible vendor defined operational states.

This specification does not define whether combinations of operational states are mutually exclusive. The vendor of the Target Environment TCB and the author of Verifier policies are presumed to have specified this.

6.1.2 Multiple DiceTcbInfo Structures Extension

The initial state of a DICE TCB may be represented by multiple measurements, for example when it is composed of elements supplied by different vendors or when other inputs (for example fuse state) that affect the functionality of the TCB need to be measured. This certificate extension defines a sequence of DiceTcbInfo structures, one for each measurement.

The declaration of DiceMultiTcbInfo is as follows:

```
tcg-dice-MultiTcbInfo OBJECT IDENTIFIER ::= {tcg-dice 5}
DiceTcbInfoSeq ::= SEQUENCE SIZE (1..MAX) OF DiceTcbInfo
```

Use of both the DiceTcbInfo and DiceTcbInfoSeq extensions independently as top-level entities is not recommended. However, if both are used, the DiceTcbInfo extension SHALL be treated as the first element of the list of DiceTcbInfo structures contained in DiceTcbInfoSeq.

6.1.3 UEID Evidence Extension

This extension contains a UEID [4] that identifies the device containing the private key and is identified by the certificate's subjectPublicKey. In the case of its inclusion as a CRL extension, the device containing the private key is identified by the certificate serial number, which identifies the certificate containing the subjectPublicKey.

The declaration of DiceUeid is as follows:

```
tcg-dice-Ueid OBJECT IDENTIFIER ::= {tcg-dice 4}
TcgUeid ::== SEQUENCE {
      ueid OCTET STRING
}
```

When filling in the UEID extension, the issuer must ensure that the content of this extension contributes to the CDI which generated the subject key (such that a change in the field value will cause a change in the CDI).

6.1.4 CWT Claims Set Evidence Extension

The CBOR Web Token (CWT) specification [10] defines a CBOR encoding of a claim set that may be used to contain Evidence. A variant of CWT that does not contain integrity protection; Unprotected CWT Claims Set (UCCS) defines a certificate Evidence extension containing UCCS formatted Evidence.

The tcg-dice-UCCS-evidence is an unsigned structure because the certificate signature authenticates, and integrity protects UCCS contents.

The declaration of DiceUccsEvidence is as follows:

```
tcg-dice-UCCS-evidence OBJECT IDENTIFIER ::= {tcg-dice 6}
UccsEvidence ::== SEQUENCE {
      uccs OCTET STRING
}
```

This extension MAY be used in addition to or in place of DiceTcbInfoSeq or DiceTcbInfo.

When filling in the UCCS extension, the issuer MUST ensure that non-constant elements of this field contributed to the CDI that generated the subject key (such that a change in the field value will also reflect a change in the CDI).

6.1.5 Manifest Evidence Extension

A SWID or CoSWID manifest may be used to contain Evidence.

The declaration of DiceManifestEvidence is as follows:

```
tcg-dice-manifest-evidence OBJECT IDENTIFIER ::= {tcg-dice 7}
```

The tcg-dice-manifest-evidence object identifier is used with the Manifest sequence (See Section 6.5.1.1). The extension SHOULD contain the equivalent of DiceTcbInfoSeq or DiceTcbInfo sequences.

The tcg-dice-manifest-evidence MUST use an unsigned manifest because the certificate signature authenticates and integrity protects manifest contents.

When filling in the manifest evidence extension, the issuer MUST ensure that non-constant elements of this field contributed to the CDI that generated the subject key (such that a change in the field value will also reflect a change in the CDI).

6.1.6 AuthorityKeyIdentifier Certificate Extension

If the AuthorityKeyIdentifier extension is supplied, the keyIdentifier must identify the Issuer public key.

6.2 CRL Extensions

The evidence extensions defined above MAY be included as a certificate revocation list (CRL) extension.

Start of informative comment

If DiceTcbInfo or DiceTcbInfoSeq extensions are present in a CRL entry, then the Verifier needs to use a DiceTcbInfo for verification instead of verifying against issuer and serialNumber fields as normal. If a match condition is found, the Verifier considers the target TCB invalid.

End of informative comment

6.3 Evidence as an X.509 Attribute Certificate

Evidence might be created using an X.509 attribute certificate that is signed by an attestation key. Evidence is collected about a target environment by the Attesting environment that controls the attestation signing key.

Attribute certificates structures that contain Evidence is outside the scope of this specification.

6.4 Evidence as a Manifest

Evidence might be created using a manifest that is signed by an attestation key. Evidence is collected about a target environment by the Attesting Environment that controls the attestation signing key.

6.5 Endorsements

Attestation Verifiers require attestation reference values. Endorsers (i.e., manufacturers and suppliers) create Endorsements that can be used as reference values by a Verifier seeking to compare Evidence to values that correspond to a manufacturer's result. Endorsements may also contain assertions that are not contained in Evidence but may be required by an Appraisal Policy.

This specification specifies the following approaches for encoding Endorsements:

(i) X.509 identity certificate extensions containing Endorsement values.
(ii) X.509 attribute certificates containing Endorsement values.
(iii) CoSWID manifest containing Endorsement values.

(iv) SWID manifest containing Endorsement values.

Including Endorsements with identity certificates adds a manufacturing constraint that reference values and certificate public keys must be known at the time the certificate is issued.

6.5.1 Endorsements as X.509 Certificate Extensions

6.5.1.1 Manifest as an X.509 Certificate Extension

X.509 certificates [7] allow extensions that may contain additional information. Endorsement values may be conveyed using certificate extensions. This extension contains a manifest structure that is signed by an Endorser. The manifest contains reference values about the target TCB. The manifest can be used by a Verifier to appraise Evidence contained in a DiceTcbInfo extension.

The declaration of DiceEndorsementManifest is as follows:

```
tcg-dice-endorsement-manifest OBJECT IDENTIFIER ::= {tcg-dice 2}
Manifest ::== SEQUENCE {0
      format     ManifestFormat,
      manifest   OCTET STRING,
}
ManifestFormat ::= ENUMERATED {
      swid-xml              (0),
      coswid-cbor           (1),
      coswid-json           (2)
}
```

The Manifest Format fields are:

- *format* – defines the manifest schema and encoding format:
  - *swid-xml* – manifest contains an XML [11] encoded SWID Tag manifest as defined by [12].
  - *coswid-cbor* – manifest contains a CBOR [8] encoded CoSWID manifest as defined by [9].
  - *coswid-json* – manifest contains a JSON [13] encoded CoSWID manifest.
- *manifest* – a signed or not signed manifest containing endorsement or evidence claims about a TCB.

6.5.1.2 Endorsement URI as an X.509 Certificate Extension

This extension contains a URI that locates a reference manifest. The manifest contains reference values about the target TCB. The manifest can be used by a Verifier to appraise Evidence contained in the DiceTcbInfo extension.

The declaration for DiceEndorsementManifestUri is as follows:

```
tcg-dice-endorsement-manifest-uri OBJECT IDENTIFIER ::= {tcg-dice 3}
EndorsementManifestURI ::== SEQUENCE {
      emUri      UTF8String,
}
```

The DiceEndorsementManifestUri fields are:

- *emUri* – is a universal resource identifier [14] that contains an object reference to an Endorsement structure. For example, a SWID Tag schema contains a 'tagId' attribute that may be encoded in an emURI.

6.5.2 Endorsements Using X.509 Attribute Certificates

X.509 attribute certificates [15] may contain attribute values that endorse an Attester.

Attribute certificate structures, other than the Endorsement certificate extensions defined previously that contain Endorsements, are outside the scope of this specification.

6.5.3 Endorsements Using Stand-alone Manifests

Endorsement manifests refer to either a SWID Tag [12] or CoSWID [9] structure. The manifest is a signed structure containing Endorsement claims. The manifest signing key may have an associated certificate (e.g., IEEE 802.1AR IDevID) or other credential that contains Endorsement claims.

7 Attesting Environment

This section provides additional guidance, requirements, and design considerations for Attesting Environments.

7.1 Compound Device Identifiers

The Attesting Environment (i.e., the issuer of Evidence) MUST ensure that each non-constant evidence field has contributed to a corresponding CDI value. If a CDI value of an Attester changes, then at least one non-constant evidence field has also changed, and vice versa. This ensures consistency between what is asserted as Evidence and actual conditions described by Evidence.

When generating attestation keys, if the subject key is not derived or generated using the CDI or the CDI is not consistent with actual conditions, then implicitly attested component state may be inaccurate. For non-constant fields, the issuer MUST ensure that it derives the value by measuring the Target Environment whenever a change is made.

7.2 Security Validation

It is often necessary or desirable to ensure an Attester, and therefore, an Attesting Environment, has been validated and complies with a standard set of security requirements. The Federal Information Processing Standard (FIPS) Publication 140-3 [16] is one important example. It is a U.S. government standard that defines minimum security requirements for cryptographic modules in information technology products. This section provides guidance to help facilitate compliance for DICE-based Attesters.

7.2.1 Cryptographic Keys

Start of informative comment

Cryptographic keys in FIPS must be generated from the output of an approved Deterministic Random Bit Generator (DRBG). First, a DRBG is instantiated to create its initial internal state, as illustrated in Figure 12. Once instantiated, a DRBG acts as a one-way function in combination with a monotonic counter and optional entropy for prediction resistance. The monotonic counter represents the DRBG's internal state. The restriction on using the DRBG's internal state only once precludes creation of the same key multiple times.

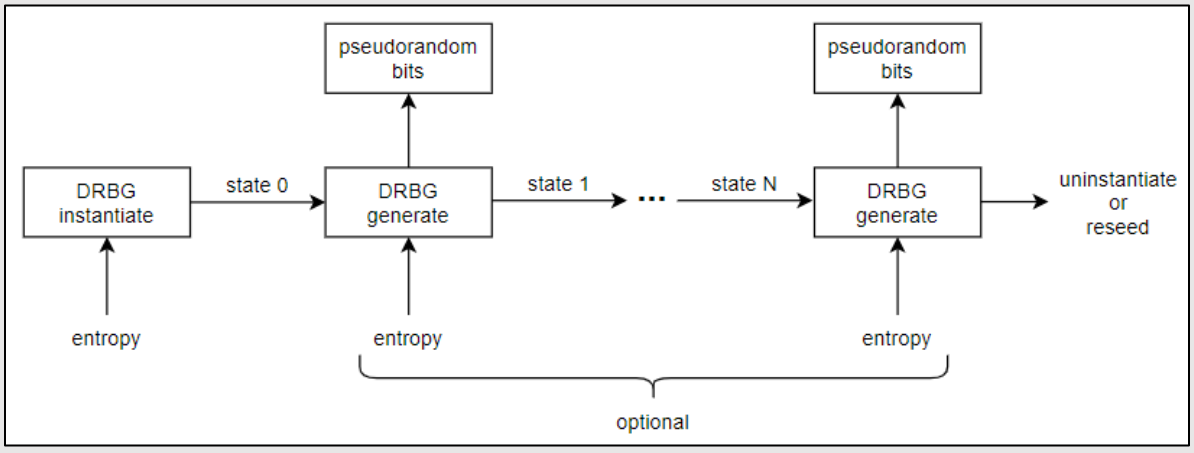

Figure 12: Cryptographic Key Origination in FIPS, DRBG States

To stay within FIPS boundaries, an asymmetric key of any DICE type (ECA, attestation, identity) can be generated only once and stored. The dependency of the generated keys on a CDI is achieved by inputting its creation components into the DRBG as shown in Figure 13. After generated keys are created, they can be stored instead of being later regenerated by reusing DRBG state. A retrieval mechanism can be used to protect the generated keys. A retrieval mechanism that is dependent on a CDI value is required to guarantee that any changes to the creation components of any key result in a different key.

Therefore, a key retrieval mechanism of adequate cryptographic strength is required to accommodate all key types. The strength of the retrieval mechanism must be equal to or higher than the cryptographic strength of the keys to which the retrieval mechanism controls access.

End of informative comment

7.2.2 Retrieval Mechanisms

A key retrieval mechanism controls access to stored asymmetric keys. One way to control access is to encrypt an asymmetric key with an encryption key linked to the creation components of the asymmetric key. This way both asymmetric and encryption keys are derived from the same origin and are bound cryptographically. The asymmetric key is generated once, and the encryption key is recalculated periodically. Any change to the creation components produces an invalid decryption key and precludes retrieval of the valid asymmetric key. An attractive feature of this approach is the ability to store encrypted asymmetric keys in untrusted memory. However, doing so allows for unauthorized modification and substitution which FIPS requires protection against, so additional measures, such as integrity checks, are required. The strength of this retrieval mechanism is dependent on the combination of selected encryption, key derivation, and integrity protection algorithms.

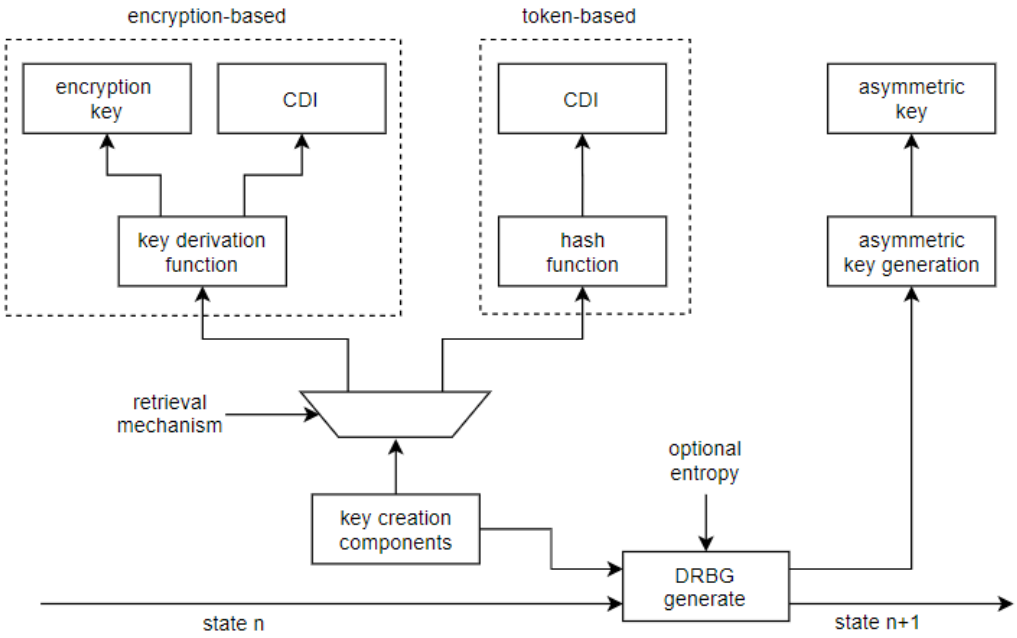

Figure 13: Key Generation for Retrieval Mechanisms

Another way to control access is to employ a logical safeguard mechanism. To retrieve a valid asymmetric key from storage, the requesting party must prove the integrity of key creation components by calculating a retrieval token. Because the requesting party itself is one of the creation components, access is by default self-authenticated. A major benefit of this approach is the possibility of replacing encryption and key derivation algorithms by a single hash function.

In FIPS, a cryptographic key is associated with a single purpose and cryptographic algorithm. As a result, the encryption-based approach requires the CDI to be separated from an encryption key as they are used by different algorithms. On the other hand, the token-based approach allows for creation of a single value per layer (the CDI) which is then used for both linkage to subsequent layers, and access to stored asymmetric keys.

7.2.3 Protected storage

Protected storage is used by retrieval mechanisms to support integrity checks. Regardless of whether the keys are stored encrypted or not, unauthorized modification of stored values must be detected. In addition to integrity checks,

the token-based approach also stores and protects expected retrieval tokens for comparison with the periodically calculated tokens.

7.3 Evidence

7.3.1 Freshness

It may be appropriate to collect a freshness attribute with collected claims that describes the period of time where changes to non-constant fields could have escaped detection by the Attesting Environment. Verifiers are expected to determine whether freshness values are sufficient.

7.3.2 Privacy

If the manifest, attribute certificate, or identity certificate containing attestation extensions does not contain the component's identity or firmware measurement then an attestation Verifier might not be able to associate attestation evidence with an appraisal policy for evidence.

Evidence and Endorsement values conveyed over a public network may be subject to privacy sensitive observation. It is the responsibility of the conveyance protocol carrying Evidence or Endorsement values to confidentiality protect its payloads.

