Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 8 January 2026

M. Ounsworth
Entrust
H. Tschofenig
Siemens
H. Birkholz
Fraunhofer SIT
M. Wiseman

N. Smith
Intel Corporation
7 July 2025

Use of Remote Attestation with Certification Signing Requests
draft-ietf-lamps-csr-attestation-20

Abstract

A PKI end entity requesting a certificate from a Certification Authority (CA) may wish to offer trustworthy claims about the platform generating the certification request and the environment associated with the corresponding private key, such as whether the private key resides on a hardware security module.

This specification defines an attribute and an extension that allow for conveyance of Evidence and Attestation Results in Certificate Signing Requests (CSRs), such as PKCS#10 or Certificate Request Message Format (CRMF) payloads. This provides an elegant and automatable mechanism for transporting Evidence to a Certification Authority.

Including Evidence and Attestation Results along with a CSR can help to improve the assessment of the security posture for the private key, and can help the Certification Authority to assess whether it satisfies the requested certificate profile.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://lamps-wg.github.io/csr-attestation/draft-ietf-lamps-csr-attestation.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-ietf-lamps-csr-attestation/.

Source for this draft and an issue tracker can be found at https://github.com/lamps-wg/csr-attestation.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

When requesting a certificate from a Certification Authority (CA), a PKI end entity may wish to include Evidence or Attestation Results of the security properties of its environments in which the private keys are stored in that request.

Evidence are appraised by Verifiers, which typically produces Attestation Results that serve as input for validating incoming certificate requests against specified certificate policies. Verifiers are associated with Registration Authorities (RAs) or CAs and function as logical entities responsible for processing Evidence and producing Attestation Results. As remote attestation technology matures, it is natural for a Certification Authority to want proof that the requesting entity is in a state that matches the certificate profile. At the time of writing, the most notable example is the Code-Signing Baseline Requirements (CSBR) document maintained by the CA/Browser Forum [CSBR], which requires compliant CAs to "ensure that a Subscriber’s Private Key is generated, stored, and used in a secure environment that has controls to prevent theft or misuse", which is a natural fit to enforce via remote attestation.

This specification defines an attribute and an extension that allow for conveyance of Evidence and Attestation Results in Certificate Signing Requests (CSRs), such as PKCS#10 [RFC2986] or Certificate Request Message Format (CRMF) [RFC4211] payloads. This CSR extension satisfies CA/B Forum’s CSBR [CSBR] requirements for key protection assurance.

As outlined in the IETF RATS architecture [RFC9334], an Attester (typically a device) produces a signed collection of Claims that constitute Evidence about its running environment(s). The term "attestation" is not explicitly defined in RFC 9334 but was later clarified in [I-D.ietf-rats-tpm-based-network-device-attest]. It refers to the process of generating and evaluating remote attestation Evidence.

After the Verifier appraises the Evidence, it generates a new structure called an Attestation Result. A Relying Party utilizes Attestation Results to inform risk or policy-based decisions that consider trustworthiness of the attested entity. This document relies on Section 3 as the foundation for how the various roles within the RATS architecture correspond to a certificate requester and a CA/RA.

The IETF RATS architecture [RFC9334] defines two communication patterns: the _background-check model_ and the _passport model_. In the background-check model, the Relying Party receives Evidence in the CSR from the Attester and must interact with a Verifier service directly to obtain Attestation Results. In contrast, the passport model requires the Attester to first interact with the Verifier service to obtain an Attestation Result token that is then relayed to the Relying Party. This specification defines both communication patterns.

Several standard and proprietary remote attestation technologies are in use. This specification thereby is intended to be as technology-agnostic as it is feasible with respect to implemented remote attestation technologies. Hence, this specification focuses on (1) the conveyance of Evidence and Attestation Results via CSRs while making minimal assumptions about content or format of the transported payload and (2) the conveyance of sets of certificates used for validation of Evidence.

The certificates typically contain one or more certification paths rooted in a device manufacturer trust anchor and the end entity certificate being on the device in question. The end entity certificate is associated with key material that takes on the role of an Attestation Key and is used as Evidence originating from the Attester.

This document specifies a CSR Attribute (or Extension for Certificate Request Message Format (CRMF) CSRs) for carrying Evidence and Attestation Results.

* Evidence is placed into an EvidenceStatement along with an OID to identify its type and optionally a hint to the Relying Party about which Verifier (software package, a microservice or some other service) will be capable of parsing it. A set of EvidenceStatement structures may be grouped together along with the set of CertificateChoice structures needed to validate them to form a EvidenceBundle.

* Attestation Results are carried in the AttestationResult along with an OID to identify its type. A set of AttestationResult structures may be grouped together to form an AttestationResultBundle.

A CSR may contain one or more Evidence payloads. For example Evidence asserting the storage properties of a private key, Evidence asserting firmware version and other general properties of the device, or Evidence signed using different cryptographic algorithms. Like-wise a CSR may also contain one or more Attestation Result payloads.

With these attributes, additional information is available to an RA or CA, which may be used to decide whether to issue a certificate and what certificate profile to apply. The scope of this document is, however, limited to the conveyance of Evidence and Attestation Results within CSR. The exact format of the Evidence and Attestation Results being conveyed is defined in various standard and proprietary specifications.

2. Conventions and Definitions

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.

This document re-uses the terms defined in [RFC9334] related to remote attestation. Readers of this document are assumed to be familiar with the following terms: Evidence, Claim, Attestation Result (AR), Attester, Verifier, Target Environment, Attesting Environment, Composite Device, Lead Attester, Attestation Key, and Relying Party (RP).

The term "Certification Request" message is defined in [RFC2986]. Specifications, such as [RFC7030], later introduced the term "Certificate Signing Request (CSR)" to refer to the Certification Request message. While the term "Certification Request" would have been correct, the mistake was unnoticed. In the meanwhile CSR is an abbreviation used beyond PKCS#10. Hence, it is equally applicable to other protocols that use a different syntax and even a different encoding, in particular this document also considers messages in the Certificate Request Message Format (CRMF) [RFC4211] to be "CSRs". In this document, the terms "CSR" and Certificate Request message are used interchangeably.

The term Hardware Security Modules (HSM) is used generically to refer to the combination of hardware and software designed to protect keys from unauthorized access. Other commonly used terms include Secure Element and Trusted Execution Environment.

3. Architecture

Figure 1 shows the high-level communication pattern of the RATS background check model where the Attester transmits the Evidence in the CSR to the Registration Authority (RA) and the Certification Authority (CA), which is subsequently forwarded to the Verifier. The Verifier appraises the received Evidence and computes an Attestation Result, which is then processed by the RA/CA prior to the certificate issuance. The RA and CA are depicted as separate entities with the RA consuming the Attestation Results and deciding whether or not to forward the certificate request to the CA. In some deployments they are co-located roles. In other deployments, the RA uses a proprietary interface into the CA. In either case, communication between RA and CA is out-of-scope, they can be conceptualized as a single Relying Party entity for the purposes of this specification. This diagram overlays PKI entities with RATS roles in parentheses.

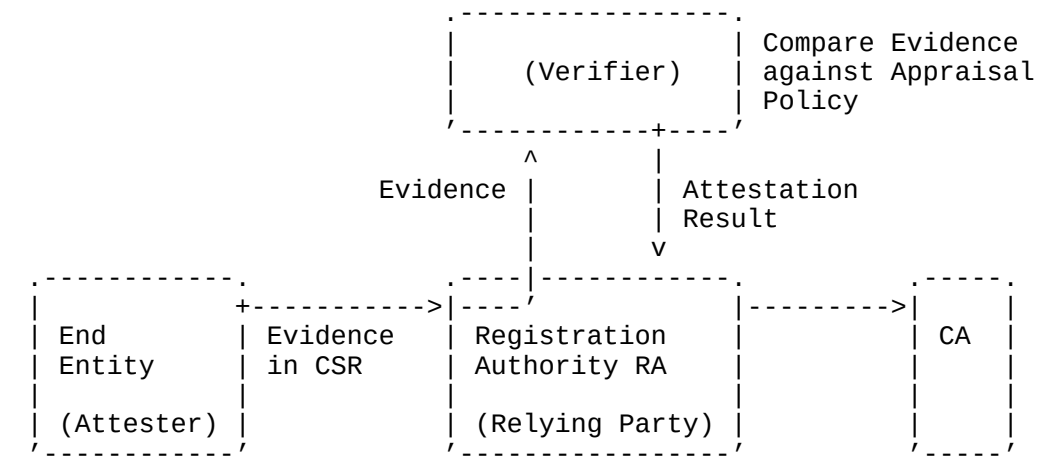


Figure 1: Example data flow demonstrating the architecture with Background Check Model.

In addition to the background-check model, the RATS architecture also defines the passport model, as described in Section 5.2 of [RFC9334]. In the passport model, the Attester transmits Evidence directly to the Verifier to obtain an Attestation Result, which is subsequently forwarded to the Relying Party.

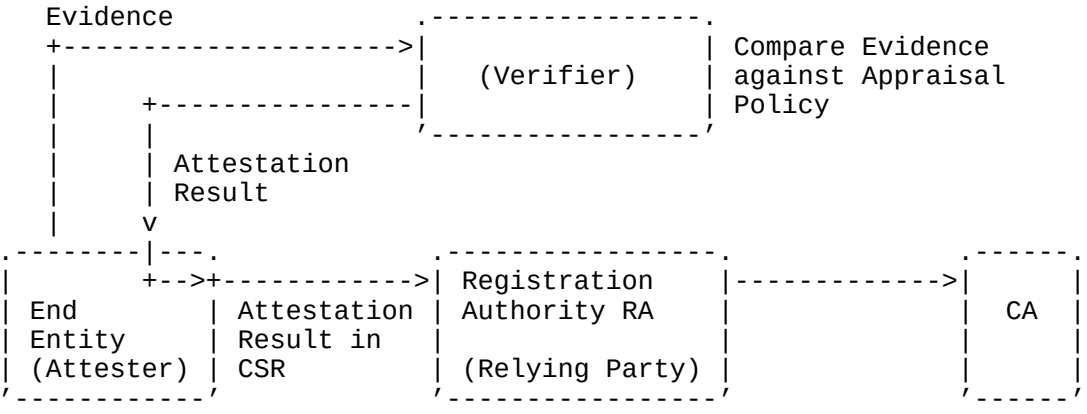


Figure 2: Example data flow demonstrating the architecture with Passport Model.

The choice of model depends on various factors. For instance, the background-check model is preferred when direct real-time interaction between the Attester and the Verifier is not feasible.

The interface by which the Relying Party passes Evidence to the Verifier and receives back Attestation Results may be proprietary or standardized, but in any case is out-of-scope for this document. Like-wise, the interface between the Attester and the Verifier used in the passport model is also out-of-scope for this document.

RFC 9334 [RFC9334] discusses different security and privacy aspects that need to be considered when developing and deploying a remote attestation solution. For example, Evidence may need to be protected against replay and Section 10 of [RFC9334] lists approach for offering freshness. There are also concerns about the exposure of persistent identifiers by utilizing attestation technology, which are discussed in Section 11 of [RFC9334]. Finally, the keying material used by the Attester needs to be protected against unauthorized access, and against signing arbitrary content that originated from outside the device. This aspect is described in Section 12 of [RFC9334]. Most of these aspects are, however, outside the scope of this specification but relevant for use with a given attestation technology.

The focus of this specification is on the transport of Evidence and Attesation Results from the Attester to the Relying Party via existing CSR messages.

4. Information Model

4.1. Model for Evidence in CSR

To support a number of different use cases for the transmission of Evidence and certificate chains in a CSR the structure shown in Figure 3 is used.

On a high-level, the structure is composed as follows: A PKCS#10 attribute or a CRMF extension contains one EvidenceBundle structure. The EvidenceBundle contains one or more EvidenceStatement structures as well as one or more CertificateChoices which enable to carry various format of certificates.

Note: Since an extension must only be included once in a certificate see Section 4.2 of [RFC5280], this PKCS#10 attribute or the CRMF extension MUST only be included once in a CSR.

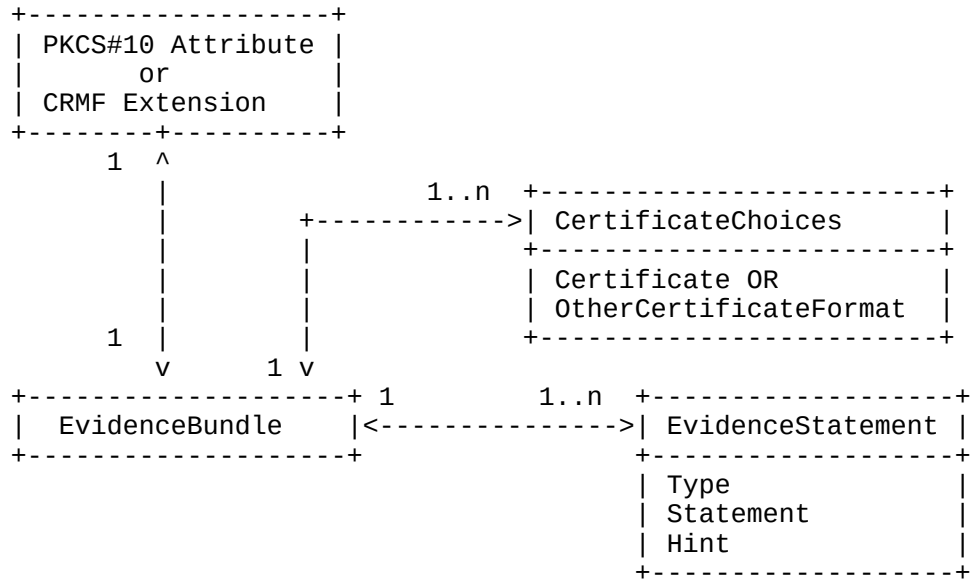


Figure 3: Information Model for CSR Evidence Conveyance.

A conformant implementation of an entity processing the CSR structures MUST be prepared to use certificates found in the EvidenceBundle structure to build a certification path to validate any EvidenceStatement. The following use cases are supported, as described in the sub-sections below.

4.1.1. Case 1 - Evidence Bundle without Certificate Chain

A single Attester, which only distributes Evidence without an attached certificate chain. In the use case, the Verifier is assumed to be in possession of the certificate chain already or the Verifier directly trusts the Attestation Key and therefore no certificate chain needs to be conveyed in the CSR.

As a result, one EvidenceBundle is included in a CSR that contains a single EvidenceStatement without the CertificateChoices structure. Figure 4 shows this use case.

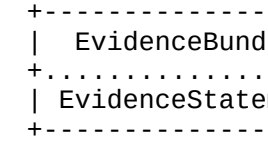


Figure 4: Case 1: Evidence Bundle without Certificate Chain.

4.1.2. Case 2 - Evidence Bundle with Certificate Chain

A single Attester, which shares Evidence together with a certificate chain, is shown in Figure 5. The CSR conveys an EvidenceBundle with a single EvidenceStatement and a CertificateChoices structure.

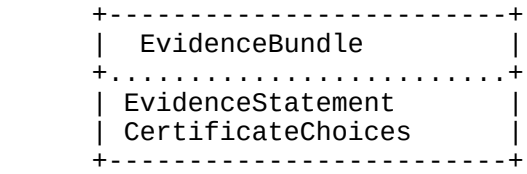


Figure 5: Case 2: Single Evidence Bundle with Certificate Chain.

4.1.3. Case 3 - Evidence Bundles with Multiple Evidence Statements and Complete Certificate Chains

In a Composite Device, which contains multiple Attesters, a collection of Evidence statements is obtained. In this use case, each Attester returns its Evidence together with a certificate chain. As a result, multiple EvidenceStatement structures and the corresponding CertificateChoices structure with the certification chains as provided by the Attester, are included in the CSR. This approach does not require any processing capabilities by a Lead Attester since the information is merely forwarded. Figure 6 shows this use case.

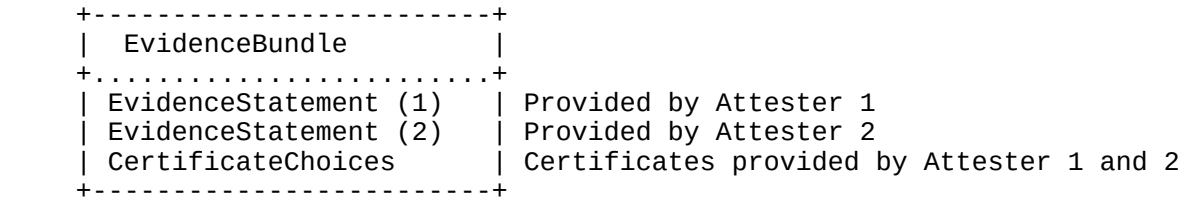


Figure 6: Case 3: Multiple Evidence Structures each with Complete Certificate Chains.

4.2. Model for Attestation Result in CSR

Figure 7 illustrates the information model for transmitting Attestation Results as a PKCS#10 attribute or a CRMF extension. This structure includes a single AttestationResultBundle, which in turn comprises one or more AttestationResult structures.

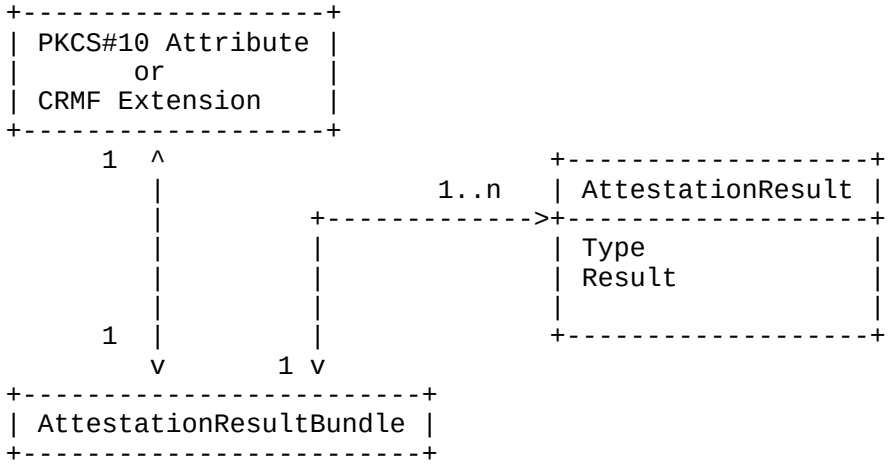


Figure 7: Information Model for CSR Attestation Result Conveyance.

A Relying Party receiving a CSR containing an Attestation Result MUST use the Type information to parse the content. The Attestation Result encoding MUST provide information for the Relying Party to determine the Verifier, who created and protected the Attestation Result against modifications.

5. ASN.1 Elements for Evidence in CSR

5.1. Object Identifiers

This document references id-pkix and id-aa, both defined in [RFC5911] and [RFC5912].

5.2. Evidence Attribute and Extension

By definition, attributes within a PKCS#10 CSR are typed as ATTRIBUTE and within a CRMF CSR are typed as EXTENSION. This attribute definition contains one Evidence bundle of type EvidenceBundle containing one or more Evidence statements of a type EvidenceStatement along with optional certificates for certification path building. This structure enables different Evidence statements to share a certification path without duplicating it in the attribute.

```
EVIDENCE-STATEMENT ::= TYPE-IDENTIFIER

EvidenceStatementSet EVIDENCE-STATEMENT ::= {
   ... -- None defined in this document --
}
```

Figure 8: Definition of EvidenceStatementSet

The expression illustrated in Figure 8 maps ASN.1 Types for Evidence Statements to the OIDs that identify them. These mappings are used to construct or parse EvidenceStatements. Evidence Statements are typically defined in other IETF standards, other standards bodies, or vendor proprietary formats along with corresponding OIDs that identify them.

This list is left unconstrained in this document. However, implementers can populate it with the formats that they wish to support.

```
EvidenceStatement ::= SEQUENCE {
   type   EVIDENCE-STATEMENT.&id({EvidenceStatementSet}),
   stmt   EVIDENCE-STATEMENT.&Type({EvidenceStatementSet}{@type}),
   hint   IA5String OPTIONAL
}
```

Figure 9: Definition of EvidenceStatement

In Figure 9, type is an OID that indicates the format of the value of stmt.

Based on the responsibilities of the different roles in the RATS architecture, Relying Parties need to relay Evidence to Verifiers for evaluation and obtain an Attestation Result in return. Ideally, the Relying Party should select a Verifier based on the received Evidence without requiring the Relying Party to inspect the Evidence itself. This "routing" decision is simple when there is only a single Verifier configured for use by a Relying Party but gets more complex when there are different Verifiers available and each of them capable of parsing only certain Evidence formats.

In some cases, the EvidenceStatement.type OID will be sufficient information for the Relying Party to correctly route it to an appropriate Verifier, however since the type OID only identifies the general data format, it is possible that multiple Verifiers are registered against the same type OID in which case the Relying Party will either require additional parsing of the evidence statement, or the Attester will be required to provide additional information.

To simplify the task for the Relying Party to select an appropriate Verifier an optional field, the hint, is available in the EvidenceStatement structure, as shown in Figure 9. An Attester MAY include the hint to the EvidenceStatement and it MAY be processed by the Relying Party. The Relying Party MAY decide not to trust the information embedded in the hint or policy MAY override any information provided by the Attester via this hint.

When the Attester populates the hint, it MUST contain a server name which uniquely identifies a Verifier. Server names are ASCII strings that contain a hostname and optional port, where the port is implied to be "443" if missing. The names use the format of the authority portion of a URI as defined in Section 3.2 of [RFC3986]. The names MUST NOT include a "userinfo" portion of an authority. For example, a valid server name might be "verifier.example.com" or "verifier.example.com:8443", but not "verifier@example.com".

Relying Parties SHOULD NOT connect to a host name provided in the hint, especially if the verifier has no previous trust relationship with that host name, instead this SHOULD be used only as a lookup string for determining between a list of Verifiers that the Relying Party is pre-configured to use.

In a typical usage scenario, the Relying Party is pre-configured with a list of trusted Verifiers and the corresponding hint values can be used to look up appropriate Verifier. The Relying Party is also configured with a trust anchor for each Verifier, which allows the Verifier to validate the signature protecting the Attestation Result. Tricking a Relying Party into interacting with an unknown and untrusted Verifier must be avoided.

Usage of the hint field can be seen in the TPM2_attest example in Appendix A.2 where the type OID indicates the OID id-TcgAttestCertify and the corresponding hint identifies the Verifier as "tpmverifier.example.com".

```
EvidenceBundle ::= SEQUENCE {
   evidences SEQUENCE SIZE (1..MAX) OF EvidenceStatement,
   certs SEQUENCE SIZE (1..MAX) OF CertificateChoices OPTIONAL,
      -- CertificateChoices MUST only contain certificate or other,
      -- see Section 10.2.2 of [RFC5652]
}
```

The CertificateChoices structure defined in [RFC6268] allows for carrying certificates in the default X.509 [RFC5280] format, or in other non-X.509 certificate formats. CertificateChoices MUST only contain certificate or other. CertificateChoices MUST NOT contain extendedCertificate, v1AttrCert, or v2AttrCert. Note that for non-

ASN.1 certificate formats, the CertificateChoices MUST use other [3] with an OtherCertificateFormat.Type of OCTET STRING, and then can carry any binary data.

The certs field contains a set of certificates that is intended to validate the contents of an Evidence statement contained in evidences, if required. For each Evidnece statement the set of certificates should contain the certificate that contains the public key needed to directly validate the Evidence statement. Additional certificates may be provided, for example, to chain the Evidence signer key back to an agreed upon trust anchor. No specific order of the certificates in certs SHOULD be expected because certificates contained in certs may be needed to validate different Evidence statements.

This specification places no restriction on mixing certificate types within the certs field. For example a non-X.509 Evidence signer certificate MAY chain to a trust anchor via a chain of X.509 certificates. It is up to the Attester and its Verifier to agree on supported certificate formats.

```
id-aa-evidence OBJECT IDENTIFIER ::= { id-aa 59 }

-- For PKCS#10
attr-evidence ATTRIBUTE ::= {
  TYPE EvidenceBundle
  COUNTS MAX 1
  IDENTIFIED BY id-aa-evidence
}

-- For CRMF
ext-evidence EXTENSION ::= {
  SYNTAX EvidenceBundle
  IDENTIFIED BY id-aa-evidence
}
```

Figure 10: Definitions of CSR attribute and extension

The Extension variant illustrated in Figure 10 is intended only for use within CRMF CSRs and is NOT RECOMMENDED to be used within X.509 certificates due to the privacy implications of publishing Evidence about the end entity’s hardware environment. See Section 9.3 for more discussion.

By the nature of the PKIX ASN.1 classes [RFC5912], there are multiple ways to convey multiple Evidence statements: by including multiple copies of attr-evidence or ext-evidence, multiple values within the attribute or extension, and finally, by including multiple

EvidenceStatement structures within an EvidenceBundle. The latter is the preferred way to carry multiple Evidence statements. Implementations MUST NOT place multiple copies of attr-evidence into a PKCS#10 CSR due to the COUNTS MAX 1 declaration. In a CRMF CSR, implementers SHOULD NOT place multiple copies of ext-evidence.

6. ASN.1 Elements for Attestation Result in CSR

6.1. Object Identifiers

This document defines the OID depicted in Figure 11 as an additional CSR Attribute (PKCS#10) or Extension (CRMF) to carry Attestation Results in a CSR.

```
-- OID for Attestation Result types
id-aa-ar OBJECT IDENTIFIER ::= { id-aa (TBD2) }
```

Figure 11: New OID for PKIX Attestation Result Formats

6.2. Attestation Result Attribute and Extension

By definition, attributes within a PKCS#10 CSR are typed as ATTRIBUTE and within a CRMF CSR are typed as EXTENSION. This attribute definition contains one AttestationResultBundle structure.

```
ATTESTATION-RESULT ::= TYPE-IDENTIFIER

AttestationResultSet ATTESTATION-RESULT ::= {
   ... -- None defined in this document --
}
```

Figure 12: Definition of AttestationResultSet

The expression illustrated in Figure 12 maps ASN.1 Types for Attestation Result to the OIDs that identify them. These mappings are used to construct or parse AttestationResults. Attestation Results are defined in other IETF standards (see [I-D.ietf-rats-ar4si]), other standards bodies, or vendor proprietary formats along with corresponding OIDs that identify them.

This list is left unconstrained in this document. However, implementers can populate it with the formats that they wish to support.

```
AttestationResult ::= SEQUENCE {
   type   ATTESTATION-RESULT.&id({AttestationResultSet}),
   stmt   ATTESTATION-RESULT.&Type({AttestationResultSet}{@type}),
}
```

Figure 13: Definition of AttestationResult

In Figure 13, type is an OID that indicates the format of the value of stmt.

```
AttestationResultBundle ::= SEQUENCE SIZE (1..MAX) OF AttestationResult

-- For PKCS#10
attr-ar ATTRIBUTE ::= {
  TYPE AttestationResultBundle
  COUNTS MAX 1
  IDENTIFIED BY id-aa-ar
}

-- For CRMF
ext-ar EXTENSION ::= {
  SYNTAX AttestationResultBundle
  IDENTIFIED BY id-aa-ar
}
```

Figure 14: Definitions of CSR attribute and extension

7. Implementation Considerations

7.1. Is the CSR constructed inside or outside the Attester?

This specification is applicable both in cases where a CSR is constructed internally or externally to the Attesting Environment, from the point of view of the calling application. This section is particularly applicable to the background check model.

Cases where the CSR is generated internally to the Attesting Environment are straightforward: the Hardware Security Model (HSM) generates and embeds the Evidence and the corresponding certification paths when constructing the CSR.

Cases where the CSR is generated externally may require extra communication between the CSR generator and the Attesting Environment, first to obtain the necessary Evidence about the subject key, and then to use the subject key to sign the CSR; for example, a CSR generated by a popular crypto library about a subject key stored on a PKCS#11 [PKCS11] device.

As an example, assuming that the HSM is, or contains, the Attesting Environment and some cryptographic library is assembling a CSR by interacting with the HSM over some network protocol, then the interaction would conceptually be:

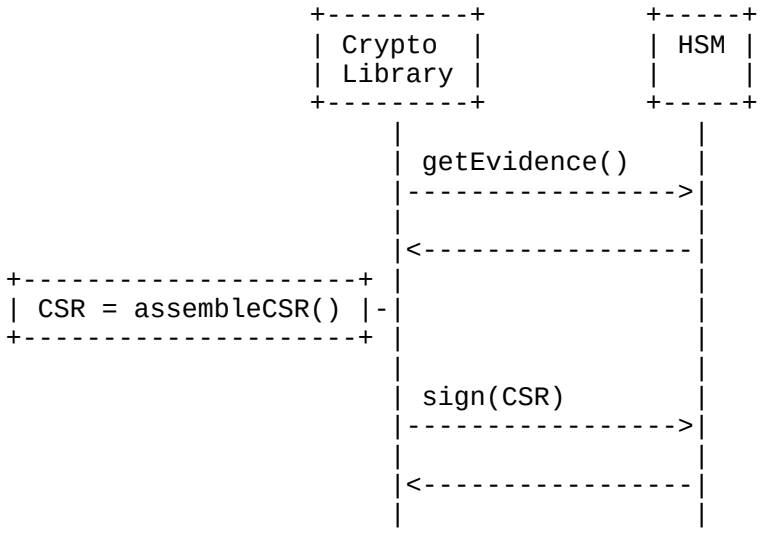


Figure 15: Example interaction between CSR generator and HSM.

7.2. Separation of RA and CA roles with respect to Attestation Results

As described in Section 3, CSRs MAY contain either Evidence or Attestation Results (AR), and also the Registration Authority (RA) and Certification Authority (CA) MAY be conceptualized as a single Relying Party entity, or as separate entities. There are some implications here worth discussion.

In many cases, the Evidence contained within a CSR is intended to be consumed by the RA and not to be placed into the issued certificate. In some RA / CA architectures, it MAY be appropriate for the RA to "consume" the Evidence and remove it from the CSR, re-signing the CSR with an RA signing key. A CRMF CSR also allows the RA to indicate that it verified the CSR without the need to re-signing the CSR.

In any case where the RA and CA roles are separated, and Evidence is evaluated and consumed by the RA, the RA does at least implicitly produce Attestation Results as defined in the RATS Architecture [RFC9334]. For example, the decision to reject the Evidence and fail back to the client, or to accept the Evidence and forward a request to the CA could be viewed as a boolean Attestation Result. Similarly, if acceptance or rejection of the Evidence controls the presence or absence of a certain policy OID or other extension in the issued certificate, this could also be viewed as an Attestation Result.

Alternatively, the RA MAY place explicit Attestation Results into its request to the CA; either for consumption by the CA or for inclusion in the issued certificate. The exact mechanisms for doing this are out-of-scope for this document, but are areas for implementation consideration and potential future standardization work.

8. IANA Considerations

IANA is requested to open two new registries, allocate a value from the "SMI Security for PKIX Module Identifier" registry for the included ASN.1 module, and allocate values from "SMI Security for S/MIME Attributes" to identify two attributes defined within.

8.1. Module Registration - SMI Security for PKIX Module Identifier

IANA is asked to register the following within the registry id-mod SMI Security for PKIX Module Identifier (1.3.6.1.5.5.7.0).

* Decimal: IANA Assigned - *Replace TBDMOD*
* Description: CSR-ATTESTATION-2023 - id-mod-pkix-attest-01
* References: This Document

8.2. Object Identifier Registrations - SMI Security for S/MIME Attributes

IANA is asked to register the following within the registry id-aa SMI Security for S/MIME Attributes (1.2.840.113549.1.9.16.2).

* Evidence Statement
* Decimal: IANA Assigned - This was early-allocated as 59 so that we could generate the sample data.
* Description: id-aa-evidence
* References: This Document
* Attestation Result
* Decimal: IANA Assigned - - *Replace TBD2*
* Description: id-aa-ar
* References: This Document

8.3. Attestation Evidence OID Registry

IANA is asked to create a registry that helps developers to find OID/Evidence mappings that may be encountered in the wild, as well as a link to their specification document. This registry should follow the rules for "Specification Required" as laid out in [RFC5226].

Registration requests should be formatted as per the registration template below, and receive a three-week review period on the [[TBD]] mailing list, with the advice of one or more Designated Experts [RFC8126]. However, to allow for the allocation of values prior to publication, the Designated Experts may approve registration once they are satisfied that such a specification will be published.

Registration requests sent to the mailing list for review should use an appropriate subject (e.g., "Request to register attestation evidence: example").

IANA must only accept registry updates from the Designated Experts and should direct all requests for registration to the review mailing list.

8.3.1. Registration Template

The registry has the following columns:

* OID: The OID number, which has already been allocated. IANA does not allocate OID numbers for use with this registry.

* Description: Brief description of the use of the Evidence and the registration of the OID.

* Reference(s): Reference to the document or documents that register the OID for use with a specific attestation technology, preferably including URIs that can be used to retrieve copies of the documents. An indication of the relevant sections may also be included but is not required.

* Change Controller: The entity that controls the listed data format. For data formats specified in Standards Track RFCs, list the "IESG". For others, give the name of the responsible party. This does not necessarily have to be a standards body, for example in the case of proprietary data formats the Reference may be to a company or a publicly-available reference implementation. In most cases the third party requesting registration in this registry will also be the party that registered the OID. As the intention is for this registry to be a helpful reference, rather than a normative list, a fair amount of discretion is left to the Designated Expert.

8.3.2. Initial Registry Contents

The initial registry contents is shown in the table below. It lists entries for several evidence encoding OIDs including an entry for the Conceptual Message Wrapper (CMW) [I-D.ietf-rats-msg-wrap].

OID	Description	Reference(s)	Change Controller
2 23 133 5 4 1	tcg-dice-TcbInfo	[TCGRegistry]	TCG
2 23 133 5 4 3	tcg-dice-endorsement-manifest-uri	[TCGRegistry]	TCG
2 23 133 5 4 4	tcg-dice-Ueid	[TCGRegistry]	TCG
2 23 133 5 4 5	tcg-dice-MultiTcbInfo	[TCGRegistry]	TCG
2 23 133 5 4 6	tcg-dice-UCCS-evidence	[TCGRegistry]	TCG
2 23 133 5 4 7	tcg-dice-manifest-evidence	[TCGRegistry]	TCG
2 23 133 5	tcg-dice-MultiTcbInfoComp	[TCGRegistry]	TCG

4 8			
2 23 133 5 4 9	tcg-dice- conceptual- message-wrapper	[TCGRegistry]	TCG
2 23 133 5 4 11	tcg-dice- TcbFreshness	[TCGRegistry]	TCG
2 23 133 20 1	tcg-attest-tpm- certify	[TCGRegistry]	TCG
1 3 6 1 5 5 7 1 35	id-pe-cmw	[I-D.ietf-rats-msg-wrap]	IETF

Table 1: Initial Contents of the Attestation Evidence OID Registry

The current registry values can be retrieved from the IANA online website.

9. Security Considerations

In the RATS architecture, when Evidence or an Attestation Result is presented to a Relying Party (RP), the RP may learn detailed information about the Attester unless that information has been redacted or encrypted. Consequently, a certain amount of trust must be placed in the RP, which raises potential privacy concerns because an RP could be used to track devices. This observation is noted in Section 11 of [RFC9334].

Typically, the RPs considered in the RATS architecture are application services that use remote attestation, rather than RAs or CAs. Devices inherently place significant trust in RA/CA infrastructure elements, and therefore any additional information revealed through remote attestation to such entities is generally less concerning than disclosure to application services. The problem of copying Evidence by CAs into an X.509 certificate is discussed in Section 9.3.

These privacy risks can be mitigated using several approaches, including:

* Shared Attestation Keys: A manufacturer of devices may provision all devices with the same attestation key(s), or share a common attestation key across devices of the same product family. This approach anonymizes individual devices by making them indistinguishable from others using the same key(s). However, it also means losing the ability to revoke a single attestation key if a specific device is compromised. Care must be taken to avoid embedding uniquely identifying information in the Evidence, as that would reduce the privacy benefits of using remote attestation.

* Per-Use Attestation Keys: Devices may be designed to dynamically generate distinct attestation keys (and request the corresponding certificates) for each use case, device, or session. This is analogous to the Privacy CA model, in which a device is initially provisioned with an attestation key and certificate; then, in conjunction with a privacy-preserving CA, it can obtain unique keys and certificates as needed. This strategy reduces the potential for tracking while maintaining strong security assurances. This is the model described in this document.

* Anonymous Attestation Mechanisms: Direct anonymous attestation (DAA) or similar cryptographic methods can be employed to generate blinded attestation signatures. In these schemes, the verifier can validate the attestation using a root key but does not gain a global correlation handle. Thus, repeated use of the same attestation key cannot be exploited to track devices. [I-D.ietf-rats-daa] extends the RATS architecture with such a DAA scheme, significantly enhancing privacy.

9.1. Background Check Model Security Considerations

A PKCS#10 or CRMF Certification Request message typically consists of a distinguished name, a public key, and optionally a set of attributes, collectively signed by the entity requesting certification. In general usage, the private key used to sign the CSR MUST be different from the Attesting Key utilized to sign Evidence about the Target Environment, though exceptions MAY be made where CSRs and Evidence are involved in bootstrapping the Attesting Key.

To demonstrate that the private key applied to sign the CSR is generated, and stored in a secure environment that has controls to prevent theft or misuse (including being non-exportable / non-recoverable), the Attesting Environment has to collect claims about this secure environment (or Target Environment, as shown in Figure 16).

Figure 16 shows the interaction inside an Attester. The Attesting Environment, which is provisioned with an Attestation Key, retrieves claims about the Target Environment. The Target Environment offers key generation, storage and usage, which it makes available to services. The Attesting Environment collects these claims about the Target Environment and signs them and exports Evidence for use in remote attestation via a CSR.

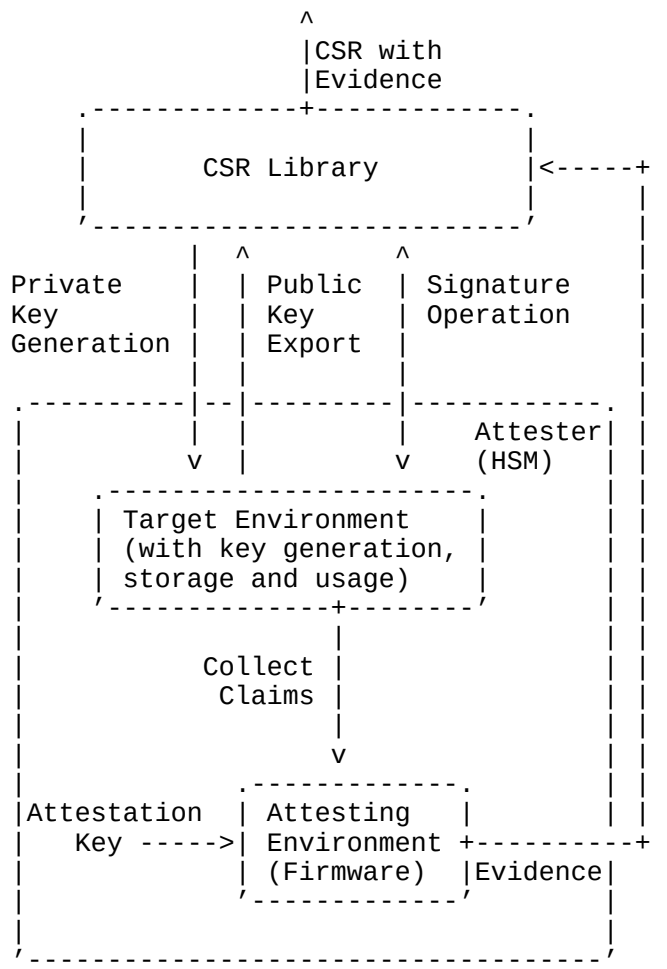


Figure 16: Interaction between Attesting and Target Environment

Figure 16 places the CSR library outside the Attester, which is a valid architecture for certificate enrollment. The CSR library may also be located inside the trusted computing base. Regardless of the placement of the CSR library, an Attesting Environment MUST be able

to collect claims about the Target Environment such that statements about the storage of the keying material can be made. For the Verifier, the provided Evidence must allow an assessment to be made whether the key used to sign the CSR is stored in a secure location and cannot be exported.

Evidence communicated in the attributes and structures defined in this document are meant to be used in a CSR. It is up to the Verifier and to the Relying Party (RA/CA) to place as much or as little trust in this information as dictated by policies.

This document defines the transport of Evidence of different formats in a CSR. Some of these encoding formats are based on standards while others are proprietary formats. A Verifier will need to understand these formats for matching the received claim values against policies.

Policies drive the processing of Evidence at the Verifier: the Verifier's Appraisal Policy for Evidence will often be based on specifications by the manufacturer of a hardware security module, a regulatory agency, or specified by an oversight body, such as the CA Browser Forum. The Code-Signing Baseline Requirements [CSBR] document is an example of such a policy that has been published by the CA Browser Forum and specifies certain properties, such as non-exportability, which must be enabled for storing publicly-trusted code-signing keys. Other policies influence the decision making at the Relying Party when evaluating the Attestation Result. The Relying Party is ultimately responsible for making a decision of what information in the Attestation Result it will accept. The presence of the attributes defined in this specification provide the Relying Party with additional assurance about an Attester. Policies used at the Verifier and the Relying Party are implementation dependent and out of scope for this document. Whether to require the use of Evidence in a CSR is out-of-scope for this document.

9.2. Freshness for the Background Check Model

Evidence generated by an Attester generally needs to be fresh to provide value to the Verifier since the configuration on the device may change over time. Section 10 of [RFC9334] discusses different approaches for providing freshness, including a nonce-based approach, the use of timestamps and an epoch-based technique. The use of nonces requires that nonce to be provided by the Relying Party in some protocol step prior to Evidence and CSR generation, and the use of timestamps requires synchronized clocks which cannot be guaranteed in all operating environments. Epochs also require an out-of-band communication channel. This document leaves the exchange of nonces and other freshness data to certificate management protocols, see

[I-D.ietf-lamps-attestation-freshness]. Developers, operators, and designers of protocols, which embed Evidence-carrying-CSRs, MUST consider what notion of freshness is appropriate and available in-context; thus the issue of freshness is left up to the discretion of protocol designers and implementers.

In the case of Hardware Security Modules (HSM), the definition of "fresh" is somewhat ambiguous in the context of CSRs, especially considering that non-automated certificate enrollments are often asynchronous, and considering the common practice of re-using the same CSR for multiple certificate renewals across the lifetime of a key. "Freshness" typically implies both asserting that the data was generated at a certain point-in-time, as well as providing non-replayability. Certain use cases may have special properties impacting the freshness requirements. For example, HSMs are typically designed to not allow downgrade of private key storage properties; for example if a given key was asserted at time T to have been generated inside the hardware boundary and to be non-exportable, then it can be assumed that those properties of that key will continue to hold into the future.

Note: Freshness is also a concern for remote attestation in the passport model; however, the protocol between the Attester and the Verifier lies outside the scope of this specification.

9.3. Publishing Evidence in an X.509 Extension

This document specifies an Extension for carrying Evidence in a PKCS#10 or CRMF Certificate Signing Request (CSR), but it is intentionally NOT RECOMMENDED for a CA to copy the attr-evidence for PKCS#10 or ext-evidence extension for CRMF into the published certificate. The reason for this is that certificates are considered public information and the Evidence might contain detailed information about hardware and patch levels of the device on which the private key resides. The certificate requester has consented to sharing this detailed device information with the CA but might not consent to having these details published. These privacy considerations are beyond the scope of this document and may require additional signaling mechanisms in the CSR to prevent unintended publication of sensitive information, so we leave it as "NOT RECOMMENDED". Often, the correct layer at which to address this is either in certificate profiles, a Certificate Practice Statement (CPS), or in the protocol or application that carries the CSR to the RA/CA where a flag can be added indicating whether the RA/CA should consider the evidence to be public or private.

9.4. Type OID and Verifier Hint

The EvidenceStatement includes both a type OID and a hint field with which the Attester can provide information to the Relying Party about which Verifier to invoke to parse a given piece of Evidence. Care should be taken when processing these data since at the time they are used, they are not yet verified. In fact, they are protected by the CSR signature but not by the signature from the Attester and so could be maliciously replaced in some cases. The authors' intent is that the type OID and hint will allow an RP to select between Verifier with which it has pre-established trust relationships. The RP MUST NOT blindly make network calls to unknown domain names and trust the results. Implementers should also be cautious around type OID or hint values that cause a short-circuit in the verification logic, such as None, Null, or similar values that could cause the Evidence to appear to be valid when in fact it was not properly checked.

9.5. Additional Security Considerations

In addition to the security considerations listed here, implementers should be familiar with the security considerations of the specifications on this this depends: PKCS#10 [RFC2986], CRMF [RFC4211], as well as general security concepts relating to remote attestation; many of these concepts are discussed in Section 6 of [RFC9334], Section 7 of [RFC9334], Section 9 of [RFC9334], Section 11 of [RFC9334], and Section 12 of [RFC9334]. Implementers should also be aware of any security considerations relating to the specific Evidence and Attestation Result formats being carried within the CSR.

10. References

10.1. Normative References

[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>.

[RFC2986] Nystrom, M. and B. Kaliski, "PKCS #10: Certification Request Syntax Specification Version 1.7", RFC 2986, DOI 10.17487/RFC2986, November 2000, <https://www.rfc-editor.org/rfc/rfc2986>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/rfc/rfc3986>.

[RFC4211] Schaad, J., "Internet X.509 Public Key Infrastructure Certificate Request Message Format (CRMF)", RFC 4211, DOI 10.17487/RFC4211, September 2005, <https://www.rfc-editor.org/rfc/rfc4211>.

[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>.

[RFC5911] Hoffman, P. and J. Schaad, "New ASN.1 Modules for Cryptographic Message Syntax (CMS) and S/MIME", RFC 5911, DOI 10.17487/RFC5911, June 2010, <https://www.rfc-editor.org/rfc/rfc5911>.

[RFC5912] Hoffman, P. and J. Schaad, "New ASN.1 Modules for the Public Key Infrastructure Using X.509 (PKIX)", RFC 5912, DOI 10.17487/RFC5912, June 2010, <https://www.rfc-editor.org/rfc/rfc5912>.

[RFC6268] Schaad, J. and S. Turner, "Additional New ASN.1 Modules for the Cryptographic Message Syntax (CMS) and the Public Key Infrastructure Using X.509 (PKIX)", RFC 6268, DOI 10.17487/RFC6268, July 2011, <https://www.rfc-editor.org/rfc/rfc6268>.

[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>.

10.2. Informative References

[CSBR] CA/Browser Forum, "Baseline Requirements for Code-Signing Certificates, v.3.7", February 2024, <https://cabforum.org/uploads/Baseline-Requirements-for-the-Issuance-and-Management-of-Code-Signing.v3.7.pdf>.

[I-D.bft-rats-kat]
Brossard, M., Fossati, T., Tschofenig, H., and H. Birkholz, "An EAT-based Key Attestation Token", Work in Progress, Internet-Draft, draft-bft-rats-kat-05, 21 November 2024, <https://datatracker.ietf.org/doc/html/draft-bft-rats-kat-05>.

[I-D.ffm-rats-cca-token]
Frost, S., Fossati, T., and G. Mandyam, "Arm's Confidential Compute Architecture Reference Attestation Token", Work in Progress, Internet-Draft, draft-ffm-rats-cca-token-01, 3 March 2025, <https://datatracker.ietf.org/doc/html/draft-ffm-rats-cca-token-01>.

[I-D.ietf-lamps-attestation-freshness]
Tschofenig, H. and H. Brockhaus, "Nonce-based Freshness for Remote Attestation in Certificate Signing Requests (CSRs) for the Certification Management Protocol (CMP) and for Enrollment over Secure Transport (EST)", Work in Progress, Internet-Draft, draft-ietf-lamps-attestation-freshness-04, 7 July 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-lamps-attestation-freshness-04>.

[I-D.ietf-rats-ar4si]
Voit, E., Birkholz, H., Hardjono, T., Fossati, T., and V. Scarlata, "Attestation Results for Secure Interactions", Work in Progress, Internet-Draft, draft-ietf-rats-ar4si-08, 6 February 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-rats-ar4si-08>.

[I-D.ietf-rats-daa]
Birkholz, H., Newton, C., Chen, L., and D. Thaler, "Direct Anonymous Attestation for the Remote Attestation Procedures Architecture", Work in Progress, Internet-Draft, draft-ietf-rats-daa-07, 3 March 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-rats-daa-07>.

[I-D.ietf-rats-msg-wrap]
Birkholz, H., Smith, N., Fossati, T., Tschofenig, H., and D. Glaze, "RATS Conceptual Messages Wrapper (CMW)", Work in Progress, Internet-Draft, draft-ietf-rats-msg-wrap-16, 3 July 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-rats-msg-wrap-16>.

[I-D.ietf-rats-tpm-based-network-device-attest]
Fedorkow, G., Voit, E., and J. Fitzgerald-McKay, "TPM-based Network Device Remote Integrity Verification", Work in Progress, Internet-Draft, draft-ietf-rats-tpm-based-network-device-attest-14, 22 March 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-rats-tpm-based-network-device-attest-14>.

[I-D.tschofenig-rats-psa-token]
Tschofenig, H., Frost, S., Brossard, M., Shaw, A. L., and T. Fossati, "Arm’s Platform Security Architecture (PSA) Attestation Token", Work in Progress, Internet-Draft, draft-tschofenig-rats-psa-token-24, 23 September 2024, <https://datatracker.ietf.org/doc/html/draft-tschofenig-rats-psa-token-24>.

[PKCS11] OASIS, "PKCS #11 Cryptographic Token Interface Base Specification Version 2.40", April 2015, <http://docs.oasis-open.org/pkcs11/pkcs11-base/v2.40/os/pkcs11-base-v2.40-os.html>.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", RFC 5226, DOI 10.17487/RFC5226, May 2008, <https://www.rfc-editor.org/rfc/rfc5226>.

[RFC7030] Pritikin, M., Ed., Yee, P., Ed., and D. Harkins, Ed., "Enrollment over Secure Transport", RFC 7030, DOI 10.17487/RFC7030, October 2013, <https://www.rfc-editor.org/rfc/rfc7030>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/rfc/rfc8126>.

[SampleData]
"CSR Attestation Sample Data", n.d., <https://github.com/lamps-wg/csr-attestation-examples>.

[TCGDICE1.1]
Trusted Computing Group, "DICE Attestation Architecture", January 2024, <https://trustedcomputinggroup.org/wp-content/uploads/DICE-Attestation-Architecture-Version-1.1-Revision-18_pub.pdf>.

[TCGRegistry]
Trusted Computing Group, "TCG OID Registry", October 2024, <https://trustedcomputinggroup.org/resource/tcg-oid-registry/>.

[TPM20] Trusted Computing Group, "Trusted Platform Module Library Specification, Family 2.0", n.d., <https://trustedcomputinggroup.org/resource/tpm-library-specification/>.

Appendix A. Examples

This section provides several examples and sample data for embedding Evidence in CSRs. The first example embeds Evidence produced by a TPM in the CSR. The second example conveys an Arm Platform Security Architecture token, which provides claims about the used hardware and software platform, into the CSR.

After publication of this document, additional examples and sample data will be collected at the following GitHub repository [SampleData]:

https://github.com/lamps-wg/csr-attestation-examples

A.1. Extending EvidenceStatementSet

As defined in Section 5.2, EvidenceStatementSet acts as a way to provide an ASN.1 compiler or runtime parser with a list of OBJECT IDENTIFIERs that are known to represent EvidenceStatements -- and are expected to appear in an EvidenceStatement.type field, along with the ASN.1 type that should be used to parse the data in the associated EvidenceStatement.stmt field. Essentially this is a mapping of OIDs to data structures. Implementers are expected to populate it with mappings for the Evidence types that their application will be handling.

This specification aims to be agnostic about the type of data being carried, and therefore does not specify any mandatory-to-implement Evidence types.

As an example of how to populate EvidenceStatementSet, implementing the TPM 2.0 and PSA Evidence types given below would result in the following EvidenceStatementSet definition:

```
EvidenceStatementSet EVIDENCE-STATEMENT ::= {
  --- TPM 2.0
  { Tcg-attest-tpm-certify IDENTIFIED BY tcg-attest-tpm-certify },
  ...,

  --- PSA
  { OCTET STRING IDENTIFIED BY { 1 3 6 1 5 5 7 1 99 } }
}
```

A.2. TPM V2.0 Evidence in CSR

This section describes TPM2 key attestation for use in a CSR.

This is a complete and canonical example that can be used to test parsers implemented against this specification. Readers who wish the sample data may skip to Appendix A.2.6; the following sections explain the TPM-specific data structures needed to fully parse the contents of the evidence statement.

A.2.1. TCG Key Attestation Certify

There are several ways in TPM2 to provide proof of a key’s properties. (i.e., key attestation). This description uses the simplest and most generally expected to be used, which is the TPM2_Certify and the TPM2_ReadPublic commands.

This example does not describe how platform attestation augments key attestation. The properties of the key (such as the name of the key, the key usage) in this example do not change during the lifetime of the key.

A.2.2. TCG OIDs

The OIDs in this section are defined by TCG TCG has a registered arc of 2.23.133

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

tcg-kp-AIKCertificate OBJECT IDENTIFIER ::= { id-tcg 8 3 }

tcg-attest OBJECT IDENTIFIER ::= { tcg 20 }

tcg-attest-tpm-certify OBJECT IDENTIFIER ::= { tcg-attest 1 }
```

The tcg-kp-AIKCertificate OID in extendedKeyUsage identifies an AK Certificate in RFC 5280 format defined by TCG. This certificate would be a certificate in the EvidenceBundle defined in Section 5.2. (Note: The abbreviation AIK was used in TPM 1.2 specification. TPM 2.0 specifications use the abbreviation AK. The abbreviations are interchangeable.)

A.2.3. TPM2 AttestationStatement

The EvidenceStatement structure contains a sequence of two fields: a type and a stmt. The 'type' field contains the OID of the Evidence format and it is set to tcg-attest-tpm-certify. The content of the structure shown below is placed into the stmt, which is a concatenation of existing TPM2 structures. These structures will be explained in the rest of this section.

```
Tcg-csr-tpm-certify ::= SEQUENCE {
  tpmSAttest       OCTET STRING,
  signature        OCTET STRING,
  tpmTPublic       OCTET STRING OPTIONAL
}
```

A.2.4. Introduction to TPM2 concepts

The definitions in the following sections are specified by the Trusted Computing Group (TCG). TCG specification including the TPM2 set of specifications [TPM20], specifically Part 2 defines the TPM 2.0 structures. Those familiar with TPM2 concepts may skip to Appendix A.2.3 which defines an ASN.1 structure specific for bundling a TPM attestation into an EvidenceStatement, and Appendix A.2.6 which provides the example. For those unfamiliar with TPM2 concepts this section provides only the minimum information to understand TPM2 Attestation in CSR and is not a complete description of the technology in general.

A.2.5. TCG Objects and Key Attestation

This provides a brief explanation of the relevant TPM2 commands and data structures needed to understand TPM2 Attestation used in this RFC. NOTE: The TPM2 specification used in this explanation is version 1.59, section number cited are based on that version. Note also that the TPM2 specification comprises four documents: Part 1: Architecture; Part 2: Structures; Part 3: Commands; Part 4: Supporting Routines.

Note about convention: All structures starting with TPM2B_ are:

* a structure that is a sized buffer where the size of the buffer is contained in a 16-bit, unsigned value.

* The first parameter is the size in octets of the second parameter. The second parameter may be any type.

A full explanation of the TPM structures is outside the scope of this document. As a simplification references to TPM2B_ structures will simply use the enclosed TPMT_ structure by the same name following the '_'.

A.2.5.1. TPM2 Object Names

All TPM2 Objects (e.g., keys are key objects which is the focus of this specification). A TPM2 object name is persistent across the object's life cycle whether the TPM2 object is transient or persistent.

A TPM2 Object name is a concatenation of a hash algorithm identifier and a hash of the TPM2 Object's TPMT_PUBLIC.

```
Name  nameAlg || HnameAlg (handle→publicArea)
nameAlg is a TCG defined 16 bit algorithm identifier
```

publicArea is the TPMT_PUBLIC structure for that TPM2 Object.

The size of the Name field can be derived by examining the nameAlg value, which defines the hashing algorithm and the resulting size.

The Name field is returned in the TPM2B_ATTEST data field.

```
typedef struct {
     TPM_GENERATED magic;
     TPMI_ST_ATTEST type;
     TPM2B_NAME qualifiedSigner;
     TPM2B_DATA extraData;
     TPMS_CLOCK_INFO clockInfo;
     UINT64 firmwareVersion;
     TPMU_ATTEST attested;
} TPMS_ATTEST;
```

where for a key object the attested field is

```
typedef struct {
     TPM2B_NAME name;
     TPM2B_NAME qualifiedName;
} TPMS_CERTIFY_INFO;
```

A.2.5.2. TPM2 Public Structure

Any TPM2 Object has an associated TPM2 Public structure defined as TPMT_PUBLIC. This is defined below as a 'C' structure. While there are many types of TPM2 Objects each with its own specific TPMT_PUBLIC structure (handled by the use of 'unions') this document will specifically define TPMT_PUBLIC for a TPM2 key object.

```
typedef struct {
     TPMI_ALG_PUBLIC type;
     TPMI_ALG_HASH nameAlg;
     TPMA_OBJECT objectAttributes;
     TPM2B_DIGEST authPolicy;
     TPMU_PUBLIC_PARMS parameters;
     TPMU_PUBLIC_ID unique;
} TPMT_PUBLIC;
```

Where: * type and nameAlg are 16 bit TCG defined algorithms. * objectAttributes is a 32 bit field defining properties of the object, as shown below

```
typedef struct TPMA_OBJECT {
     unsigned Reserved_bit_at_0 : 1;
     unsigned fixedTPM : 1;
     unsigned stClear : 1;
     unsigned Reserved_bit_at_3 : 1;
     unsigned fixedParent : 1;
     unsigned sensitiveDataOrigin : 1;
     unsigned userWithAuth : 1;
     unsigned adminWithPolicy : 1;
     unsigned Reserved_bits_at_8 : 2;
     unsigned noDA : 1;
     unsigned encryptedDuplication : 1;
     unsigned Reserved_bits_at_12 : 4;
     unsigned restricted : 1;
     unsigned decrypt : 1;
     unsigned sign : 1;
     unsigned x509sign : 1;
     unsigned Reserved_bits_at_20 : 12;
} TPMA_OBJECT;
```

* authPolicy is the Policy Digest needed to authorize use of the object.

* Parameters are the object type specific public information about the key.

  - For key objects, this would be the key's public parameters.

* unique is the identifier for parameters

The size of the TPMT_PUBLIC is provided by the following structure:

```
typedef struct {
     UINT16     size;
     TPMT_PUBLIC publicArea;
} TPM2B_PUBLIC;
```

A.2.5.3. TPM2 Signatures

TPM2 signatures use a union where the first field (16 bits) identifies the signature scheme. The example below shows an RSA signature where TPMT_SIGNATURE->sigAlg will indicate to use TPMS_SIGNATURE_RSA as the signature.

```
typedef struct {
     TPMI_ALG_SIG_SCHEME sigAlg;
     TPMU_SIGNATURE signature;
} TPMT_SIGNATURE;

typedef struct {
     TPMI_ALG_HASH hash;
     TPM2B_PUBLIC_KEY_RSA sig;
} TPMS_SIGNATURE_RSA;
```

A.2.5.4. Attestation Key

The uniquely identifying TPM2 key is the Endorsement Key (the EK). As this is a privacy sensitive key, the EK is not directly used to attest to any TPM2 asset. Instead, the EK is used by an Attestation CA to create an Attestation Key (the AK). The AK is assumed trusted by the Verifier and is assume to be loaded in the TPM during the execution of the process described in the subsequent sections. The description of how to create the AK is outside the scope of this document.

A.2.5.5. Attester Processing

The only signed component is the TPM2B_ATTEST structure, which returns only the (key’s) Name and the signature computed over the Name but no detailed information about the key. As the Name is comprised of public information, the Name can be calculated by the Verifier but only if the Verify knows all the public information about the Key.

The Attester’s processing steps are as follows:

Using the TPM2 command TPM2_Certify obtain the TPM2B_ATTEST and TPMT_SIGNATURE structures from the TPM2. The signing key for TPMT_SIGNATURE is an Attention Key (or AK), which is assumed to be available to the TPM2 upfront. More details are provided in Appendix A.2.5.4

The TPM2 command TPM2_Certify takes the following input:

* TPM2 handle for Key (the key to be attested to)

* TPM2 handle for the AK (see Appendix A.2.5.4)

It produces the following output:

* TPM2B_ATTEST in binary format

* TPMT_SIGNATURE in binary format

Then, using the TPM2 command TPM2_ReadPublic obtain the Keys TPM2B_PUBLIC structure. While the Key’s public information can be obtained by the Verifier in a number ways, such as storing it from when the Key was created, this may be impractical in many situations. As TPM2 provided a command to obtain this information, this specification will include it in the TPM2 Attestation CSR extension.

The TPM2 command TPM2_ReadPublic takes the following input:

* TPM2 handle for Key (the key to be attested to)

It produces the following output:

* TPM2B_PUBLIC in binary format

A.2.5.6. Verifier Processing

The Verifier has to perform the following steps once it receives the Evidence:

* Verify the TPM2B_ATTEST using the TPMT_SIGNATURE.

* Use the Key’s "expected" Name from the provided TPM2B_PUBLIC structure. If Key’s "expected" Name equals TPM2B_ATTEST->attestationData then returned TPM2B_PUBLIC is the verified.

A.2.6. Sample CSR

This CSR demonstrates a certification request for a key stored in a TPM using the following structure:

```
CSR {
  attributes {
    id-aa-evidence {
      EvidenceBundle {
        Evidences {
          EvidenceStatement {
            type: tcg-attest-tpm-certify,
            stmt: <TcgAttestTpmCertify_data>
            hint: "tpmverifier.example.com"
          }
        },
        certs {
          akCertificate,
          caCertificate
        }
      }
    }
  }
}
```

Note that this example demonstrates most of the features of this specification:

* The data type is identified by the OID id-TcgCsrCertify contained in the EvidenceStatement.type field.

* The signed evidence is carried in the EvidenceStatement.stmt field.

* The EvidenceStatement.hint provides information to the Relying Party about which Verifier (software) will be able to correctly parse this data. Note that the type OID indicates the format of the data, but that may itself be a wrapper format that contains further data in a proprietary format. In this example, the hint says that software from the package "tpmverifier.example.com" will be able to parse this data.

* The evidence statement is accompanied by a certificate chain in the EvidenceBundle.certs field which can be used to verify the signature on the evidence statement. How the Verifier establishes trust in the provided certificates is outside the scope of this specification.

This example does not demonstrate an EvidenceBundle that contains multiple EvidenceStatements sharing a certificate chain.

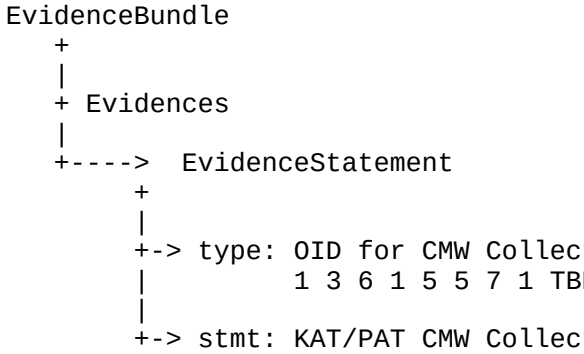
```
-----BEGIN CERTIFICATE REQUEST-----
MIINmzCCDIUCAQAwdTELMAkGA1UEBhMCWloxETAPBgNVBAgMCFByb3ZpbmNlMREw
DwYDVQQHDAhMb2NhbGl0eTETMBEGA1UECgwKaWV0Zi1sYW1wczEXMBUGA1UECwwO
aWV0Zi1sYW1wcy1jc3IxEjAQBgNVBAMMCXRlc3Qta2V5MTCCASIwDQYJKoZIhvcN
AQEBBQADggEPADCCAQoCggEBAN/RvsGNf32W6tIAU4QZgFPs98rPv4ed7QnB9aK/
+x8u+1qhiTNpNC2me0FsQEDvToROGhkxtiift2RKPaVR2hyF05tthjNDnfDaMqvk
NN24rmzTuVZE3Oz86zSWE+Lk3ZfWHROVb5ZTME/VOZdYMAvwyi2fRHa/1cK9F/61
WwIpY4qMlLsabSsSmyd8RJ+/g3exfCeCYJ73Cu90F0wNtYOTxVN5o4ELvJdElT99
QaC38TFvJ+yW94wQua2/4Lt6cx0I+NVDFHLMELwFydVdZspqFdEGp4X+i59hjD4A
FDPOyJJJiF84Zo7+Qf1tIbUCbFV+Rmvob7uCiOs8O3ZEL4kCAwEAAaCCCuEwggrd
BgsqhkiG9w0BCRACOzGCCswwggrIMIIC2jCCAtYGBWeBBRQBMIICsgSBkf9UQ0eA
FwAiAAs7ZAoM+pOXvuDS3cZXWSGXpKzEfn3C/aWh2zIlNmdIvQAEAP9VqgAAAAB9
6ovmAAAANwAAAABIBUBEwAVSCIAIgALRsPuEbWtPA+cXiHVz6zdm6DfOYX8urrR
WvLWAoEkW8MAIgALVNyWWGbUmLvXnu/dEowodTbdqKJlrhaYITil2pXo7ooEggEA
hnwVHoLE43BfVyozOqnx9VfsdS7U4ZOP4pS04vrfGCLegjXEHjxcMyU3DcJtd7sv
jw5/IxnLXLD/WXAe1H5XbOreOgYVejzMO16DPWBV2m7LOUvvwSsIRhHJaL5wUxfu
LZoWY6Up7Q+gFtDvoHfRrKsbeMRoOWwQulUok0PhZw0R4ZQyFrc4/rBr9kS9Vpmt
A+3IMuZ/qujraPqbC/zn1OE20+AtiKU6L9kJUtlejf8RchWn4ffi4ugK4u7RvMQS
/lGn+5G1IfVQY55iMKdlHa9nyzknQQBYopF2JP+sntC2Zf65IasWyE7NWOsQSbyr
6/OSLggeNf5gU8SrRfzaAgSCARYAAQALAAYAcgAAABAAEAgAAAAAAAEA39G+wY1/
fZbq0gBThBmAU+z3ys+/h53tCcH1or/7Hy77WqGJM2k0LaZ7QWxAQO9OhE4aGTG2
KJ+3ZEo9pVHaHIXTm22GM0Od8Noyq+Q03biubNO5VkTc7PzrNJYT4uTdl9YdE5Vv
llMwT9U5l1gwC/DKLZ9Edr/Vwr0X/rVbAiljioyUuxptKxKbJ3xEn7+Dd7F8J4Jg
nvcK73QXTA21g5PFU3mjgQu8l0SVP31BoLfxMW8n7Jb3jBC5rb/gu3pzHQj41UMU
cswQvAXJ1V1mymoV0Qanhf6Ln2GMPgAUM87IkkmIXzhmjv5B/W0htQJsVX5Ga+hv
u4KI6zw7dkQviRYXdHBtdmVyaWZpZXIuZXhhbXBsZS5jb20wggfmMIIEaTCCA1Gg
AwIBAgIUfX4oc7u4KUbyz61VtQN5g2A2QMQwDQYJKoZIhvcNAQELBQAwdzELMAkG
A1UEBhMCWloxETAPBgNVBAgMCFByb3ZpbmNlMREwDwYDVQQHDAhMb2NhbGl0eTET
MBEGA1UECgwKaWV0Zi1sYW1wczEXMBUGA1UECwwOaWV0Zi1sYW1wcy1jc3IxFDAS
BgNVBAMMC3Rlc3Qtcm9vdENBMB4XDTI0MTAyMTIwMTcxMloXDTI0MTEyMDIwMTcx
MlowczELMAkGA1UEBhMCWloxETAPBgNVBAgMCFByb3ZpbmNlMREwDwYDVQQHDAhM
b2NhbGl0eTETMBEGA1UECgwKaWV0Zi1sYW1wczEXMBUGA1UECwwOaWV0Zi1sYW1w
cy1jc3IxEDAOBgNVBAMMB3Rlc3QtYWswggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAw
ggEKAoIBAQDXjF+XF1wsywJx5e5MT1Q1TD8deZqxkQYGZM9TpW+rvFjnRdSDP88M
iLOPYcRIJQ+efHvo81o6IL7n0U0TSmaP63gaMCkOLr2BgNpBHGkfSbN2b16usBrQ
cYQF+o9NU5Itt/s7knvlhNiObOeWRBgtNPXeikyHycYjcZgoGd/UDvPRiRJ0pSru
SBDeS1x0uoTsvep0JSRBUiKh8d7D0H3UCmZZzVPzVBS1Z/Vl3a3HOjUgoAvXIBzu
kh/5BKdQSh5hfRnXi5v6lJGroWFWvsHVF2PLoOC2oaIrLZ3UVa05K4wVT/wKbi0O
cReufE9rK6CvmHEAUXi1cs+jynZfYaFDAgMBAAGjgfAwge0wgZ4GA1UdIwSBljCB
k6F7pHkwdzELMAkGA1UEBhMCWloxETAPBgNVBAgMCFByb3ZpbmNlMREwDwYDVQQH
DAhMb2NhbGl0eTETMBEGA1UECgwKaWV0Zi1sYW1wczEXMBUGA1UECwwOaWV0Zi1s
YW1wcy1jc3IxFDASBgNVBAMMC3Rlc3Qtcm9vdENBghR5pP+xxKsc67pLOqPiIZSU
4fgpzDAMBgNVHRMBAf8EAjAAMAsGA1UdDwQEAwIHgDAQBgNVHSUECTAHBgVngQUI
AzAdBgNVHQ4EFgQUH4QfxvcmPg9x21uQW5cfGyfxbMcwDQYJKoZIhvcNAQELBQAD
ggEBAC3Zz6B+u1H+72CG0j/s6lRPX/YP/DPjh5IIt9HdOJaWc5lsbCvgHSED7N/+
```

```
voCfCRf7IU2Oh5+2q9+9N5ARinXPmrsPZNyLR8vWkb27/hl9OTi0Ly7DtJMtUJTR
zq53dZc7kdTDNYpPnbIbanSOW3lSL5E173C8wyTsp/vQKteYTfmsDKw9hXHIU2eS
eUpZmKcHShXlbDEEbTuYLJMDASKmMCmPjIgJrSX/18wEoAgo2BVjjzwfhoc2SLnQ
dikvN6oa6Ee0zYRiImXpM7cuErC88jOr0quURA1U8fsQd8hYGRUk/6oPMXzIfZKs
z/yzAUQ570+mkdd2iFVlqTCQ9eUwggN1MIICXQIUeaT/scSrHOu6Szqj4iGUlOH4
KcwwDQYJKoZIhvcNAQELBQAwdzELMAkGA1UEBhMCWloxETAPBgNVBAgMCFByb3Zp
bmNlMREwDwYDVQQHDAhMb2NhbGl0eTETMBEGA1UECgwKaWV0Zi1sYW1wczEXMBUG
A1UECwwOaWV0Zi1sYW1wcy1jc3IxFDASBgNVBAMMC3Rlc3Qtcm9vdENBMB4XDTI0
MTAyMTIwMTcwOFoXDTI0MTEyMDIwMTcwOFowdzELMAkGA1UEBhMCWloxETAPBgNV
BAgMCFByb3ZpbmNlMREwDwYDVQQHDAhMb2NhbGl0eTETMBEGA1UECgwKaWV0Zi1s
YW1wczEXMBUGA1UECwwOaWV0Zi1sYW1wcy1jc3IxFDASBgNVBAMMC3Rlc3Qtcm9v
dENBMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAxPODF6ujxbdDWuXn
zlqzYIO4rpQKolKfKbOFUCB6SjdA5XzArLTSYKD3ZXhqm7unFkmHC2HtqArq5jgv
cQ2fzJeNGbcuyJYSwa9WJjJ5qY6gXEY+G7sFgZ5ZeEWGQ+zjrnuJh/PtlhJ2/R7w
tdC82DAULaxnFjOS6Xm6pUB8RaZEtZ6HwfPUXYgeK9IRG2CbEs8jkoBQTrSKpdpC
0myJrrS0PoTFClDpq61je7uQgU6b9IAOfXi1oX5NdYcfqxcPch4wqbkldKsjC/i7
xMcem8hnb3WUvAmbsLP1I0Fx8nb0ug/T2ED5U9tPBXbKgeD72aU2L9GNs+ypE9Az
bTB6cwIDAQABMA0GCSqGSIb3DQEBCwUAA4IBAQBFBMAGsP1xKq4R4bzbWnQ4K2H+
rYNi8UEQzxN6BiANi9+8hbLHfx6gFZlz+QxwVyH6oQnNrsVVDVjEYH4/yvy7NdOx
VitFfqOYwyaNeK5oXx1l5otOaObtwzB7RVQNSlipzEVW4RPsJmx/8F7yNLtdgooP
U0QzUSqDcoKK36E4O3s85xfyNlEdJ2vJcJ/ZZx5QQlnhNTf5bWlr3U9x6DebeAqs
+wvvepdaNzulHBXaKIqAgSx9N+Y22vj+vw8GO4L8HndDPMhdE/Ct1h1yygm65j6h
aCmQRUTKzj49q6W4NHG7iPea+7bgMq7G4LRo9DSFEfMWOkQeVUSPPiTsaAahMAsG
CSqGSIb3DQEBCwOCAQEA1aPYnm8suCRwMTC7kOdOjvVP2+2pHxsI5vnLfffFgsaN
yoOz97t6bAmQXPQCEcjCQZqAynuJQITBbf5OowrNCOL8YRLaFu2zUS8H+XJUIU0I
YSs3H8UyfeYo5VDKJClU/OcSzGgGm6J9JDQiHUuEFrqbpE19aSztpXrEH9YYP87A
NyW9vxpLse2rpE4akcp+V+C958KJEoYQc9EfjvM3LqLmzp7pvyUalv21BbOweK8V
IYVK7djq+LCmYwJwdKrOYWmrDyH8P+me8nPtk9BdcvW+sj2qu/opZmYEL4KAqAvi
BW5TzPFUgQhmMalis/J4WY3Q0tvOMXRQQZCmO2N4Pg==
-----END CERTIFICATE REQUEST-----
```

A.3. PSA Attestation Token in CSR

The Platform Security Architecture (PSA) Attestation Token is defined in [I-D.tschofenig-rats-psa-token] and specifies claims to be included in an Entity Attestation Token (EAT). [I-D.bft-rats-kat] defines key attestation based on the EAT format. In this section the platform attestation offered by [I-D.tschofenig-rats-psa-token] is combined with key attestation by binding the key attestation token (KAT) to the platform attestation token (PAT) with the help of the nonce. For details see [I-D.bft-rats-kat]. The resulting KAT-PAT bundle is, according to Section 5.1 of [I-D.bft-rats-kat], combined in a CMW collection [I-D.ietf-rats-msg-wrap].

The encoding of this KAT-PAT bundle is shown in the example below.



The value in EvidenceStatement->stmt is based on the KAT/PAT example from Section 6 of [I-D.bft-rats-kat] and the result of CBOR encoding the CMW collection shown below (with line-breaks added for readability purposes):

```
{
  "kat":
    h'd28443A10126A058C0A30A5820B91B03129222973C214E42BF31D68
      72A3EF2DBDDA401FBD1F725D48D6BF9C8171909C4A40102200121
      5820F0FFFA7BA35E76E44CA1F5446D327C8382A5A40E5F29745DF
      948346C7C88A5D32258207CB4C4873CBB6F097562F61D5280768C
      D2CFE35FBA97E997280DBAAAE3AF92FE08A101A40102200121582
      0D7CC072DE2205BDC1537A543D53C60A6ACB62ECCD890C7FA27C9
      E354089BBE13225820F95E1D4B851A2CC80FFF87D8E23F22AFB72
      5D535E515D020731E79A3B4E47120584056F50D131FA83979AE06
      4E76E70DC75C070B6D991AEC08ADF9F41CAB7F1B7E2C47F67DACA
      8BB49E3119B7BAE77AEC6C89162713E0CC6D0E7327831E67F3284
      1A',
  "pat":
    h'd28443A10126A05824A10A58205CA3750DAF829C30C20797EDDB794
      9B1FD028C5408F2DD8650AD732327E3FB645840F9F41CAB7F1B7E
      2C47F67DACA8BB49E3119B7BAE77AEC6C89162713E0CC6D0E7327
      831E67F32841A56F50D131FA83979AE064E76E70DC75C070B6D99
      1AEC08AD'
}
```

A.4. Confidential Compute Architecture (CCA) Platform Token in CSR

The Confidential Compute Architecture (CCA) Platform Token is described in [I-D.ffm-rats-cca-token] and is also based on the EAT format. Although the full CCA attestation is composed of Realm and Platform Evidence, for the purposes of this example only the Platform token is provided.

```
EvidenceBundle
   +
   |
   + Evidences
   |
   +---->  EvidenceStatement
        +
        |
        +-> type: OID for CCA Platform Attestation Toekn
        |         1 3 6 1 5 5 7 1 TBD
        |
        +-> stmt: CCA Platform Token
```

Although the CCA Platform Token follows the EAT/CMW format, it is untagged. This is because the encoding can be discerned in the CSR based on the OID alone. The untagged token based on a sample claim set is provided below:

(long lines wrapped for readability)

```
/ Sample Platform Claims Set in CDDL                          /
{
   / cca-platform-profile / 265:"tag:arm.com,2023:cca_platform#1.0.0"
   / arm-platform-challenge, SHA-256 calculation of 'RAK' / 10:
            h'c9cdc457ebe981d563b19b5a8e0e3cbef5b944d58e278c9c6779f
            77beb65bbd5'
   / arm-platform-lifecycle / 2395: h'3000' /secured/
   / arm-platform-sw-components / 2399: [ {1:"ROTFMC", 2:h'903a36d3a
            0a511ecac4548fee8601af54247c110ce220f680a0b274441729105' ,
            5:h'd4cf61e472d18c8e926ce0d44496674792587c88706e8a123b294
            c000895d9ea' },
      {1:"ROTFW", 2:h'59d4116525e974b5b62ffd7c4ffcbaa0b98e08263403aeb
            6638797132d2af959' , 5:h'd4cf61e472d18c8e926ce0d4449667479
            2587c88706e8a123b294c000895d9ea' } ]
   /arm-platform-id/ 256: h'  946338159d767f9f37098a00a60f133b6d57886f
            c656f5f9eed13760b4893fa1'
   /arm-platform-implementation-id/ 2396: h' 000000000000000000000000
            0000000000000000000000000000000000000001'
}

/ This is a full CWT-formatted token. The payload is a sample
/ Platform Claim Set. The main structure                       /
/ visible is that of the COSE_Sign1.                          /

61( 18( [
    h'a10126',  / protected headers  /
    {}, / empty unprotected headers /
    h' a4190109782374616 73a61726d2e636f6d2c323032333a6363615f706c746
```

```
      66f726d23312e392e300a580020c9cdc457ebe981d563b19b5a8e0e3cbef5b9
      44d58e278c9c6779f77beb65bbd519095b42300019095f82a30166524f54464
      d4302580020903a36d3a0a511ecac4548fee8601af54247c110ce220f680a0b
      27444172910505580020d4cf61e472d18c8e926ce0d44496674792587c88706
      e8a123b294c000895d9eaae0165524f5446575800200259d4116525e974b5b6
      2ffd7c4ffcbaa0b98e08263403aeb6638797132d2af95905580020d4cf61e47
      2d18c8e926ce0d44496674792587c88706e8a123b294c000895d9ea19010078
      20946338159d767f9f37098a00a60f133b6d57886fc656f5f9eed13760b4893
      fa11a095c5820000000000000000000000000000000000000000000000000000
      00000000000001' /payload/
      h'cbbfa929cb9b846cb5527d7ef9b7657256412a5f22a6e1a8d3a0c71145022
      100db4b1b97913b1cd9d6e11c1fadbc0869882ba6644b9db09d221f198e3286
      654b' /signature/
  ] ) )

  /Untagged serialized token/
  h'8443a10126a0590141a41901097823746167 3a61726d2e636f6d2c323032333a
  6363615f706c74666f726d23312e392e300a580020c9cdc457ebe981d563b19b5a
  8e0e3cbef5b944d58e278c9c6779f77beb65bbd519095b42300019095f82a30166
  524f54464d4302580020903a36d3a0a511ecac4548fee8601af54247c110ce220f
  680a0b27444172910505580020d4cf61e472d18c8e926ce0d44496674792587c88
  706e8a123b294c000895d9eaae0165524f5446575800200259d4116525e974b5b6
  2ffd7c4ffcbaa0b98e08263403aeb6638797132d2af95905580020d4cf61e472d1
  8c8e926ce0d44496674792587c88706e8a123b294c000895d9ea19010078209463
  38159d767f9f37098a00a60f133b6d57886fc656f5f9eed13760b4893fa11a095c
  5820000000000000000000000000000000000000000000000000000000000000
  015840cbbfa929cb9b846cb5527d7ef9b7657256412a5f22a6e1a8d3a0c7114502
  2100db4b1b97913b1cd9d6e11c1fadbc0869882ba6644b9db09d221f198e3286654b'

  Realm evidence can be included in a CMW bundle, similar to the PSA
  token.  In this case, the CSR is constructed as follows:

 EvidenceBundle
    +
    |
    + Evidences
    |
    +---->  EvidenceStatement
         +
         |
         +-> type: OID for CMW Collection
         |         1 3 6 1 5 5 7 1 TBD
         |
         +-> stmt: Realm Token/Platform Token CMW Collection or
                          Realm Claim Set/Platform Token CMW Collection
```

Appendix B. ASN.1 Module

```
   CSR-ATTESTATION-2025
     { iso(1) identified-organization(3) dod(6) internet(1) security(5)
     mechanisms(5) pkix(7) id-mod(0) id-mod-pkix-attest-01(TBDMOD) }

   CsrAttestation DEFINITIONS IMPLICIT TAGS ::= BEGIN

   EXPORTS ALL;

   IMPORTS

   Certificate, id-pkix
   FROM PKIX1Explicit-2009 -- from [RFC5912]
   { iso(1) identified-organization(3) dod(6) internet(1)
   security(5) mechanisms(5) pkix(7) id-mod(0)
   id-mod-pkix1-explicit-02(51) }

   CertificateChoices
     FROM CryptographicMessageSyntax-2010
       { iso(1) member-body(2) us(840) rsadsi(113549)
       pkcs(1) pkcs-9(9) smime(16) modules(0) id-mod-cms-2009(58) }

   EXTENSION, ATTRIBUTE, AttributeSet{}, SingleAttribute{}
   FROM PKIX-CommonTypes-2009 -- from [RFC5912]
   { iso(1) identified-organization(3) dod(6) internet(1)
   security(5) mechanisms(5) pkix(7) id-mod(0)
   id-mod-pkixCommon-02(57) }

   id-aa
     FROM SecureMimeMessageV3dot1
       { iso(1) member-body(2) us(840) rsadsi(113549)
       pkcs(1) pkcs-9(9) smime(16) modules(0) msg-v3dot1(21) }
     ;

   EVIDENCE-STATEMENT ::= TYPE-IDENTIFIER

   EvidenceStatementSet EVIDENCE-STATEMENT ::= {
      ... -- None defined in this document --
   }

   ATTESTATION-RESULT ::= TYPE-IDENTIFIER

   AttestationResultSet ATTESTATION-RESULT ::= {
      ... -- None defined in this document --
   }

   EvidenceStatement ::= SEQUENCE {
      type   EVIDENCE-STATEMENT.&id({EvidenceStatementSet}),
      stmt   EVIDENCE-STATEMENT.&Type({EvidenceStatementSet}{@type}),
```

```
      hint   IA5String OPTIONAL
   }

   AttestationResult ::= SEQUENCE {
      type   ATTESTATION-RESULT.&id({AttestationResultSet}),
      stmt   ATTESTATION-RESULT.&Type({AttestationResultSet}{@type}),
   }

   -- Arc for Evidence types
   id-aa-evidence OBJECT IDENTIFIER ::= { id-aa 59 }

   -- Arc for Attestation Result types
   id-aa-ar OBJECT IDENTIFIER ::= { id-aa (TBD2) }

   -- For PKCS#10 (Evidence)
   attr-evidence ATTRIBUTE ::= {
     TYPE EvidenceBundle
     COUNTS MAX 1
     IDENTIFIED BY id-aa-evidence
   }

   -- For CRMF (Evidence)
   ext-evidence EXTENSION ::= {
     SYNTAX EvidenceBundle
     IDENTIFIED BY id-aa-evidence
   }

   -- For PKCS#10 (Attestation Result)
   attr-ar ATTRIBUTE ::= {
     TYPE AttestationResultBundle
     COUNTS MAX 1
     IDENTIFIED BY id-aa-ar
   }

   -- For CRMF (Attestation Result)
   ext-ar EXTENSION ::= {
     SYNTAX AttestationResultBundle
     IDENTIFIED BY id-aa-ar
   }

   EvidenceBundle ::= SEQUENCE {
      evidences SEQUENCE SIZE (1..MAX) OF EvidenceStatement,
      certs SEQUENCE SIZE (1..MAX) OF CertificateChoices OPTIONAL,
         -- CertificateChoices MUST NOT contain the depreciated
         -- certificate structures or attribute certificates,
         -- see Section 10.2.2 of [RFC5652]
   }
```

```
   AttestationResultBundle ::= SEQUENCE SIZE (1..MAX)
                               OF AttestationResult

   END
```

B.1. TCG DICE Example in ASN.1

This section gives an example of extending the ASN.1 module above to carry an existing ASN.1-based Evidence Statement. The example used is the Trusted Computing Group DICE Attestation Conceptual Message Wrapper, as defined in [TCGDICE1.1].

```
   CsrAttestationDiceExample DEFINITIONS IMPLICIT TAGS ::= BEGIN

   IMPORTS

   tcg-dice-conceptual-message-wrapper FROM TcgDiceAttestation
   DiceConceptualMessageWrapper FROM TcgDiceAttestation
   tcg-dice-TcbInfo FROM TcgDiceAttestation
   DiceTcbInfo FROM TcgDiceAttestation
   EvidenceStatementSet FROM CsrAttestation
   ;

   tcgDiceCmwEvidenceStatementES EVIDENCE-STATEMENT ::= {
     DiceConceptualMessageWrapper IDENTIFIED BY tcg-dice-conceptual-
                                                    message-wrapper }

   tcgDiceTcbInfoEvidenceStatementES EVIDENCE-STATEMENT ::= {
     DiceTcbInfo IDENTIFIED BY tcg-dice-TcbInfo }
   -- where ConceptualMessageWrapper, tcg-dice-conceptual-message-
                                                          wrapper,
   -- DiceTcbInfo, and tcg-dice-TcbInfo
   -- are defined in DICE-Attestation-Architecture-Version-1.1-
   --                                      Revision-18_6Jan2024.pdf

   EvidenceStatementSet EVIDENCE-STATEMENT ::= {
     tcgDiceEvidenceStatementES,
     tcgDiceTcbInfoEvidenceStatementES
     ...
   }
   END

   TcgDiceAttestation DEFINITIONS AUTOMATIC TAGS ::= BEGIN

   EXPORTS ALL;

   tcg OBJECT IDENTIFIER ::= { 2 23 133 }
   tcg-dice OBJECT IDENTIFIER ::= { tcg platformClass(5) dice(4) }
```

```
   tcg-dice-TcbInfo OBJECT IDENTIFIER ::= { tcg-dice tcbinfo(1) }
   tcg-dice-endorsement-manifest-uri OBJECT IDENTIFIER ::= {
                                        tcg-dice manifest-uri(3) }
   tcg-dice-Ueid OBJECT IDENTIFIER ::= { tcg-dice ueid(4) }
   tcg-dice-MultiTcbInfo OBJECT IDENTIFIER ::= {tcg-dice
                                                multitcbinfo(5) }
   tcg-dice-UCCS-evidence OBJECT IDENTIFIER ::= {tcg-dice
                                                uccs-evidence(6) }
   tcg-dice-manifest-evidence OBJECT IDENTIFIER ::= {tcg-dice
                                               manifest-evidience(7) }
   tcg-dice-MultiTcbInfoComp OBJECT IDENTIFIER ::= {tcg-dice
                                                multitcbinfocomp(8) }
   tcg-dice-conceptual-message-wrapper OBJECT IDENTIFIER ::= {
                                                    tcg-dice cmw(9) }
   tcg-dice-TcbFreshness OBJECT IDENTIFIER ::= { tcg-dice
                                                tcb-freshness(11) }

   DiceConceptualMessageWrapper ::= SEQUENCE {
     cmw OCTET STRING
   }

   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,
     flagsMask [10]IMPLICIT OperationalFlagsMask OPTIONAL,
     integrityRegisters [11] IMPLICIT IrList 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),
     notReplayProtected (4),
```

```
    notIntegrityProtected (5),
    notRuntimeMeasured (6),
    notImmutable (7),
    notTcb (8),
    fixedWidth (31)
  }

  OperationalFlagsMask ::= BIT STRING {
    notConfigured (0),
    notSecure (1),
    recovery (2),
    debug (3),
    notReplayProtected (4),
    notIntegrityProtected (5),
    notRuntimeMeasured (6),
    notImmutable (7),
    notTcb (8),
    fixedWidth (31)
  }

  IrList ::= SEQUENCE SIZE (1..MAX) OF IntegrityRegister

  IntegrityRegister ::= SEQUENCE {
    registerName IA5String OPTIONAL,
    registerNum INTEGER OPTIONAL,
    hashAlg OBJECT IDENTIFIER,
    digest OCTET STRING
  }

  EndorsementManifestURI ::= SEQUENCE {
    emUri UTF8String
  }

  TcgUeid ::= SEQUENCE {
    ueid OCTET STRING
  }

  DiceTcbInfoSeq ::= SEQUENCE SIZE (1..MAX) OF DiceTcbInfo

  DiceTcbInfoComp ::= SEQUENCE SIZE (1..MAX) OF TcbInfoComp

  TcbInfoComp ::= SEQUENCE {
    commonFields [0] IMPLICIT DiceTcbInfo,
    evidenceValues [1] IMPLICIT DiceTcbInfoSeq
  }

  UccsEvidence ::= SEQUENCE {
    uccs OCTET STRING
```

```
}

Manifest ::= SEQUENCE {
  format ManifestFormat,
  manifest OCTET STRING
}

ManifestFormat ::= ENUMERATED {
  swid-xml    (0),
  coswid-cbor (1),
  coswid-json (2),
  tagged-cbor (3)
}

DiceTcbFreshness ::= SEQUENCE {
  nonce OCTET STRING
}
END
```

B.2. TCG DICE TcbInfo Example in CSR

This section gives an example of extending the ASN.1 module above to carry an existing ASN.1-based evidence statement. The example used is the Trusted Computing Group DiceTcbInfo, as defined in [TCGDICE1.1].

```
// SET of CSR Attributes
A0 82 00 8E
  // CSR attributes
  30 82 00 8A
    // OBJECT IDENTIFIER id-aa-evidence (1 2 840 113549 1 9 16 2 59)
    06 0B 2A 86 48 86 F7 0D 01 09 10 02 3B
      // SET -- This attribute
      31 79
        // EvidenceBundle ::= SEQUENCE
        30 75
          // EvidenceStatements ::= SEQUENCE SIZE (1..MAX)
                                           OF EvidenceStatement
          30 73
            // EvidenceStatement ::= SEQUENCE
            30 71
              // type: OBJECT IDENTIFIER tcg-dice-TcbInfo
              //                         (2.23.133.5.4.1)
              06 06 67 81 05 05 04 01
              // stmt: SEQUENCE
              30 4E
                // CONTEXT_SPECIFIC | version (02)
                // version = ABCDEF123456
```

```
                    82 0C 41 42 43 44 45 46 31 32 33 34 35 36
                    // CONTEXT_SPECIFIC | svn (03)
                    // svn = 4
                    83 01 04
                    // CONTEXT_SPECIFIC | CONSTRUCTED | fwids (06)
                    A6 2F
                    // SEQUENCE
                    30 2D
                      // OBJECT IDENTIFIER SHA256
                      06 09 60 86 48 01 65 03 04 02 01
                      // OCTET STRING
                      // fwid = 0x0000....00
                      04 20 00 00 00 00 00 00 00 00
                      00 00 00 00 00 00 00 00
                      00 00 00 00 00 00 00 00
                      00 00 00 00 00 00 00 00
                    // CONTEXT_SPECIFIC | vendorInfo (08)
                    // vendor info = 0x00000000
                    88 04 00 00 00 00
                    // CONTEXT_SPECIFIC | type (09)
                    // type = 0x00000000
                    89 04 00 00 00 00
                  // hint: IA5String "DiceTcbInfo.example.com"
                  0C 17 44 69 63 65 54 63 62 49 6e 66 6f
                  2e 65 78 61 6d 70 6c 65 2e 63 6f 6d

   // BER only
   a0 82 00 8c 30 82 00 88 06 0b 2a 86 48 86 f7 0d
   01 09 10 02 3b 30 79 31 77 30 75 30 73 30 71 06
   06 67 81 05 05 04 01 30 4e 82 0c 41 42 43 44 45
   46 31 32 33 34 35 36 83 01 04 a6 2f 30 2d 06 09
   60 86 48 01 65 03 04 02 01 04 20 00 00 00 00 00
   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
   00 00 00 00 00 00 00 00 00 00 00 88 04 00 00 00
   00 89 04 00 00 00 00 16 17 44 69 63 65 54 63 62
   49 6e 66 6f 2e 65 78 61 6d 70 6c 65 2e 63 6f 6d
```

Appendix C. Acknowledgments

This specification is the work of a design team created by the chairs of the LAMPS working group. The following persons, in no specific order, contributed to the work directly, participated in design team meetings, or provided review of the document.

Richard Kettlewell, Chris Trufan, Bruno Couillard, Jean-Pierre Fiset, Sander Temme, Jethro Beekman, Zsolt Rzsahegyi, Ferenc Pet, Mike Agrenius Kushner, Tomas Gustavsson, Dieter Bong, Christopher Meyer, Carl Wallace, Michael Richardson, Tomofumi Okubo, Olivier Couillard,

John Gray, Eric Amador, Darren Johnson, Herman Slatman, Tiru Reddy, James Hagborg, A.J. Stein, John Kemp, Daniel Migault and Russ Housley.

We would like to specifically thank Mike StJohns for his work on an earlier version of this draft.

We would also like to specifically thank Giri Mandyam for providing the appendix illustrating the confidential computing scenario, and to Corey Bonnell for helping with the hackathon scripts to bundle it into a CSR.

Finally, we would like to thank Andreas Kretschmer, Hendrik Brockhaus, David von Oheimb, and Thomas Fossati for their feedback based on implementation experience.

Authors' Addresses

Mike Ounsworth
Entrust Limited
2500 Solandt Road Suite 100
Ottawa, Ontario K2K 3G5
Canada
Email: mike.ounsworth@entrust.com

Hannes Tschofenig
Siemens
Germany
Email: Hannes.Tschofenig@gmx.net

Henk Birkholz
Fraunhofer SIT
Rheinstrasse 75
64295 Darmstadt
Germany
Email: henk.birkholz@sit.fraunhofer.de

Monty Wiseman
United States of America
Email: mwiseman32@acm.org

Ned Smith
Intel Corporation
United States

Email: ned.smith@intel.com