PKIX Key Attestation
draft-ietf-rats-pkix-key-attestation

Mike Ounsworth, Jean-Pierre Fiset , Hannes Tschofenig, Henk Birkholz, Monty Wiseman, Ned Smith

IETF 122 Bangkok

Agenda

1. Terminology
    1. What is “PKIX”?
    2. What is “Key Attestation”?
2. Our draft’s data model

ENTRUST

Terminology

PKIX Key Attestation

Intended for Hardware and software that operates within PKI applications.

Assertions about the **current** protection properties of a given application key.

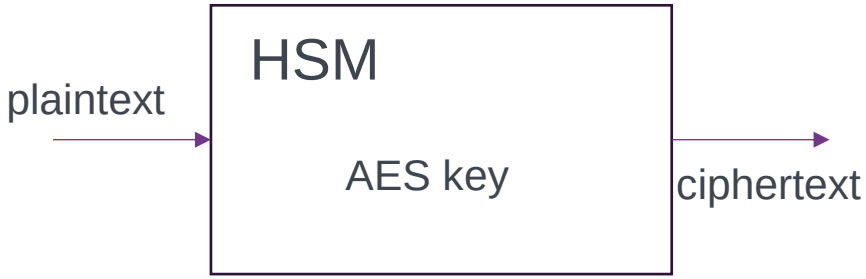
ENTRUST

What is “PKIX” ?

- “Public Key Infrastructure Extensions”
- The term generally means X.509, CSR, and supporting protocols:
  - enrollment protocols: CMP, SCEP, EST, CMC
  - revocation protocols: CRL, OCSP
- Sometimes CMS, S/MIME, and applications built on top of them, e.g., PDF signing, Windows code signing, etc. are considered part of PKIX.
- Sometimes people will include the entire set of PKCS standards as part of PKIX. In particular, PKCS#11 is the most common interface between PKI software applications and the hardware designed to support them.

- **We are using the term “PKIX” to mean Hardware and software that operates within PKI applications.**

Hardware Security Modules (HSMs)



- A device whose primary purpose is to hold cryptographic secret keys (symmetric or asymmetric) and offer interfaces to perform operations using those keys without exposing them.

ENTRUST

Two types of HSMs

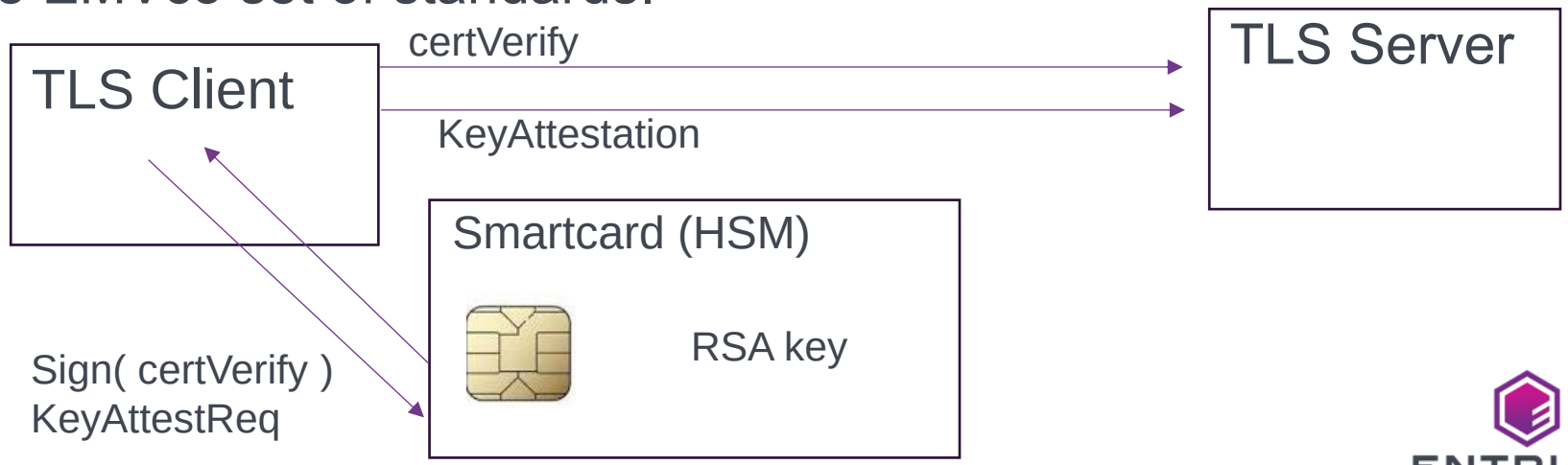
- HSMs generally split into two categories:
  - **Those intended for PKI applications**. (a.k.a. General-Purpose HSMs)
    HSMs that are generally designed to support applications that use X.509 certificates. They generally expose a PKCS#11 or similar interface and likely handle ASN.1 data structures such as X.509 certificates, SubjectPublicKeyInfo structures, etc.
  - **Those intended for Financial EMV applications.** (a.k.a. Payment HSMs)
    HSMs that generally expose interfaces and data key / ciphertext data structures compliant with the EMVco set of standards.

Two types of HSMs

- HSMs generally split into two categories:
  - **Those intended for PKI applications**. (a.k.a. General-Purpose HSMs)
    HSMs that are generally designed to support applications that use X.509 certificates. They generally expose a PKCS#11 or similar interface and likely handle ASN.1 data structures such as X.509 certificates, SubjectPublicKeyInfo structures, etc.
  - ~~**Those intended for Financial EMV applications.** (a.k.a. Payment HSMs)~~
    ~~HSMs that generally expose interfaces and data key / ciphertext data structures compliant with the EMVco set of standards.~~

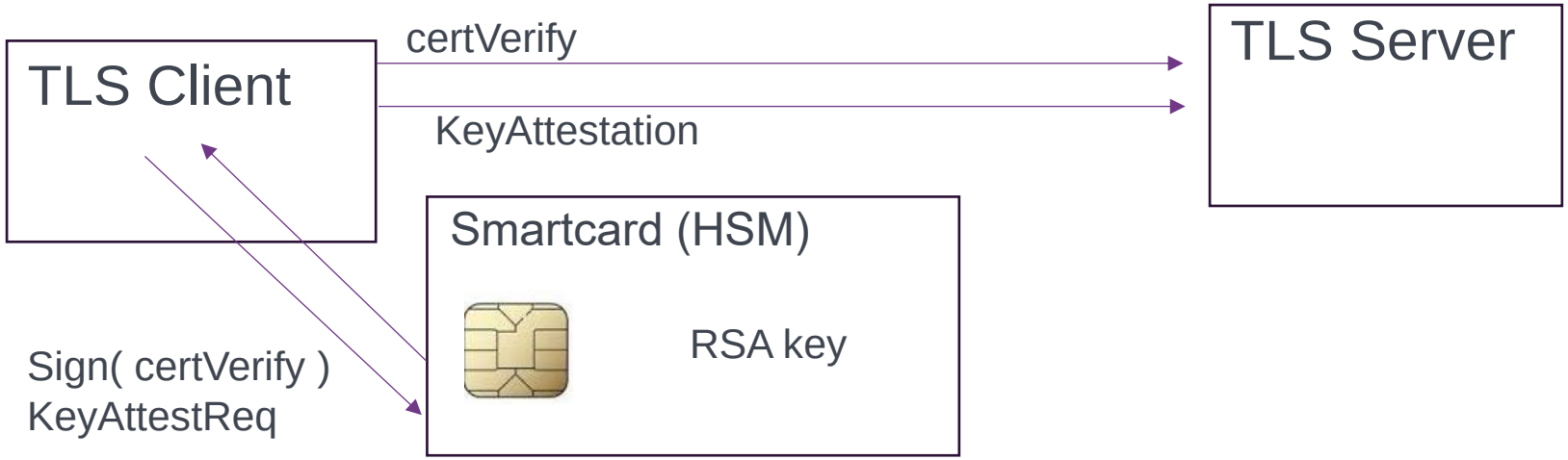
Our target usecase

Ex.:



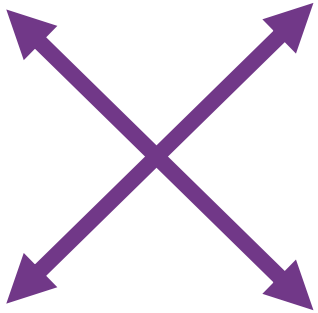
ENTRUST

HSMs intended for PKI applications



PKI Applications

- Certification Authority
  - Issuing certs,
  - Issuing CRLs, OCSP
- TLS server / client
- S/MIME email
- PDF signing
- Windows code signing
- 802.11 WiFi supplicant
- Physical building access ID badge cards
- Etc, etc, etc



PKIX HSM form-factor

- Rack-mount network HSM
- PCIe card HSM
- TPM
- Confidential Computing
- USB token
- Smartcard

Any HSM form factor can support any PKI application.

Conceptually they all look the same as the Outlook + smartcard diagram.

ENTRUST

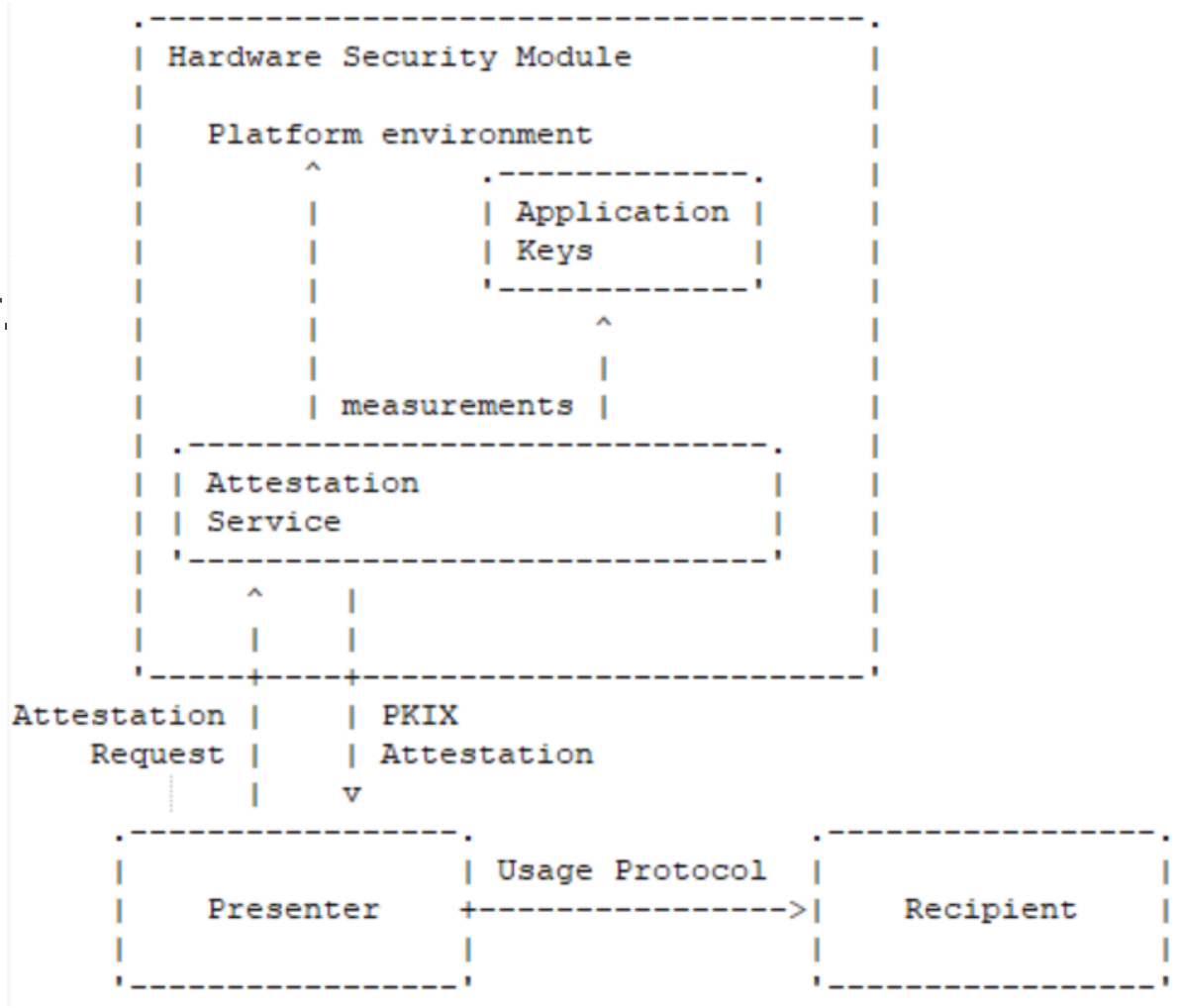
What is “key attestation” ?

- Evidence containing properties of the environment(s) in which the private keys are generated and stored. For example, a Relying Party may want to know whether a private key is stored in a hardware security module and cannot be exported in cleartext.

ENTRUST

What is “PKIX Key Attestation” ?

- A PKI Application wants to provide proof to its peer of the current properties of a given key.
- The HSM is the RATS Attester.
- The PKI Application is “the Presenter” (this role does not exist in the RATS Architecture).
- The PKI Application’s peer is “the Receiver” and is the RATS Verifier / RP.



ENTRUST

Design Goals

- Define a trusted assertion format for describing common vendor-neutral attributes of PKIX HSMs.
- MUST be ASN.1, as must as possible re-use structures and information models from PKCS#11 and LAMPS PKIX ASN.1 modules.
- The Attestation Key (AK) MUST be in an X.509 Certificate – no other (non-PKIX) certificate formats are supported.
- The PKI Application (Presenter) requests a fresh attestation from the HSM containing a set of attributes appropriate for the PKI Application, and then presents it to their peer over whatever protocol is appropriate (CSR-attest, attested-TLS, etc).

ENTRUST

Out-of-scope

- We ARE NOT defining a way to carry evidence, attestation results, endorsements, or anything else within an X.509v3 certificate.
  - CMW already has a small section with OIDs for this.
  - There are privacy and freshness implications here that may require defining Certificate Signing Policy profiles for when it is and is not appropriate for a CA to copy trusted assertion data into an issued X.509 certificate.

Our draft’s data model

Data Model – cryptographic envelope

The envelop structure is:

```
~~~asn.1
PkixAttestation ::= SEQUENCE {
    tbs TbsPkixAttestation,
    signatures SEQUENCE SIZE (0..MAX) of SignatureBlock
}

TbsPkixAttestation ::= SEQUENCE {
    version INTEGER,
    reportedEntities SEQUENCE SIZE (1..MAX) OF ReportedEntity
}

SignatureBlock ::= SEQUENCE {
   certChain SEQUENCE of Certificate,
   signatureAlgorithm AlgorithmIdentifier,
   signatureValue OCTET STRING
}
~~~
```

```
~~~asn.1
ReportedEntity ::= SEQUENCE {
    entityType          OBJECT IDENTIFIER,
    reportedAttributes SEQUENCE SIZE (1..MAX) OF ReportedAttribute
}
~~~
```

Is loosely CMS-like.

ENTRUST

Transaction Entities

- Only one Transaction Entity per PKIXAttestation.

Attribute	AttributeValue	Reference	Multiple Allowed	Request Contains a Value	Description
nonce	bytes	RFCthis	No	**MUST**	Repeats a "nonce" provided during the atttestation request.
timestamp	time	[I-D.ietf-rats-eat]	No	**MUST NOT**	The time at which this attestation was generated. Corresponds to EAT IAT claim.

ENTRUST

Platform Entities

- Only one Platform Entity per PKIXAttestation.

Attribute	AttributeValue	Reference	Multiple Allowed	Request Contains a Value	Description
vendor	utf8String	RFCthis	No	**MUST NOT**	A human-readable string by which the vendor identifies themself.
oemid	bytes	[I-D.ietf-rats-eat]	No	**MUST NOT**	The EAT OEM ID as defined in [I-D.ietf-rats-eat].
hwmodel	utf8String	[I-D.ietf-rats-eat]	No	**MUST NOT**	Model or product line of the hardware module.
hwserial	utf8String	RFCthis	No	**MUST NOT**	Serial number of the hardware module, often matches the number engraved or stickered on the case.

ENTRUST

Platform Entities

- Only one Platform Entity per PKIXAttestation.

fipsboot	bool	[FIPS.140-3]	No	**MUST NOT**	Indicates whether the devices is currently running in FIPS mode.
fipsver	utf8String	[FIPS.140-3]	No	**MUST NOT**	Indicates the version of the FIPS CMVP standard that is being enforced. At time of writing this is typically "FIPS 140-2" or "FIPS 140-3".
fipslevel	int	[FIPS.140-3]	No	**MUST NOT**	Indicates the FIPS Level to which the device is currently operating in compliance with.
envid	utf8String	RFCthis	Yes	**MAY**	An environment ID, which will typically be a URI, UUID, or similar.
envdesc	utf8String	RFCthis	Yes	**MUST NOT**	Further description of the environment.

ENTRUST

Key Entities

- Any number of Key Entities per PKIXAttestation.
  - Ex.: you could ask the HSM to attest a single key, a set of keys, or all the keys that it contains.

Attribute	AttributeValue	Reference	Multiple Allowed	Request Contains a Value	Description
identifier	utf8String	RFCthis	Yes	**MAY**	Identifies the subject key, with a vendor-specific format which could be numeric, UUID, or other textual identifier.
spki	bytes	RFCthis	No	**MAY**	A complete DER-encoded SubjectPublicKeyInfo representing the public key associated with the asymetric key pair being attested.
purpose	bytes	[PKCS11]	No	**MAY**	Defines the intended usage for the key.
extractable	bool	[PKCS11]	No	**MAY**	Indicates if the key is able to be exported from the module. Corresponds directly to PKCS#11 CKA_EXTRACTABLE.

ENTRUST

Key Entities

- Any number of Key Entities per PKIXAttestation.
  - Ex.: you could ask the HSM to attest a single key, a set of keys, or all the keys that it contains.

sensitive	bool	[PKCS11]	No	**MAY**	Indicates that the key cannot leave the module in plaintext. Corresponds directly to PKCS#11 CKA_SENSITIVE.
never-extractable	bool	[PKCS11]	No	**MAY**	Indicates if the key was able to be exported from the module. Corresponds directly to PKCS#11 CKA_NEVER_EXTRACTABLE.
local	bool	RFCthis	No	**MAY**	Indicates whether the key was generated locally or imported.
expiry	time	RFCthis	No	**MAY**	Defines the expiry date or "not after" time for the key.
protection	bytes	RFCthis	No	**MAY**	Indicates any additional key protection properties.

ENTRUST

Attestation Request format

- The same as the general PKIXAttestation, except:
  - Could carry a SignatureBlock acting as a client signature on the request, or could be an un-signed TbsPkixAttestation.
  - MUST contain a single entity: a Request Entity.
  - Some request attributes carry a value (ex.: a KeyID or Public Key if you want to request attestation of that key)
  - Some request attributes are un-valued (indicates that the HSM should collect a measurement for that attribute).

Next steps

- Continue iterating ASN.1 data model.
- Continue iterating terminology.
- Continue explanatory text.
- Continue hackathon on reference implementation [1]

- … basically, this is an -00 and needs more iteration.

https://github.com/ietf-rats-wg/key-attestation/tree/main/sampledata/idea3

ENTRUST