Open Trust Protocol

draft-pei-opentrustprotocol-0.txt

M. Pei, N. Cook, M. Yoo,
A. Atyeo, H. Tschofenig

Background Context

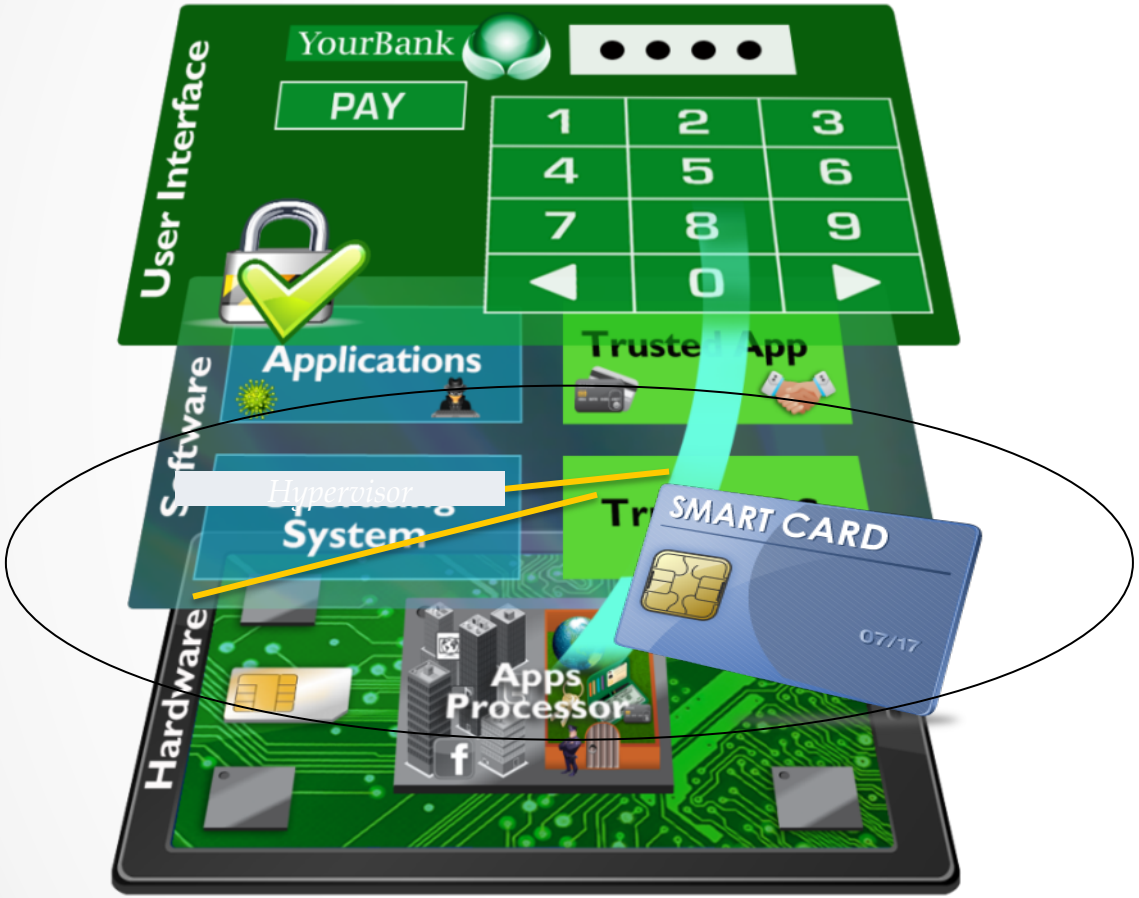
- Challenges and Proposals

The Challenge

How to access hardware security when fragmentation is growing?

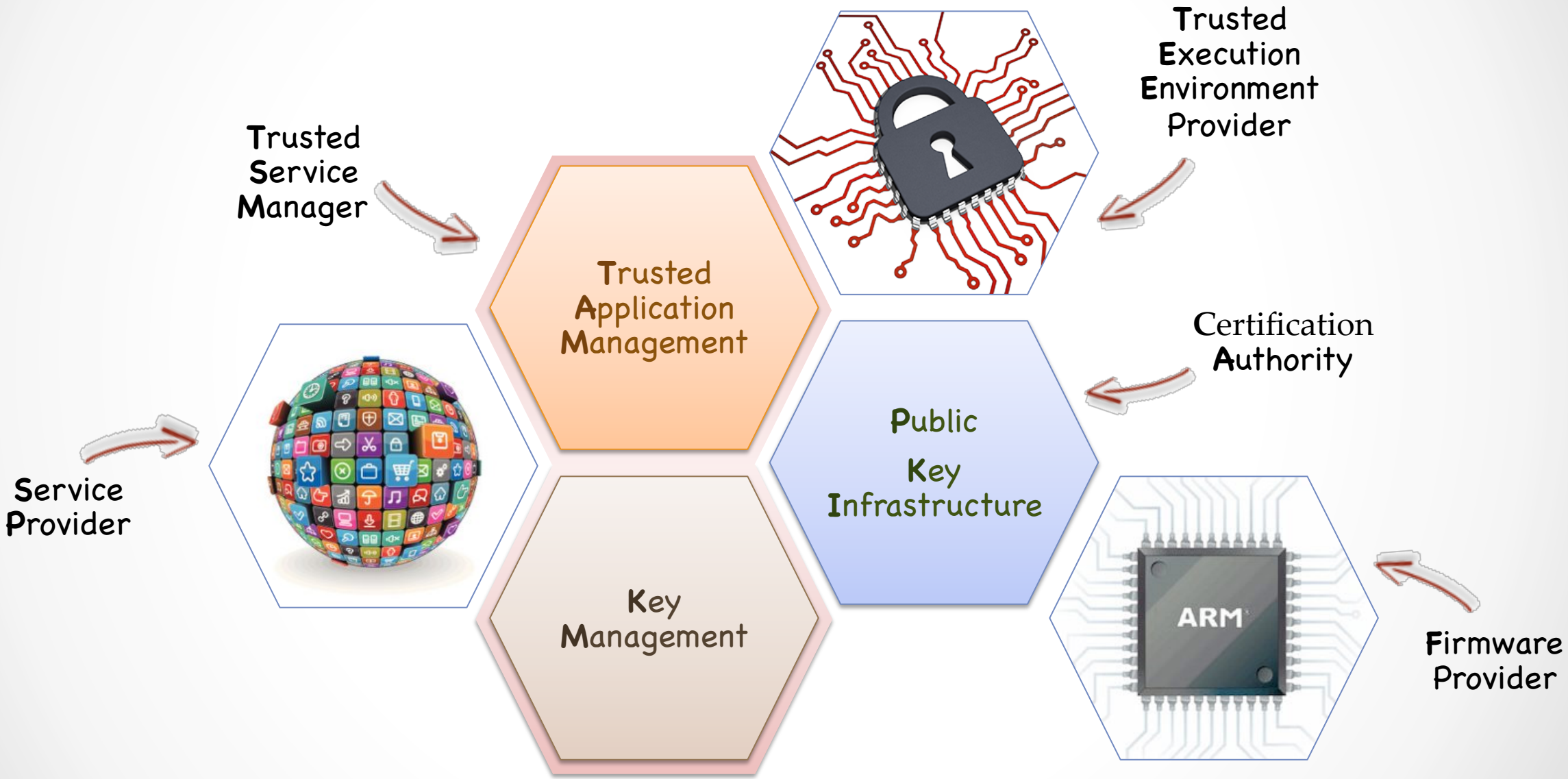
- From a Service Provider point of view there is a gap to be bridged between devices with hardware security and a wish to push keys/ Trusted Apps to devices
- Fragmentation is growing – IOT will accelerate that fragmentation
- Devices have hardware based Trusted Execution Environments (but they do not have a standard way of managing those security domains/TAs e.g. how to install / delete a Trusted App?

Today's Situation

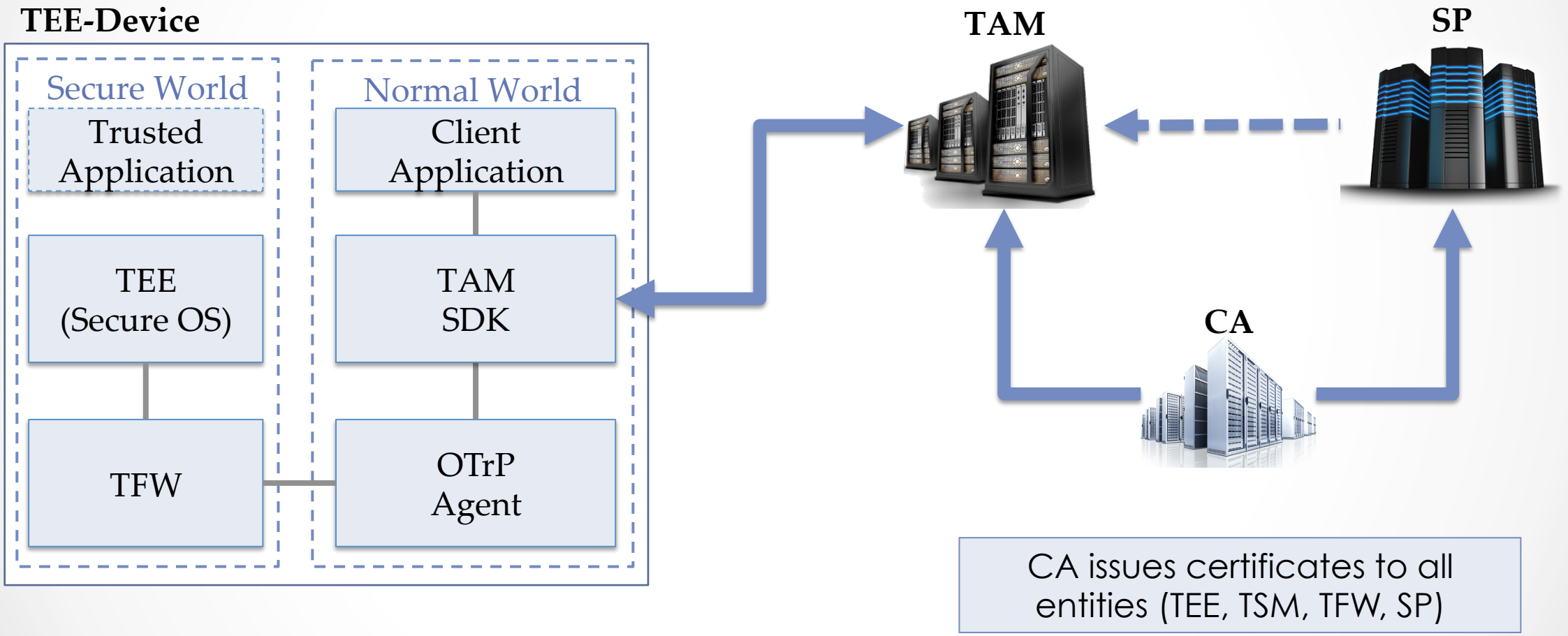


- Platforms are built with layers of hardware based security
- Devices are designed to be secure by design
- Few platforms enable Trusted App Management OTA
- Standards lacking

Ecosystem


Trusted Service Manager
Trusted Execution Environment Provider
Trusted Application Management
Certification Authority
Public Key Infrastructure
Service Provider
Key Management
ARM
Firmware Provider

Open Trust Protocol



CA issues certificates to all entities (TEE, TSM, TFW, SP)

OTrP Goals

- Simplify secure provisioning of devices
- Designed to work with any "hardware secure environment"
  - Starting with TEE, which is widely used in the mobile space.
- Creating a specification for industry use
- Focus on re-use of existing schemes (CA and PKI) and ease of implementation

Constraints

- Trust anchors are configured - not hard-coded
- The end-point is what it claims to be
  - Cryptography based authentication with certified asymmetric device keys
  - Device keys are configured for use with secure access mode only
- Device attestation
  - Attestation with public key digital signatures for remote integrity checks of hardware based credentials and applications
- The end-point is running verified code
  - Secure boot with signed code verified against trusted root CA in ROM
  - TEE signature includes own certificate for local code and identity integrity checks

OTrP Spec

OTrP Spec History

- Joint effort from OTrP Alliance founding member companies ARM, Intercede, Solacia, Symantec
- A message protocol to define the management of trusted applications and security-related configuration parameters
- Specification contributed to the IETF in July. Previously developed in alliance.

Technical Spec Content

- **Define relationship between entities**
- **Define JSON messages for trust and remote TA management** between a TSM and TEE
  - Messages for device attestation (device integrity check) by a TSM and a device to trust a TSM
  - Messages for Security domain management and TA management
  - Network communication among entities are left to implementations
- **Define an OTrP Agent in REE (Rich Execution Environment)**
  - Necessary component from REE of a device to relay message exchanges between a TSM and TEE
- **Use standard PKI artifacts and algorithms**
- **Use standard JSON messages and JSON security RFCs**

Keys

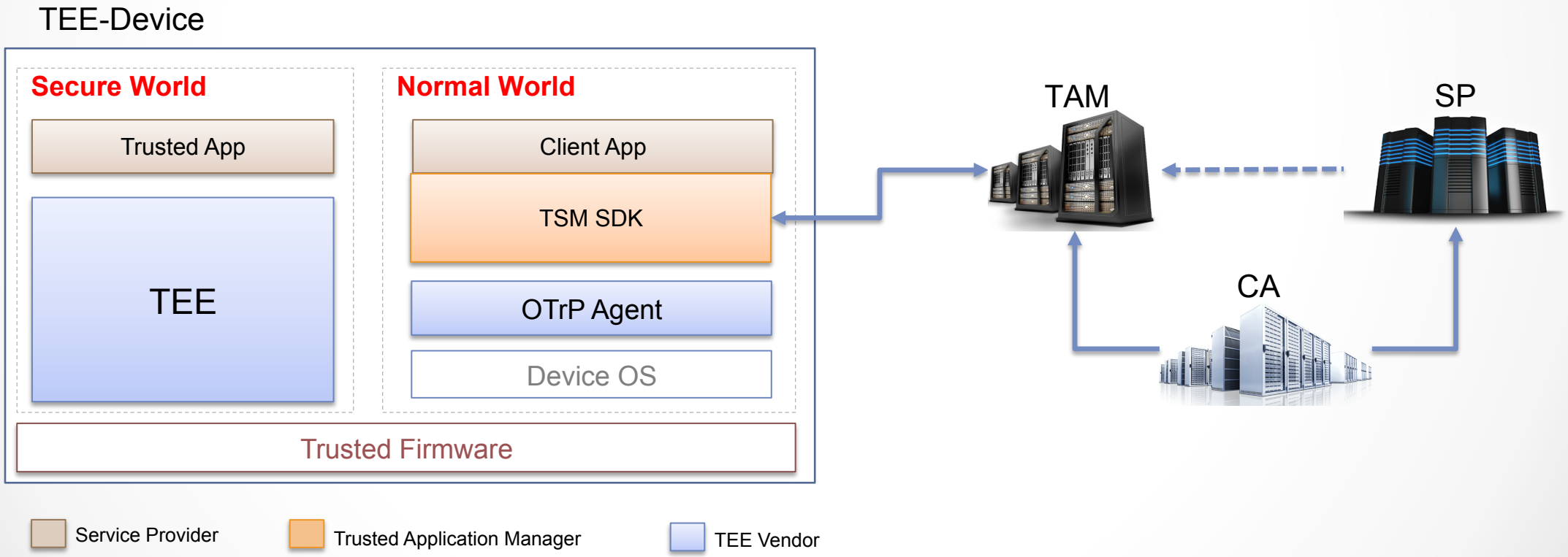
	Certificate Authority	Service Provider	TAM	Device TEE	
Keys	 **CA Certificate**	 SP Key pair and Certificate	 TAM Key pair and Certificate	 **TEE Key pair and Certificate**	 **TFW Key pair and Certificate**
Derived Keys		**SP Anonymous Public Key** 		 **SP Anonymous Key (SP AIK)**	
Trust Anchors			 **Trust Anchors:** trusted Root CA list of TEE Cert	 **Trust Anchors:** trusted Root CA list of TAM Cert	 **Trust Anchors:** trusted Root CA list of TEE Cert
Usage	**** Key pair and Certificate:*** *used to issue certificate*	**** Key pair and Certificate:*** *used to sign a TA*	**** Key pair and Certificate:*** *sign OTrP requests to be verified by TEE*	**** Key pair and Certificate:*** *device attestation to remote TAM and SP.* **** SP AIK*** *to encrypt TA binary data*	**** Key pair and Certificate:*** *evidence of secure boot and trustworthy firmware*

Key Usage

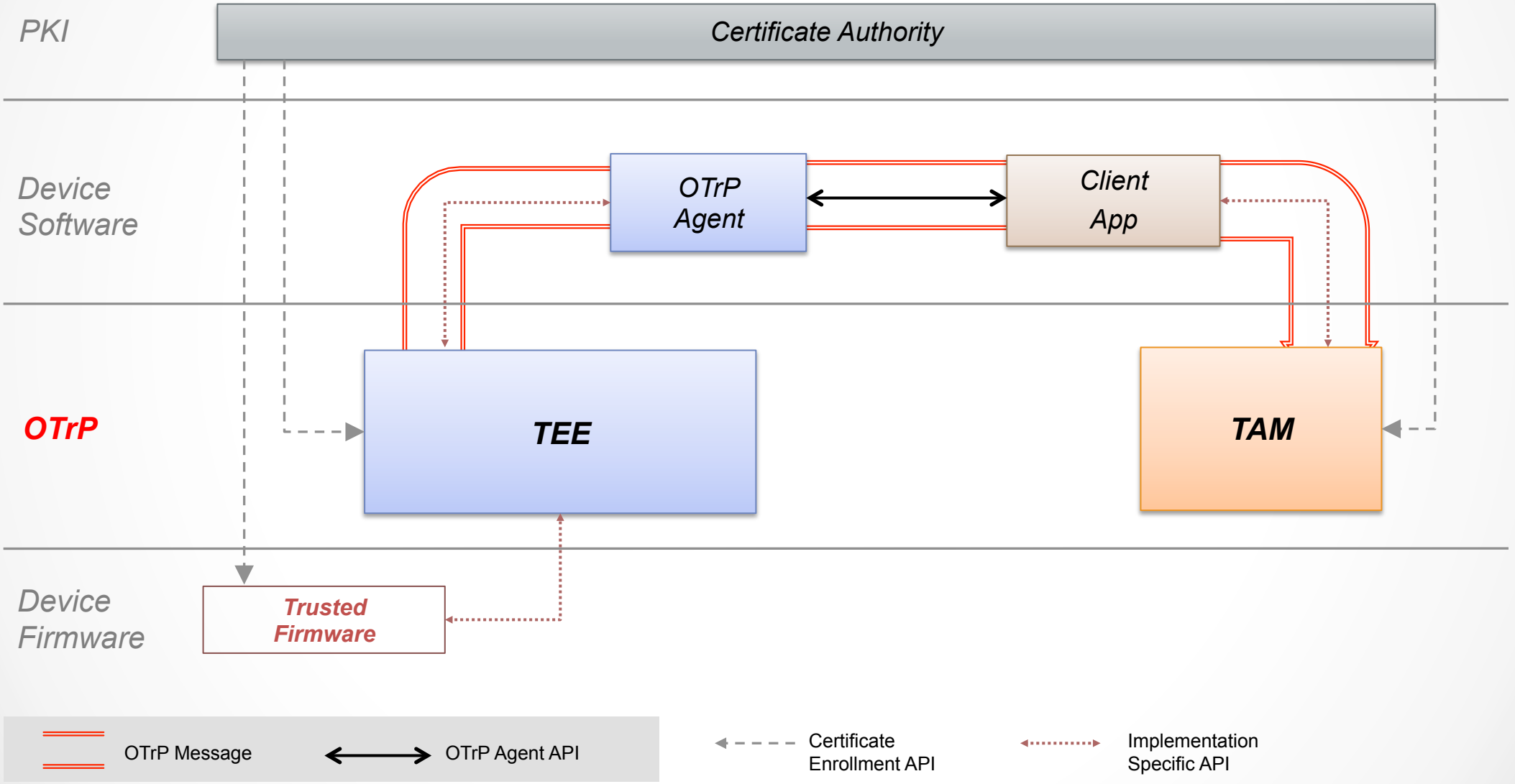
*** AIK: Attestation Identity Key, TFW: Trusted Firmware**

OTrP System Architecture

- CA issues certificates to all OTrP Components (TEE, TAM, TFW, SP)
- TAM vendor provides the SDK to communicate with TAM from Client Application
- TAM communicates with OTrP Agent to relay the OTrP message between TAM and TEE

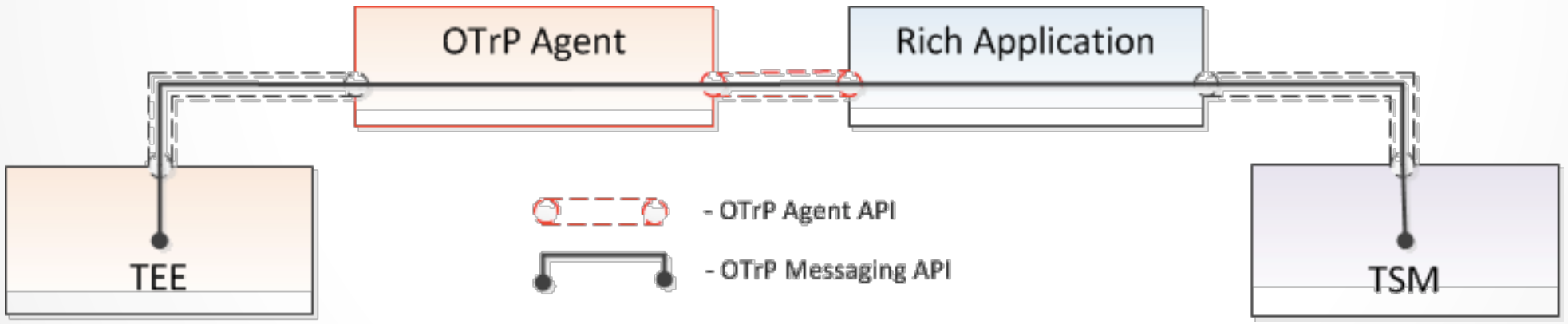


OTrP Spec Scope



OTrP Agent

- Responsible for routing OTrP Messages to the appropriate TEE
- Most commonly developed and distributed by TEE vendor
- Implements an interface as a service, SDK, etc.



OTrP Operations and Messages

✓ Remote Device Attestation

Command	Descriptions
GetDeviceState	• Retrieve information of TEE device state including SD and TA associated to a TAM

✓ Security Domain Management

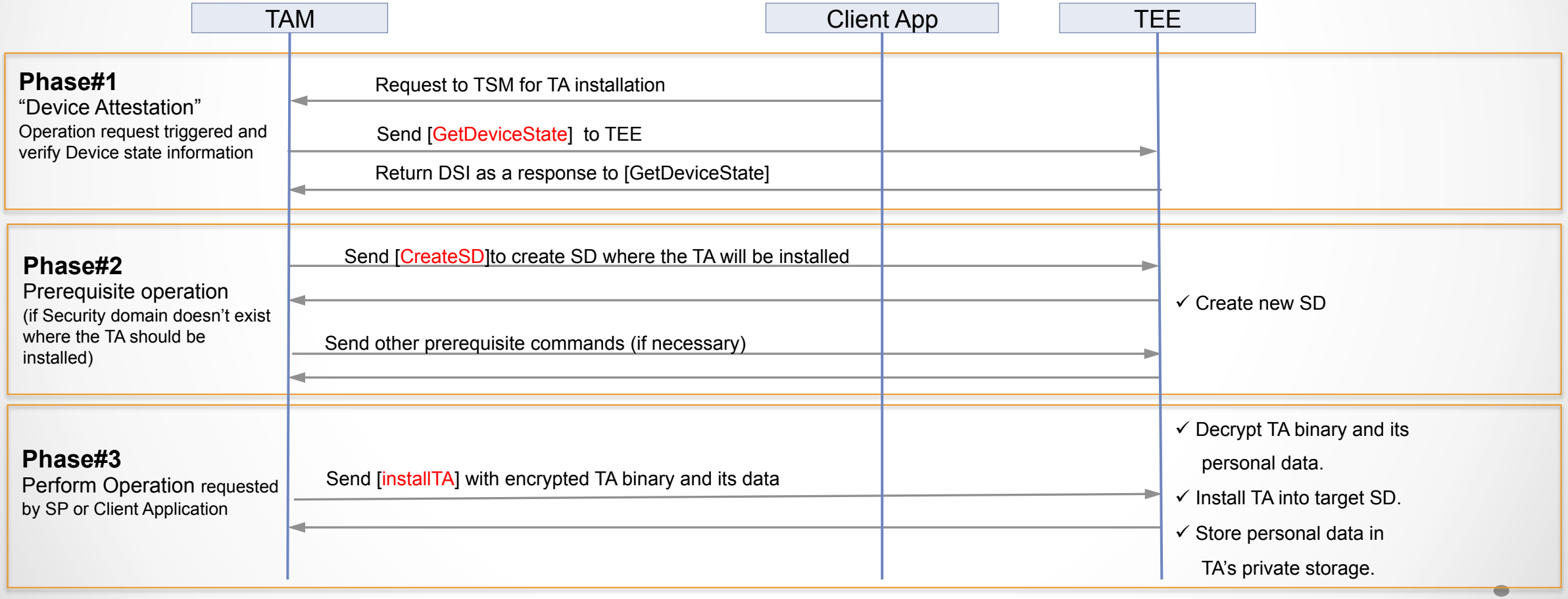
Command	Descriptions
CreateSD	• Create SD in the TEE associated to a TAM
UpdateSD	• Update sub-SD within SD or SP related information
DeleteSD	• Delete SD or SD related information in the TEE associated to a TAM

✓ Trusted Application Management

Command	Descriptions
InstallTA	• Install TA in the SD associated to a TAM
UpdateTA	• Update TA in the SD associated to a TAM
DeleteTA	• Delete TA in the SD associated to a TAM

Protocol Flow

- Security of the Operation Protocol is enhanced by applying the following three Measures:
  - ✓ Verifies validity of Message Sender's Certificate
  - ✓ Verifies signature of Message Sender to check immutability
  - ✓ Encrypted to guard against exposure of Sensitive data



JSON Message Security and Crypto Algorithms

- Use JSON signing and encryption RFCs
  - RFC 7515, JSON Web Signature (JWS)
  - RFC 7516, JSON Web Encryption (JWE)
  - RFC 7517, JSON Web Key (JWK)
  - RFC 7518, JSON Web Algorithms (JWA)
- Supported encryption algorithms
  - A128CBC-HS256
  - A256CBC-HS512
- Supported signing algorithms
  - RS256 (RSA 2048-bit key)
  - ES256 (ECC P-256)
- Examples
  - {"alg":"RS256"}
  - {"alg":"ES256"}
  - {"enc":"A128CBC-HS256"}

OTrP JSON Message Format and Convention

```
{
   “<name>[Request | Response]”: {
      "payload": "<payload contents of <name>TBS[Request | Response]>",
      "protected":"<integrity-protected header contents>",
      "header":  <non-integrity-protected header contents>,
      "signature":"<signature contents>"
   }
}
```

For example:

- CreateSDRequest
- CreateSDResponse

OTrP JSON Sample Message: GetDeviceState

```
{
  "GetDeviceStateTBSRequest": {
     "ver": "1.0",
      "rid": "<Unique request ID>",
      "tid": "<transaction ID>",
      "ocspdat": "<OCSP stapling data of TSM certificate>",
      "icaocspdat": "<OCSP stapling data for TSM CA certificates>",
      "supportedsigalgs": "<comma separated signing algorithms>"
   }
}
```

```
{
    "GetDeviceStateRequest": {
      "payload":"<BASE64URL encoding of the GetDeviceStateTBSRequest JSON above>",
      "protected": "<BASE64URL encoded signing algorithm>",
      "header": {
          "x5c": "<BASE64 encoded TSM certificate chain up to the root CA certificate>"
      },
      "signature":"<signature contents signed by TSM private key>"
    }
}
```

OTrP JSON Sample Message: CreateSD Request

```
{
   "GetDeviceStateTBSRequest": {
      "ver": "1.0",
      "rid": "<Unique request ID>",
      "tid": "<transaction ID>",
      "ocspdat": "<OCSP stapling data of TSM certificate>",
      "icaocspdat": "<OCSP stapling data for TSM CA certificates>",
      "supportedsigalgs": "<comma separated signing algorithms>
   }
}
```

OTrP JSON Sample Message: CreateSD Response

```
{
  "CreateSDTBSResponse": {
     "ver": "1.0",
     "status": "<operation result>",
     "rid": "<the request ID received>",
     "tid": "<the transaction ID received>",
     "content": ENCRYPTED {
        "reason":"<failure reason detail>", // optional
        "did": "<the device id received from the request>",
        "sdname": "<SD name for the domain created>",
        "teespaik": "<TEE SP AIK public key, BASE64 encoded>",
        "dsi": "<Updated TEE state, including all SD owned by tthis TSM>"
     }
  }
}
```

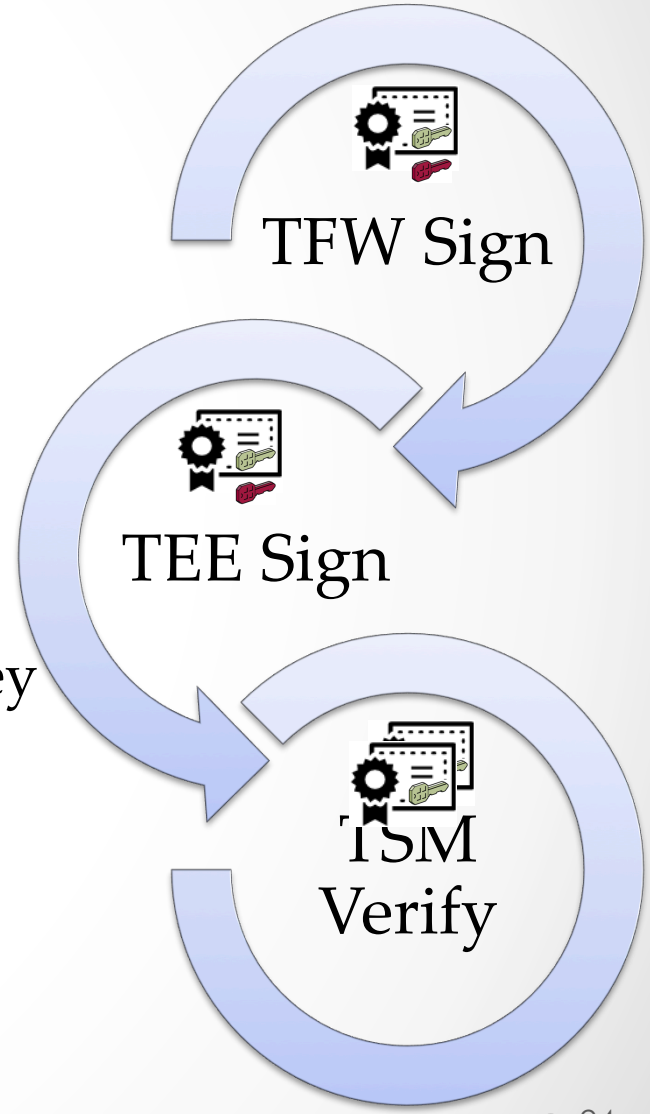
Appendix A: More Message Details

GetDeviceState

TSM acts as a trust broker by using the OTrP device attestation mechanism to asses TFW and TEE authenticity and current state prior to a management command, subject to SP controlled policy

- `GetDeviceStateRequest`
  - Signed by TSM
  - Contains TSM identifying and status (OCSP) information
  - Typically triggered by an SP Rich Application
    - Note: TSM implementation API not in OTrP scope

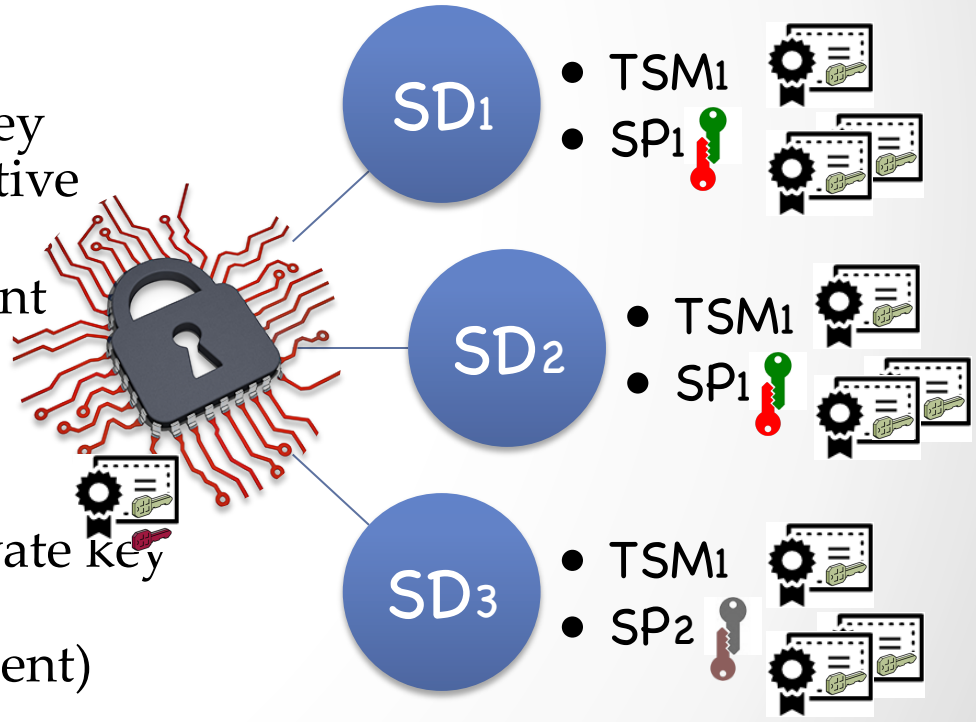
- `GetDeviceStateResponse`
  - Signed by TEE and encrypted to requesting TSM private key
  - Encapsulates TFW signed data
  - Contains TEE identifying information and a list of all SDs and TAs managed by the requesting TSM
  - May include device generated, anonymous Public Keys assigned by TEE to all registered SPs (if SD present)



Security Domain Management: CreateSD

Security boundaries on device are established with unique Security Domains assigned by TEE in response to TSM commands issued on behalf of SPs. Each SD is assigned to a TSM and an SP.
To protect the TEE identity every SD contains a unique device generated anonymous identity key pair for independent SP device attestation.

- `CreateSDRequest`
  - Signed by TSM and encrypted to target TEE private key
  - Includes TSM and SP identity information and respective certificates
  - "Last known configuration" hash is included to prevent race conditions
- `CreateSDResponse`
  - Signed by TEE and encrypted to requesting TSM private key
  - May include a device generated, anonymous public key assigned by TEE to the SP (if not already present)

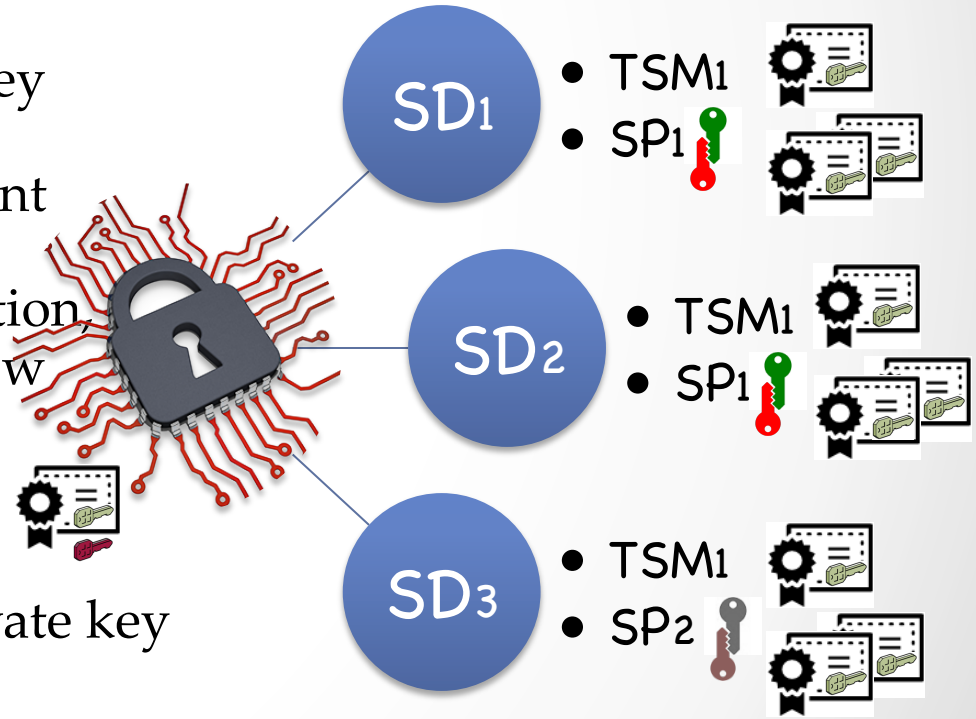


Security Domain Management: UpdateSD

Security boundaries on device are established with unique Security Domains assigned by TEE in response to TSM commands issued on behalf of SPs. Each SD is assigned to a TSM and an SP.
To protect the TEE identity every SD contains a unique device generated anonymous identity key pair for independent SP device attestation.

- `UpdateSDRequest`
  - Signed by TSM and encrypted to target TEE private key
  - Includes TSM and SP identity information
  - "Last known configuration" hash is included to prevent race conditions
  - Allowed updates: change SD and SP identity information, add new SP certificate, remove an old SP certificate, renew anonymous SP key pair

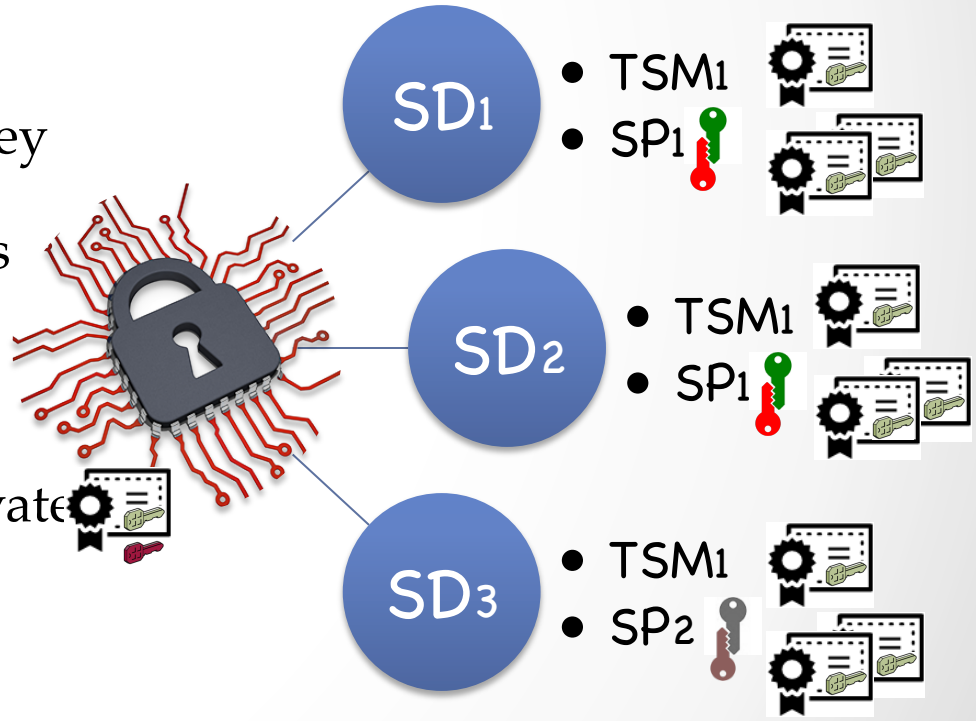
- `UpdateSDResponse`
  - Signed by TEE and encrypted to requesting TSM private key
  - May include a new device generated, anonymous public key assigned by TEE to the SP (if requested)



Security Domain Management: DeleteSD

Security boundaries on device are established with unique Security Domains assigned by TEE in response to TSM commands issued on behalf of SPs. Each SD is assigned to a TSM and an SP.
To protect the TEE identity every SD contains a unique device generated anonymous identity key pair for independent SP device attestation.

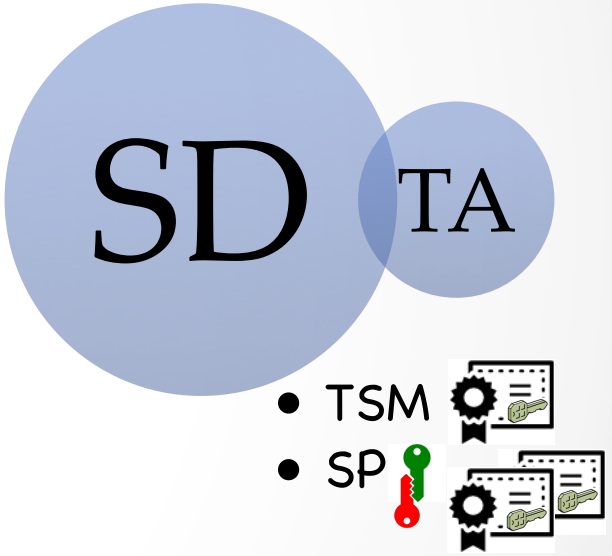
- `DeleteSDRequest`
  - Signed by TSM and encrypted to target TEE private key
  - Includes TSM identity information
  - An optional flag to delete all TAs in an SD if present is allowed, otherwise return error if not empty
- `DeleteSDResponse`
  - Signed by TEE and encrypted to requesting TSM private



TA Management: InstallTA

TSM is managing the full lifecycle of a Trusted Application in the Security Domain created on behalf of the Service Provider

- `InstallTARequest`
  - Signed by TSM and encrypted to target TEE private key
  - Includes TSM, SP and TA identity information
  - "Last known configuration" hash is included to prevent race conditions
  - Includes TA binary and personalization data encrypted to the SP anonymous private key

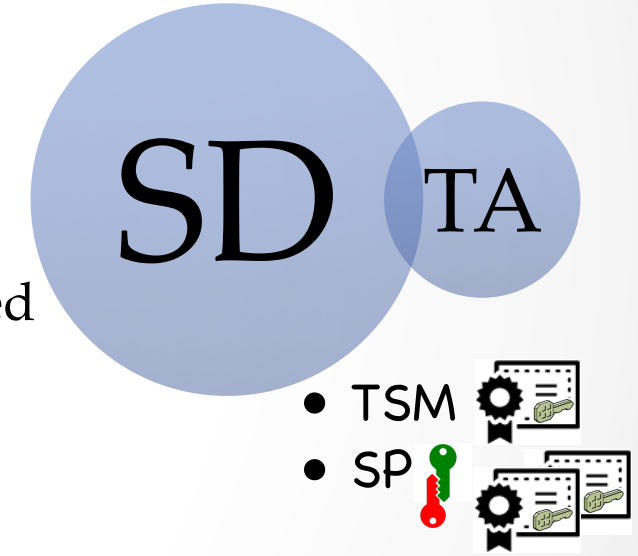


- `InstallTAResponse`
  - Signed by TEE and encrypted to requesting TSM private key

TA Management: UpdateTA

TSM is managing the full lifecycle of a Trusted Application in the Security Domain created on behalf of the Service Provider

- `UpdateTARequest`
  - Signed by TSM and encrypted to target TEE private key
  - Includes TSM, SP and TA identity information
  - "Last known configuration" hash is included to prevent race conditions
  - Allowed updates: TA binary and personalization data (encrypted to the SP anonymous private key)

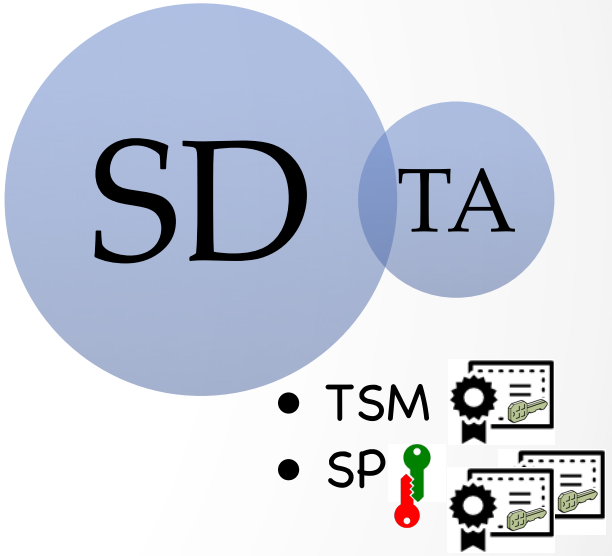


- `UpdateTAResponse`
  - Signed by TEE and encrypted to requesting TSM private key

TA Management: DeleteTA

TSM is managing the full lifecycle of a Trusted Application in the Security Domain created on behalf of the Service Provider

- `DeleteTARequest`
  - Signed by TSM and encrypted to target TEE private key
  - Includes TSM and SP identity information
  - "Last known configuration" hash is included to prevent race conditions



- `DeleteTAResponse`
  - Signed by TEE and encrypted to requesting TSM private key