ATIS/SIP Forum IPNNI

Policy Administrator and Certificate Management

SIPNOC 2018

December, 2018

Mary Barnes, iconectiv Industry Solutions

Email: mbarnes@iconectiv.com

STI-Policy Administrator (STI-PA)

Trust Authority Role

- Supports unique requirements of managing PKI infrastructure for STI and Service Providers' interactions with the PKI
  - Serves as the Trust Authority for the PKI
    - Maintains a list of valid STI-CAs
  - Serves as a Trust Anchor providing valid service providers with a unique token for authorization to get STI certificates

Certificate Issuance

- Serves no direct role in the issuance or validation of certificates:
  - Service Provider Code token mechanism used for authorization
  - Traditional PKI mechanisms for certificate validation are followed during the verification process
    - STI-PA is NOT in the Certification Path

iconectiv

STI-PA Role: Administering CAs

STI-PA Policy Management Authority approves STI-CAs

- Reviews the Certification Practice Statement (CPS) of the STI-CA to approve an STI-CA:
  - Ensures the STI-CA is operated to an acceptable level of assurance and supports the policies established the Certificate Policy (CP)
- Applies policies and other criteria as established by the STI-GA, for example:
  - STI-CA has appropriate expertise
  - STI-CA and Certificate Repository (CR) are located only in specific geographic regions (or specific regions are excluded)
  - Periodic audits recommended

STI-PA periodically distributes/updates list of valid STI-CAs

- Mechanism as specified by ATIS-1000084

STI-PA Role: Administration of Service Providers

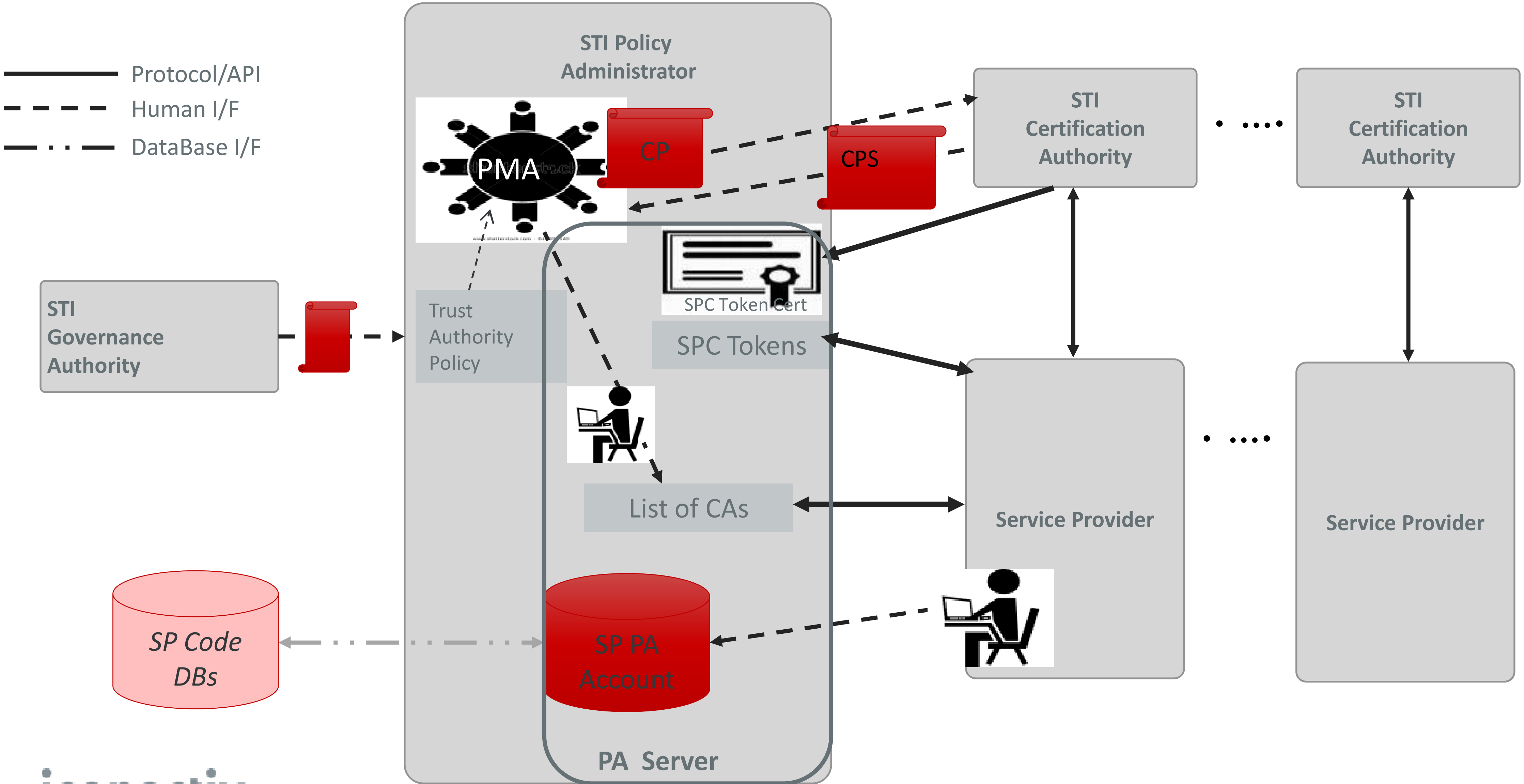
Service Provider Codes

- Existing identifiers (e.g., OCNs) are used as Service Provider Codes:
  - Service Provider codes are allocated and managed by an entity authorized by a National/Regional Regulatory Authority
  - Provide uniqueness & accountability

Service Provider Authorization

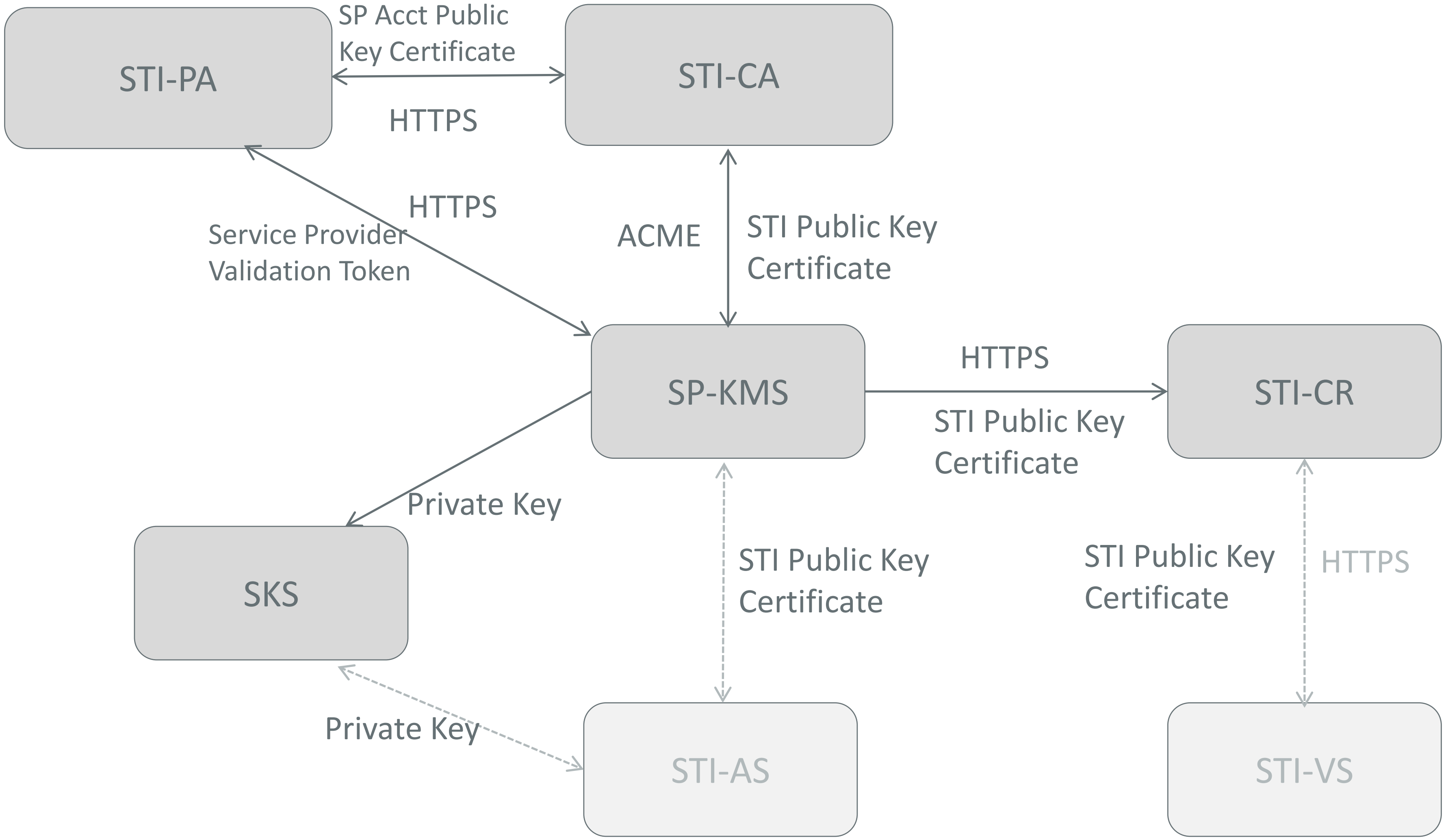
- Prior to requesting a certificate, a Service Provider must:
  - Create an account with the STI-PA
  - Create an account with an STI-CA
  - Obtain a service provider code token from the STI-PA (as Trust Anchor) per the procedures outlined in ATIS-1000080.

SHAKEN Policy Administrator Roles and Responsibilities



iconectiv

Certificate Management Architecture


STI-PA
SP Acct Public Key Certificate
HTTPS
STI-CA
HTTPS
Service Provider Validation Token
ACME
STI Public Key Certificate
SP-KMS
HTTPS
STI Public Key Certificate
STI-CR
Private Key
SKS
STI Public Key Certificate
STI Public Key Certificate
HTTPS
Private Key
STI-AS
STI-VS

Interface used during Session Setup

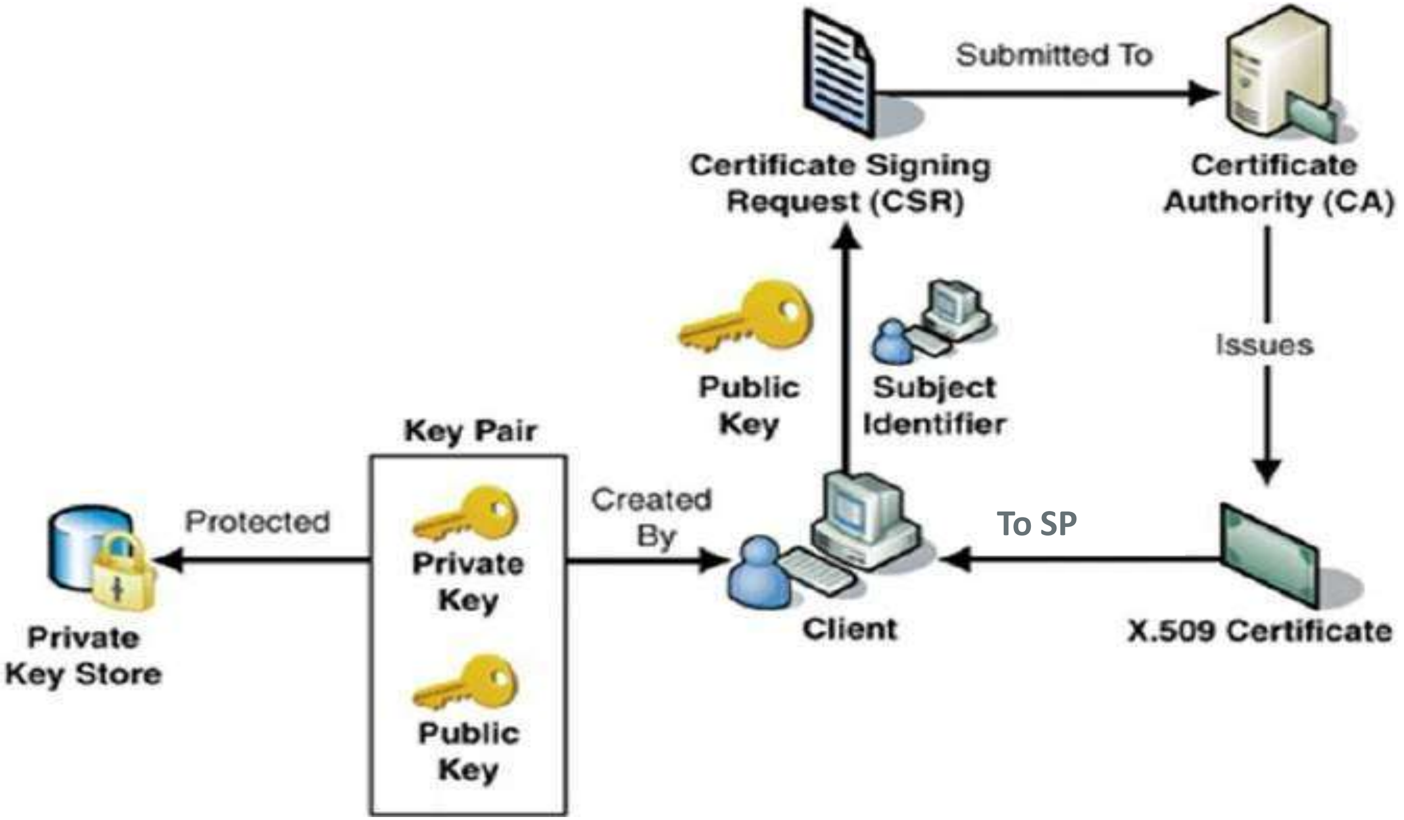
iconectiv

Certificate Management Functional Elements

STI-CA	Secure Telephone Identity Certification Authority
STI-CR	Secure Telephone Identity Certificate Repository
SP-KMS	Service Provider Key Management Server
SKS	(Service Provider) Secure Key Store

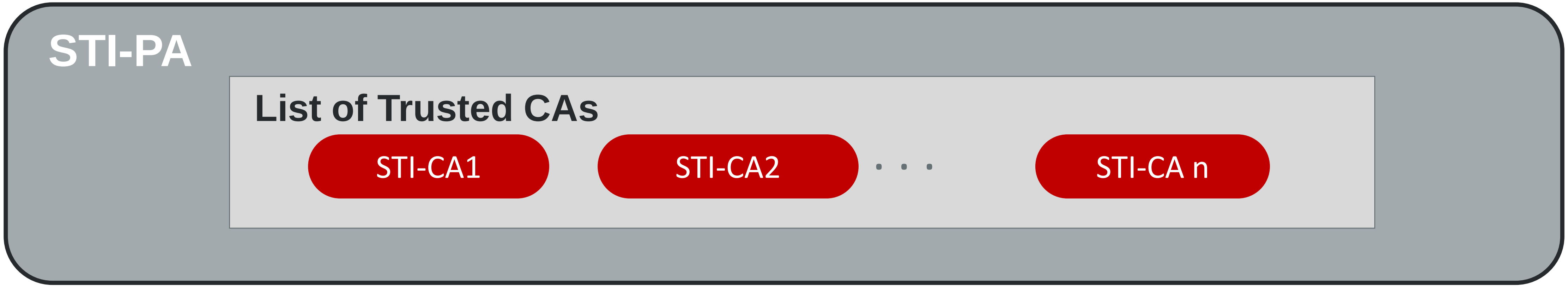
Public Key Certificates

- SHAKEN uses existing widely deployed Public Key Infrastructure principals and techniques (X.509 Certificates) and secure tokens to securely carry telephone identities through the network:
  - Private keys are only visible to the Originating Service Provide
  - Public Keys are available along the call path
  - Public Key certificate allows terminating service provider to verify the authenticity of the telephone identity of calling party



iconectiv

SHAKEN Trust Authority Model



- STI-PA is external to the PKI – maintains list of Trusted CAs on behalf of the relying parties in the PKI
- STI-PA serves as the Trust Anchor to the relying parties in the PKI
- Each STI-CA must support Certificate Policy (CP) as established by the STI-PA
- STI-PA reviews Certification Practice Statement (CPS) as provided by the STI-CAs to ensure compliance
- STI-PA also supports the distribution of Certificate Revocation Lists (CRLs)*

* Details currently being specified in IPNNI Task group

Benefits of SHAKEN PKI Model

A single STI-PA is deployed per country/region

- SHAKEN model defines STI-PA as a Trust Authority and NOT as a Root CA
- STI-CAs serve as Root CAs

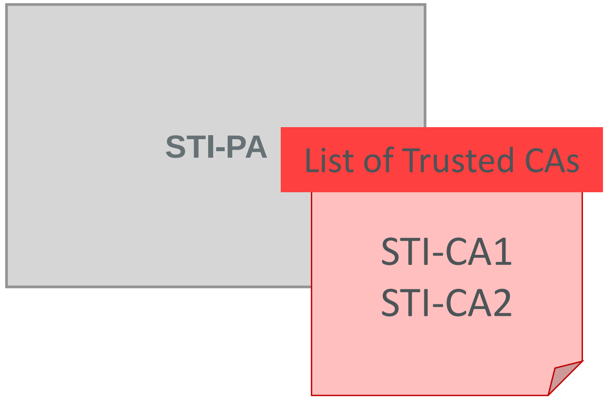
Service Provider controls which STI-CA to use

- Service Providers can use CAs that meet their business and operational requirements
- STI-PA controls who can serve as an STI-CA and who can obtain certificates – BUT does not control the PKI

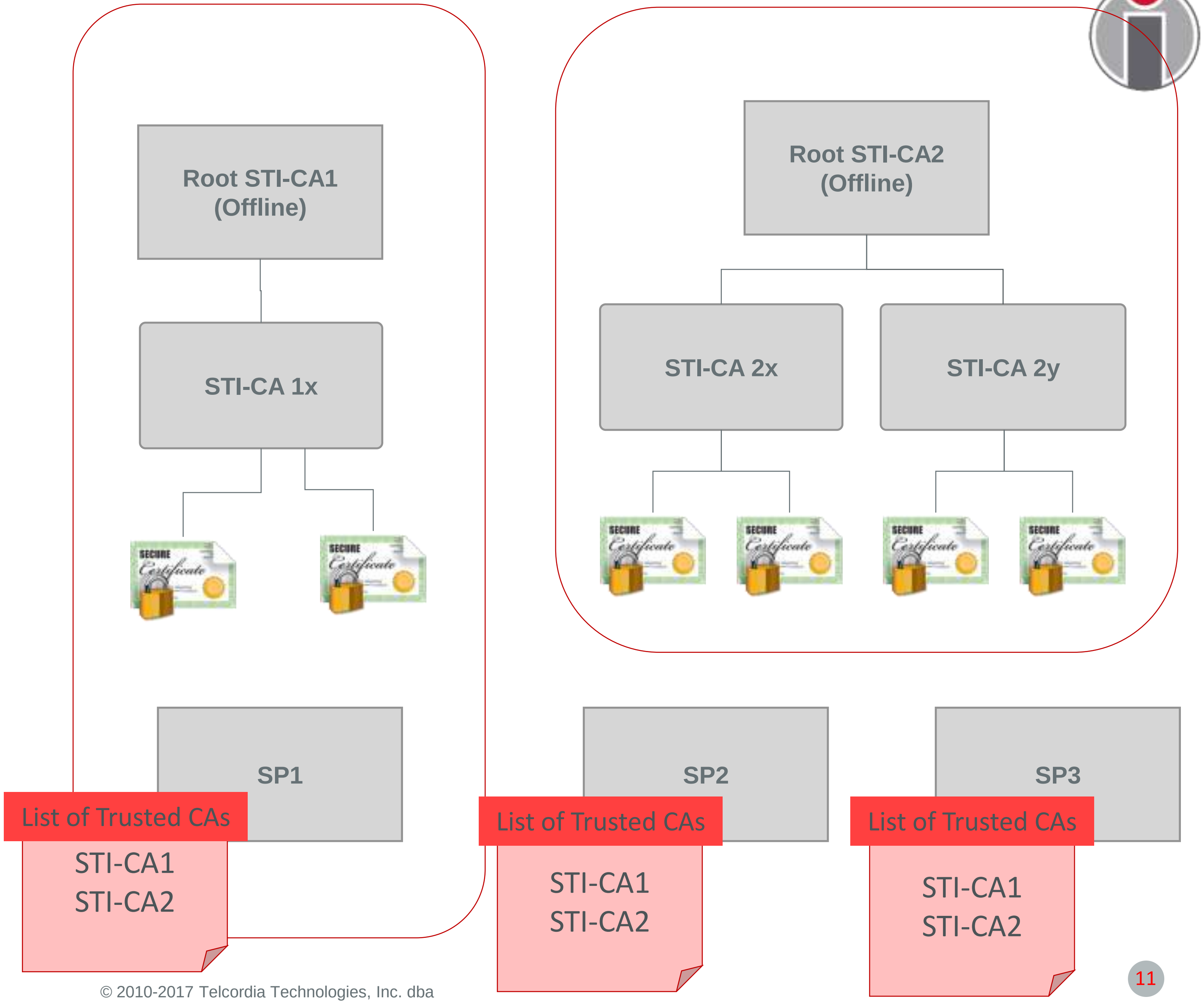
Within each STI-CA, a hierarchical, single Root CA model can be established

- Each STI-CA can have an offline root CA with one or more intermediate CAs
- Aligns with model and practices used by existing CAs

SHAKEN PKI Model



- STI-PA maintains & distributes list of Trusted CAs
- Local policy determines which issuing CA an SP uses
- Certificate is trusted due to trust in Root CA on the list of Trusted CAs



STI-Certification Authority (STI-CA)

Roles and responsibilities

Align with those of traditional PKI (RFC 5280)

New x.509 Extension

(TNAuthList, OID 26) added to CSR/certificate to support unique STI identifier requirements (RFC 8226)

Protocol

Interface between STI-CA SP-KMS uses an automated certificate management protocol (ACME) (draft-ietf-acme-acme)

Uses new Authority Token based “challenge” to support authorization of service providers to obtain certificates (draft-ietf-acme-authority-token-tnauthlist)

STI-Certificate Repository (STI-CR)

STI-VS gets STI public key certificate used to sign the Identity header field from the STI-CR during the verification process

- No new functionality or interfaces required
- Follows existing procedures as defined in RFC 5280

SP-Key Management Server (SP-KMS)

PKI Interface

SP-KMS serves as the Service Provider’s interface to the PKI

ACME client

SP-KMS hosts the ACME client which maintains an account with the ACME server hosted by the STI-CA

Private Key

SP-KMS distributes private key to a Secure Key Store for access by the STI-AS when signing the PASSporT in the Identity header field.

Public Key Certificate

SP-KMS distributes the STI public key certificate to STI-CR

STI-AS Interface

SP-KMS ensures the STI-AS has access to the STI public key certificate URL for inclusion in the PASSporT in the SIP Identity header field

ACME Overview

- ACME is a protocol developed in IETF for Automated Certificate Management.
- ACME defines an extensible framework for automating the issuance and validation procedures for certificates:
  - Allows servers to obtain certificates without manual user interaction
- ACME protocol specifications:
  - Core protocol: draft-ietf-acme-acme
  - Authority Token Based Challenge/Response: draft-ietf-acme-authority-token
  - TNAuthlist (TNs and Service Provider codes) Authority Token Profile: draft-ietf-acme-authority-token-tnauthlist

ACME Protocol Model

- ACME uses HTTPS as a transport for Javascript Object Notation (JSON) Web Tokens (JWTs) in the form of JSON Web Signature (JWS) objects - effectively a RESTful API:
  - ACME server runs at a Certification Authority (CA) and responds to client's actions if the client is authorized.
  - ACME client uses the protocol to request certificate management actions.
  - ACME client is represented by an "account key pair".
    - ACME client uses the private key to sign all messages to the server.
    - ACME server uses public to verify the authenticity and integrity of messages from the client.

ACME Protocol Resources

- ACME defines the following resource objects for representing information:
  - Account object: metadata associated with account
  - Order object: represents a client’s request for a certificate – contains information about the requested certificate, the server’s requirements and any (URL for) certificates (certificate resource) that have been issued.
  - Authorization object: contains the “challenges” (challenge resource) for identifier validation
  - Challenge object: represents the challenge to prove possession of the identifier
  - Certificate object: : represents the issued certificates

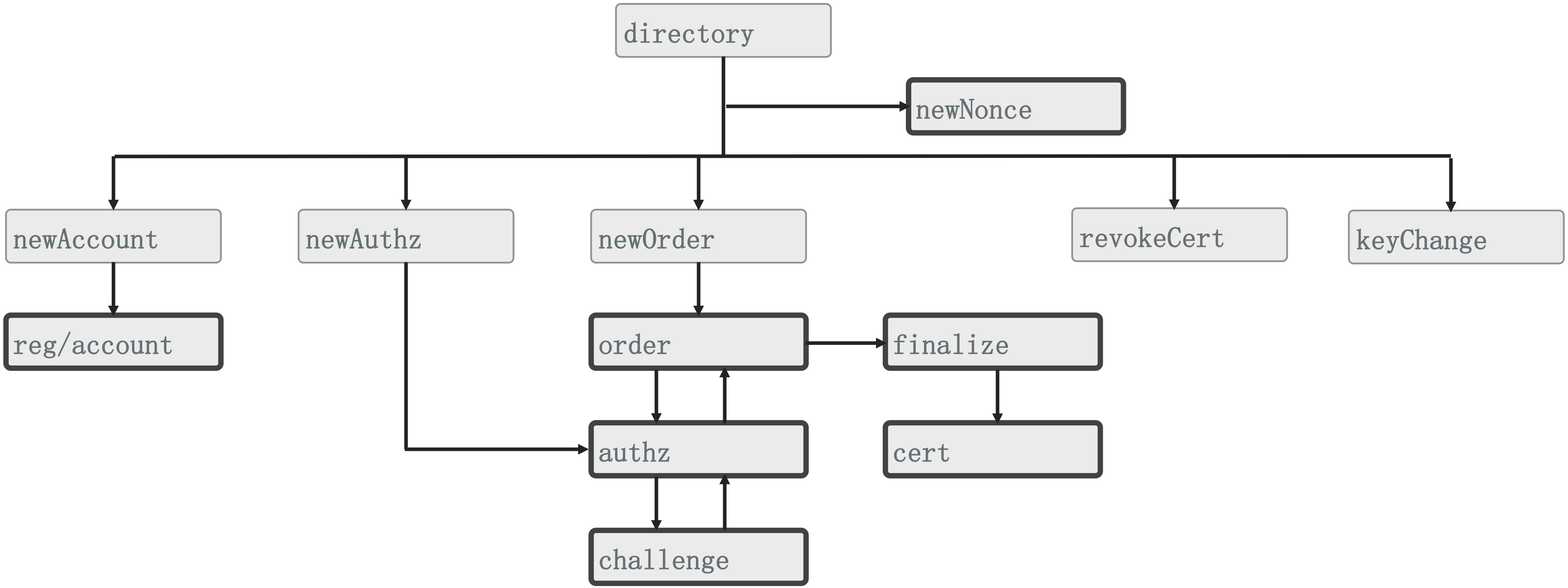
ACME Protocol Functions

- ACME uses different URLs (resources) for different management functions:
    - New nonce
    - New Account
    - New Order
    - New Authorization
    - Revoke Certificate
    - Key change
- A single Directory URL is configured in client in order to get the Directory object containing the above URLs.

ACME Protocol Resource States

- Each resource object has a status field that reflects the state of the object and is used by the client and server to effect changes such as:
  - ACME server sets the status to “valid” in the Authorization object to indicate that the requestor of the certificate has been validated.
    - In the case of challenge/response, ACME client periodically polls (POST-as-GET) the Authorization object to determine if status is “valid”
  - ACME client sets the status to “deactivated” in the Account object to deactivate an account

ACME Directory objective



ACME - high level flow

- ACME high level request flow:

  Get a nonce - HEAD newNonce

  Create Account - POST newAccount

  Submit order for a cert (order) - POST newOrder

  Fetch challenges – POST-as-GET authz

  Answer challenges – POST-as-GET challenge response

  Poll for status – POST-as-GET order

  Finalize order – POST order's finalize url

  Poll for status – POST-as-GET order

  Download certificate – POST-as-GET certificate URL

ACME - Order object

- ACME Order object represents a client’s request for a certificate, and it’s lifecycle through to issuance.

status (required, string) - status of the application. “pending”, “ready”, ”processing”, “valid”,“invalid”

expires (optional, string) - timestamp of when the server will no longer consider the application valid

identifiers (required, array of object) – an array of identifier objects that the order pertains to: type (required, string), value (required, string)

notBefore (optional, string) - requested notBefore field in the certificate

notAfter (optional, string) - requested notAfter field in the certificate

error (optional, object) – error that occurred while processing order

authorizations (required, array or string) - requirements client needs to fulfill before granting certificate

finalize (required, string) - URL that a CSR must be POSTed to once all the authorizations have satisfied

certificate (optional, string) - URL for the issued certificate

ACME - Order example

```
 {
 POST /acme/new-order HTTP/1.1
Host: sti-ca.com
Content-Type: application/jose+json
{
  "protected": base64url({
    "alg": "ES256",
    "kid": "https://sti-ca.com/acme/acct/1",
    "nonce": "5XJ1L3lEkMG7tR6pA00clA",
    "url": "https://sti-ca.com/acme/new-order"
  })
  "payload": base64url({
     ”status": ”pending”,
     "identifiers": [{"type:"TNAuthList","value":"F83n2a...avn27DN3=="}],
    "notBefore": "2016-01-01T00:00:00Z",
    "notAfter": "2016-01-08T00:00:00Z"
  }),
  "signature": "H6ZXtGjTZyUnPeKn...wEA4TklBdh3e454g"
}
  }
```

ACME - Authorization object

- ACME authorization object represents a server's authorization for an account to represent an identifier.

status (required, string) - status of the authorization. "pending", "processing", "valid", "invalid", "revoked" - default value is "pending".

expires (optional, string) - if present, contains URI for an application resource, if absent, then CA MUST consider authorization valid for all applications

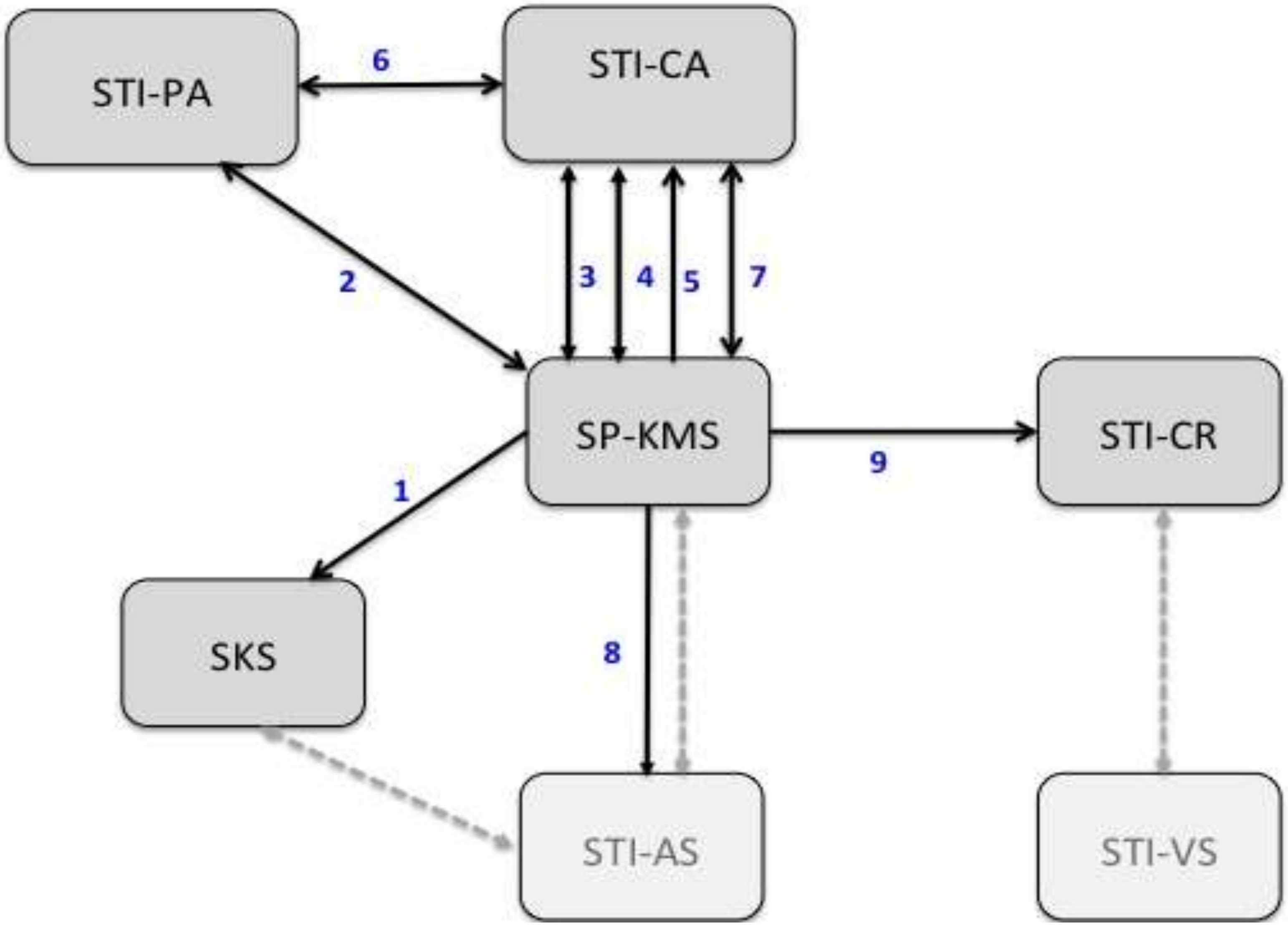
identifier (required, dictionary of string) - The identifier that the account is authorized to represent
- **type** (required, string) - type of identifier
- **value** (required, string) - identifier itself

challenges (required, array) - an array of challenges used for authorization

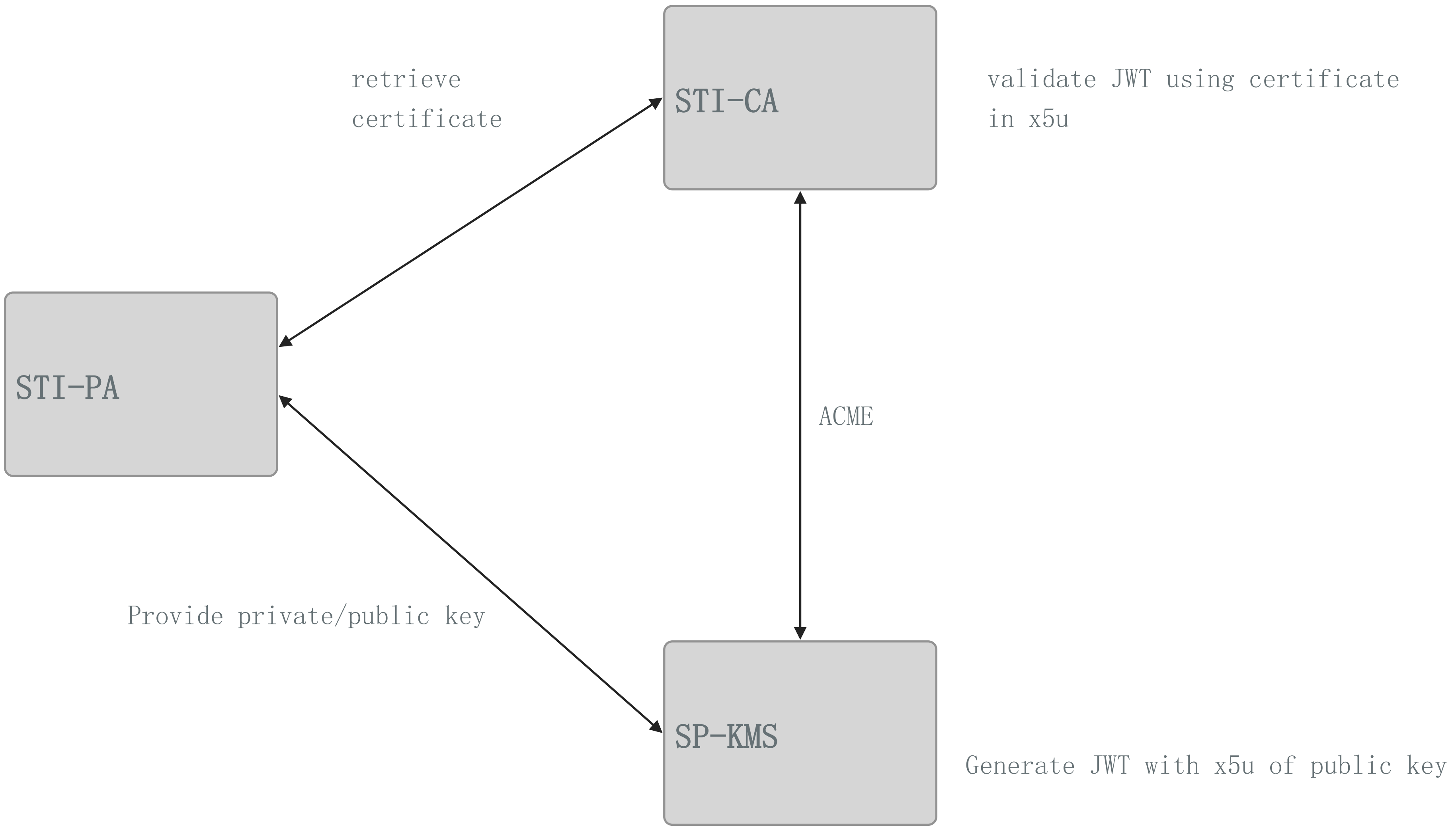
ACME – Authorization Example

```
HTTP/1.1 200 OK
Content-Type: application/json
Link: <https://sti-ca.com/acme/some-directory>;rel="index"

{
  "status": "pending",

  "identifier": {
    "type": "TNAuthList",
    "value":"F83n2a...avn27DN3=="
  },

  "challenges": [
    {
      "type": "tkauth-01",
      "tkauth-type": "ATC",
      "url": "https://sti-ca.com/authz/1234/0",
      "token": "DGyRejmCefe7v4NfDGDKfA"
    }
  ],
}
```

SHAKEN Certificate Management Call Flow



SHAKEN Service Provider Code Token Authorization



STI-CA

STI-PA

SP-KMS

retrieve certificate

validate JWT using certificate in x5u

ACME

Provide private/public key

Generate JWT with x5u of public key

Structures that manage scope of STI Certificates

TN Authorization List certificate extension

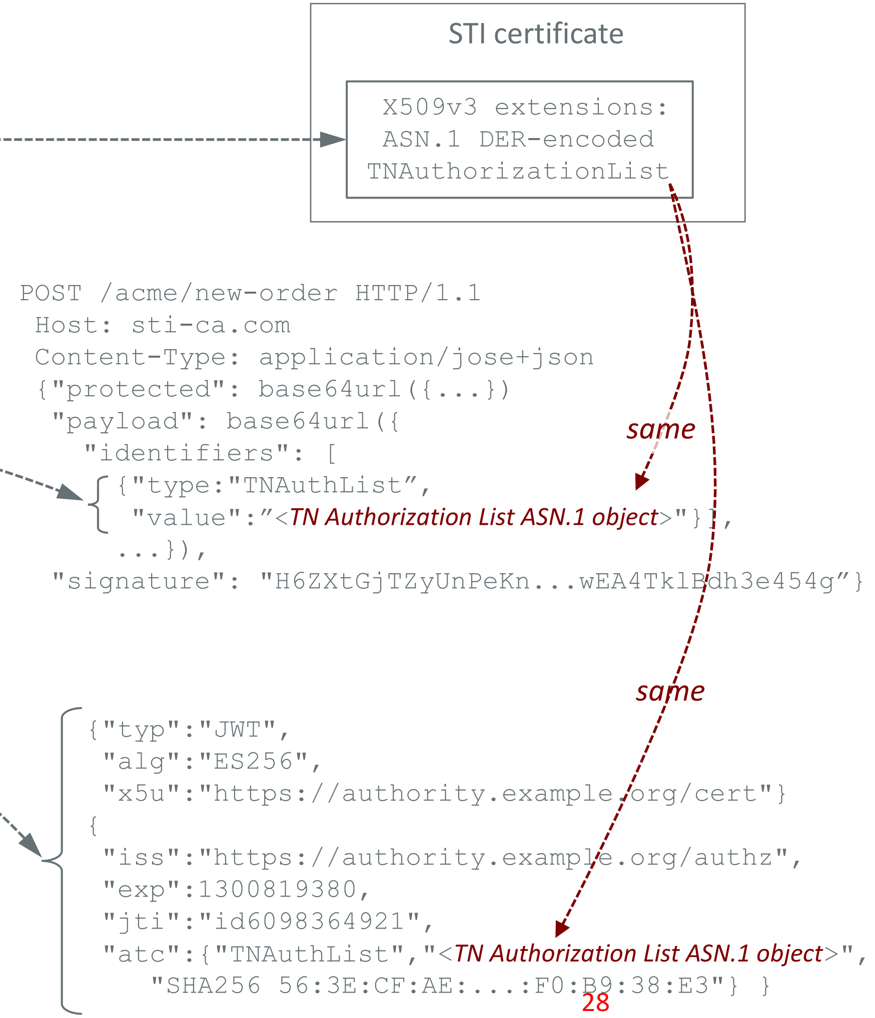
- Defined in RFC 8226
- Specifies the scope of authority of an STI certificate
- Contains identity information (SPC and TNs of cert holder)
- SHAKEN restricts scope to a single SPC (no TNs)

TNAuthList Identifier

- Defined in draft-ietf-acme-authority-token-tnauthlist
- Used during ACME certificate ordering procedure to specify scope of requested certificate (SPC and TNs)
- SHAKEN restricts scope to a single SPC (no TNs)

SPC Token

- Defined in ATIS-1000080, based on definitions in draft-ietf-acme-authority-token-tnauthlist
- Enables token holder to demonstrate authority over a TNAuthList Identifier during ACME certificate ordering procedure
- Has the same structure as a TNAuthList Authority Token defined in draft-ietf-acme-authority-token-tnauthlist, but SHAKEN restricts scope to a single SPC (no TNs)



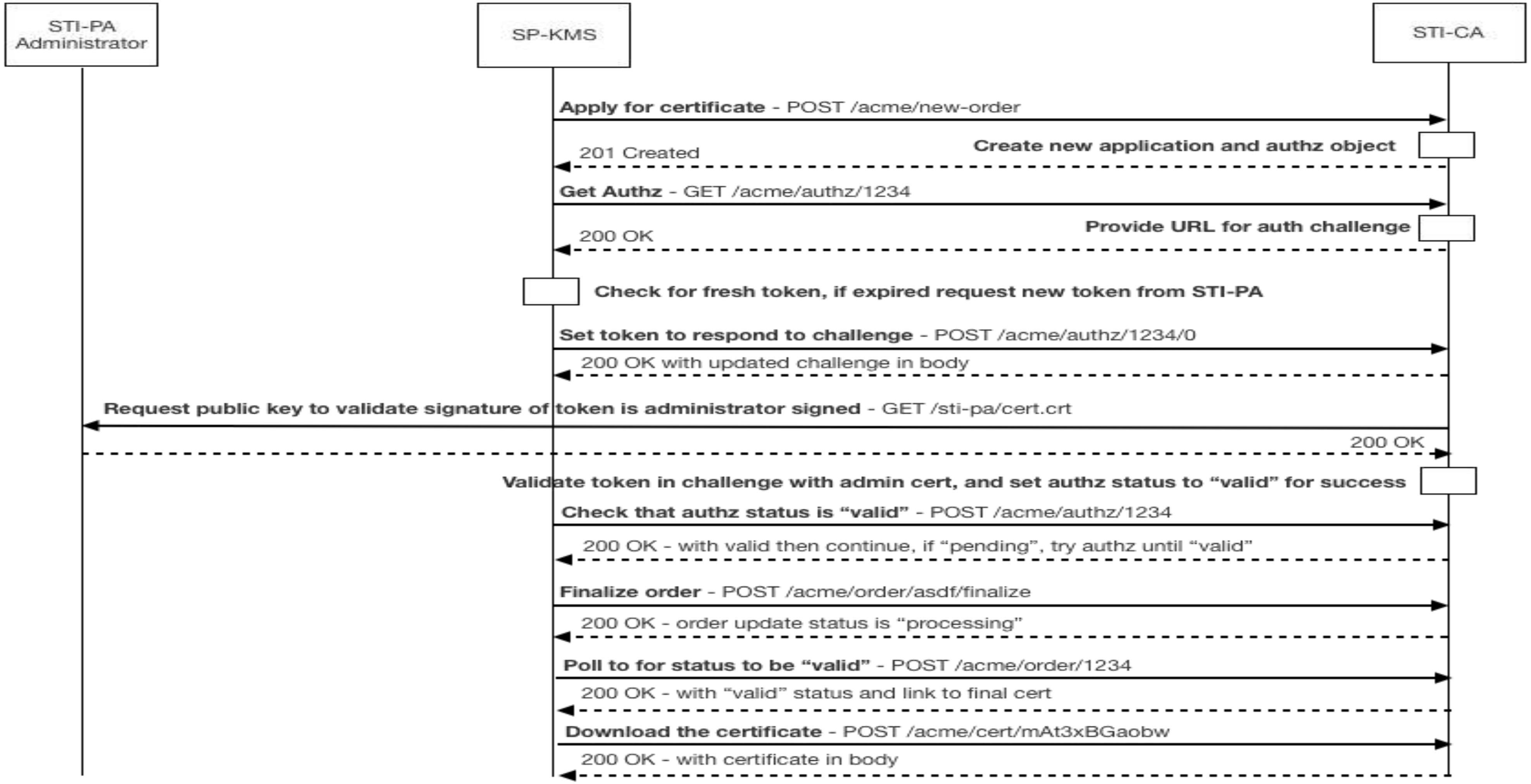
STI certificate

```
X509v3 extensions:
ASN.1 DER-encoded
TNAuthorizationList
```

```
POST /acme/new-order HTTP/1.1
 Host: sti-ca.com
 Content-Type: application/jose+json
 {"protected": base64url({...})
  "payload": base64url({
    "identifiers": [
     {"type:"TNAuthList”,
      "value":”<TN Authorization List ASN.1 object>"}],
     ...}),
  "signature": "H6ZXtGjTZyUnPeKn...wEA4TklBdh3e454g”}
```

```
{"typ":"JWT",
 "alg":"ES256",
 "x5u":"https://authority.example.org/cert"}
{
 "iss":"https://authority.example.org/authz",
 "exp":1300819380,
 "jti":"id6098364921",
 "atc":{"TNAuthList","<TN Authorization List ASN.1 object>",
    "SHA256 56:3E:CF:AE:...:F0:B9:38:E3"} }
```

SHAKEN Certificate Management Call Flow



STI-PA Administrator

SP-KMS

STI-CA

Apply for certificate - POST /acme/new-order

Create new application and authz object

201 Created

Get Authz - GET /acme/authz/1234

Provide URL for auth challenge

200 OK

Check for fresh token, if expired request new token from STI-PA

Set token to respond to challenge - POST /acme/authz/1234/0

200 OK with updated challenge in body

Request public key to validate signature of token is administrator signed - GET /sti-pa/cert.crt

200 OK

Validate token in challenge with admin cert, and set authz status to "valid" for success

Check that authz status is "valid" - POST /acme/authz/1234

200 OK - with valid then continue, if "pending", try authz until "valid"

Finalize order - POST /acme/order/asdf/finalize

200 OK - order update status is "processing"

Poll to for status to be "valid" - POST /acme/order/1234

200 OK - with "valid" status and link to final cert

Download the certificate - POST /acme/cert/mAt3xBGaobw

200 OK - with certificate in body

ACME Protocol Status

- ACME base protocol document finally approved in October 2018. RFC should be published early 2019
- Authority token mechanism replaces proposed mechanism in ATIS-1000080 (revisions being worked in IPNNI task group)
- At least 4 CA implementations supporting the ACME Protocol: Let’s Encrypt, Entrust, BuyPass (ETSI certified CA) and GlobalSign
- 35 ACME clients compatible with Let’s Encrypt (based on ACME v2 -09 version of specification)

Backup

PKI Model for SHAKEN

Model similar to inter-domain PKI*

- Each STI-CA serves as a root CA operating independently – no cross certificates
- STI-CAs must be approved by the STI-PA and follow Certificate Policy requirements defined by the STI-PA
- STI-PA maintains a Trust List of approved CAs
- Allows each Service Provider to use an STI-CA that satisfies their business needs and meets established corporate security requirements

* see examples in RFC 5217

Use of the X.509 certificates follows standard practices:

- Certification path validation per procedures defined in RFC 5280.
  - STI-PA is not in the Certification Path
- Additional step to verify that root CA is on the Trust List

Comparison to WebPKI

Similarities

- List of CAs similar to Trust Anchor Store
- User selects the CA from which to obtain certificates
- No single Root CA
- Hierarchy not imposed

Differences

- Establishes a clear governance structure with STI-GA and STI-PA
- STI-PA defines clear guidelines by which an STI-CA is added to the Trust List
- Defined procedures for providing Service Providers an updated Trust List
- STI-PA is the single entity authorized to remove an STI-CA from the Trust List
- Defined procedures for who is allowed to obtain certificates from an STI-CA

Comparison to Enterprise PKI

Similarities

- Single entity (STI-PA) controls who can issue certificates
- CAs issue X.509 v3 certificates
- RFC 5280 certificate validation procedures followed

Differences

- STI-PA as the Trust Authority (Anchor) is not the root CA
- No single Root CA
- List of Trusted CAs provides PKI trust anchor for certificates
- Service Provider selects the CA from which to obtain certificate