SIPNOC 2019
SHAKEN STI Policy Administrator and Certificate Management

December 3, 2019
Mary Barnes, iconectiv Industry Solutions
Director, SHAKEN Policy Management Authority
Email: mbarnes@iconectiv.com
mary.sip.barnes@gmail.com
SHAKEN Certificate Management Architecture

- **Policy Administrator (STI-PA)**
  - Service Provider Validation Token + CRL URL
  - HTTPS

- **Certification Authority (STI-CA)**
  - HTTPS

- **Key Management Server (SP-KMS)**
  - HTTPS
  - ACME
  - STI Public Key Certificate

- **Secure Key Store (SKS)**
  - Private Key
  - HTTPS

- **Certificate Repository (STI-CR)**
  - HTTPS
  - STI Public Key Certificate
  - ATIS-1000080/ATIS-1000084

- **List of Approved STI-CAs**
  - Interface used during Session Setup

- **STI-AS**
  - Private Key
  - HTTPS

- **STI-VS**
  - HTTPS

- **SP Acct Public Key Certificate**
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 Provider
  - 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
    - Includes a new OID including Service Provider Code (OCN)
    - Leverages existing Issuing Distribution Point OID (DistributionPointName field) for indirect CRL model
Certificate Management Roles

- Trust Anchor
  - STI-PA

- (offline) Root CA
  - Certification Authority
  - Issuing CA (ACME Server)

- Private Key
  - STI-AS

- Service Provider Validation Token
  - + CRL URL
  - Private Key
  - SKS

- STI Public Key Certificate
  - SP-KMS (ACME Client)

- STI Public Key Certificate
  - STI-CR

- Certificate Repository
  - HTTPS
  - STI-VS

- List of Trusted STI-CAs
STI-PA Role: Administration of STI-CAs

STI-PA Policy Management Authority approves STI-CAs

- Develops the Certificate Policy (CP)
- Reviews the Certification Practice Statement (CPS) of the STI-CA to approve an STI-CA
- Applies policies and other criteria as established by the STI-GA including:
  - STI-CA has appropriate expertise
  - STI-CA and Certificate Repository (CR) within the US
- Periodic audits recommended

STI-PA periodically 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 Trust Authority Model

• STI-PA 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)

List of Trusted CAs

STI-PA

STI-CA1 Root  STI-CA2 Root  ...  STI-CA n Root
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 by maintaining a list of trusted 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 certification path validation are followed during the verification process:
    • STI-PA is NOT in the Certification Path
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) (RFC 8555) (optional)

Uses new Authority Token based “challenge” to support authorization of service providers to obtain certificates (draft-ietf-authority-token & draft-ietf-acme-authority-token-tnauthlist) (protocol optional BUT validation of SPC token is mandatory)
STI-Certificate Repository (STI-CR)

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
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 PMA
- 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:

- New extension (OID) defined for SHAKEN TNAuthList
- Indirect model for CRLs:
  - STI-PA manages the CRL
  - URL to CRL distributed in the issued certificates in the DistributionPointName field in the certificate
- Certification path validation per procedures defined in RFC 5280.
  - Additional step to verify that root CA is on the Trust List
Comparison to Enterprise PKI

**Similarities**

- Single entity (STI-PA) controls who can issue certificates
- CAs issue X.509 v3 certificates
- RFC 5280 certification path 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
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, centralized 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
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 certificate issuance

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 list of Trusted CAs
- Indirect CRL model supported:
  - STI-PA maintains CRL
  - URL is distributed in the certificates
- Local policy determines which STI-CA an SP uses for certificate acquisition
- Certificate is trusted due to trust in Root CA on the list of Trusted CAs

Root STI-CA1 (Offline)

STI-CA 1x

SP1

List of Trusted CAs
STI-CA1 Root Certificate
STI-CA2 Root Certificate

Root STI-CA2 (Offline)

STI-CA 2x

SP2

List of Trusted CAs
STI-CA1 Root Certificate
STI-CA2 Root Certificate

STI-CA 2y

SP3

List of Trusted CAs
STI-CA1 Root Certificate
STI-CA2 Root Certificate
Policy Administrator Implementation Status

• iconectiv announced as Policy Administrator for the SHAKEN ecosystem in May 2019: [https://authenticate.iconectiv.com/](https://authenticate.iconectiv.com/)

• PMA has produced a CP: [https://authenticate.iconectiv.com/documents-authenticate](https://authenticate.iconectiv.com/documents-authenticate)
  • CAs are submitting CPSs, currently under review by PMA.

• PA system testing is wrapping up with Service Providers and CAs
PA functions: Administering CAs

• STI-GA provides PMA with policies associated with CA selection (Box 1)
• PMA develops Certificate Policy (Box 2)
• STI-CAs create an account with the STI-PA (portal)
• STI-CA provides CPS to PMA for review/approval (Box 2)
• The PMA director notifies PA admin when a CA has been approved (Box 3)
• The STI-CA’s root certificate is added to the list of trusted STI-CAs (Box 3) (portal)
• SPs retrieves the STI-CA trust list from the STI-PA (Box 3) (HTTPS interface)
• SPs and CAs add revoked certificates to the CRL (Box 6) (Portal)
PA functions: Administration of SPs

- STI-GA has set the policy that OCNs are used as Service Provider Codes, SPs must have numbering resources and must have a Form 499A on file with the FCC (Box 1)*

- Prior to requesting a certificate, a Service Provider must:
  - Create an account with the STI-PA (Box 4) (portal)
  - Create an account with an STI-CA – the fingerprint of the account credentials is input for the token generation**
  - Obtain a service provider code token from the STI-PA (Box 5) (API over HTTPS)
  - STI-PA provides the URL to the CRL in the SPC token request response (Box 5) (API over HTTPS)


** Not a PA function but a necessary step
STI-GA Role

Regulatory

STI-GA is the conduit for regulatory impacts to the functions associated with the overall deployment of SHAKEN within the telephone network.

Policies & Procedures

STI-GA establishes policies and procedures associated with the role and functions of the STI-PA and the overall ecosystem:

- Considers input from ATIS/SIP Forum IPNNI Task Group and industry
- Defines specific criteria for approving STI-CAs (beyond standard CP) such as geographic location of STI-CA and CR, operational experience, etc.
- Defines source of information for uniquely identifying valid Service Providers (i.e., source of Service Provider Code)

STI-PA Selection

STI-GA appoints the STI-PA. The STI-PA operates a Policy Management Authority (PMA) that establishes the Certificate Policies for STI-CAs, incorporating policies as established by the STI-GA.
Certificate Revocation

• Mechanism to support Certificate Revocation added to ATIS-1000080 Errata since not all SPs will support short-lived certificates

• Indirect CRL model is used:
  • STI-PA maintains the Certificate Revocation List (CRL) – entries are in the form specified in RFC 5280
  • A URL to the CRL is included in the response to the request for an SPC token
  • SP includes URL to the CRL in the CSR when requesting certificate issuance in the DistributionPointName field
  • SPs and STI-CAs provide information on any certificates that have been revoked to the STI-PA using an out of band mechanism
Management and Distribution of Certificate Revocation List

CertificateList ::= SEQUENCE {
  tbsCertList  TBSCertList,
  signatureAlgorithm  AlgorithmIdentifier,
  signatureValue  BIT STRING }

TBSCertList ::= SEQUENCE {
  signature  AlgorithmIdentifier,
  issuer    Name,
  thisUpdate Time,
  nextUpdate Time,
  revokedCertificates  SEQUENCE of SEQUENCE {
    userCertificate  CertificateSerialNumber,
    revocationDate  Time
  }
}

B.1. Add Revoked Certificate to CRL

A.1. Add Revoked Certificate to CRL

1.1 SPC Token Request

1.2 SPC Token + CRL URL

2.1 GET Cert (CSR w/ CRL URL)

2.2 Cert w/ CRL URL

ACME Server

STI-VS

1.1 SPC Token Request

1.2 SPC Token + CRL URL

2.1 GET Cert (CSR w/ CRL URL)

2.2 Cert w/ CRL URL

ACME Server

STI-VS

1.1 SPC Token Request

1.2 SPC Token + CRL URL

2.1 GET Cert (CSR w/ CRL URL)

2.2 Cert w/ CRL URL

ACME Server

STI-VS

1.1 SPC Token Request

1.2 SPC Token + CRL URL

2.1 GET Cert (CSR w/ CRL URL)

2.2 Cert w/ CRL URL

ACME Server

STI-VS

1.1 SPC Token Request

1.2 SPC Token + CRL URL

2.1 GET Cert (CSR w/ CRL URL)

2.2 Cert w/ CRL URL

ACME Server

STI-VS

1.1 SPC Token Request

1.2 SPC Token + CRL URL

2.1 GET Cert (CSR w/ CRL URL)

2.2 Cert w/ CRL URL

ACME Server

STI-VS

1.1 SPC Token Request

1.2 SPC Token + CRL URL

2.1 GET Cert (CSR w/ CRL URL)

2.2 Cert w/ CRL URL

ACME Server

STI-VS

1.1 SPC Token Request

1.2 SPC Token + CRL URL

2.1 GET Cert (CSR w/ CRL URL)

2.2 Cert w/ CRL URL

ACME Server

STI-VS
ACME Overview

• ACME is a protocol developed in IETF for Automated Certificate Management (Environment).

• ACME defines an extensible framework for automating the issuance and validation procedures for certificates:
  • Allows servers to obtain certificates without manual user interaction
  • Based on a simple JSON over-HTTPS interface

• ACME protocol specifications:
  • Core protocol: RFC 8555
  • Authority Token Based Challenge/Response: draft-ietf-acme-authority-token
  • TNAuthlist (TNs and Service Provider codes) Authority Token Profile: draft-ietf-acme-authority-token-tauthlist
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 key to verify the authenticity and integrity of messages from the client.
ACME Protocol Resource Objects

- **Account Object**: Metadata Associated with Account
- **Order Object**: Information for the Certificate including Certificate Signing Request (CSR)
- **Authorization Object**: Challenges for Identifier validation
- **Challenge Object**: Challenge response to prove Possession of Identifier
- **Certificate Object**
ACME Protocol SHAKEN Resource Objects

- **Account Object**
- **Order Object**
- **Authorization Object**
- **Challenge Object**
- **Certificate Object**

**Metadata Associated with Account**

- SHAKEN CSR includes TNAuthorizationList & DistributionPointName

**SHAKEN challenge:**

- Authority Token Identifier of type “TNAuthList”

**SHAKEN challenge Response includes:**

- Authority Token (“atc”) which includes SPC Token
- In TNAuthlist

**SHAKEN certificate includes:**

- “TNAuthorizationList” & “DistributionPointName”

---

- In the context of SHAKEN (ATIS-1000080), the TNAuthList contains only one Service Provider Code (SPC).
- DistributionPointName contains the URL to the Certificate Revocation List (CRL) received in response to SPC Token Request.
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 - 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

```json
{
    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": "5XJ1L3IEkMG7tR6pA00cIA",
            "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...wEA4TkIBdh3e454g"
    }
}
```
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": "DGyRejmCefe7v4NfDGDkFkA"
    }
  ]
}
SHAKEN Certificate Management Call Flow

1. **STI-PA Administrator**
   - Apply for certificate - POST /acme/new-order
   - 201 Created
   - Get Authz - GET /acme/authz/1234
   - 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 /stipa/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"

2. **SP-KMS**
   - Create new application and authz object

3. **STI-CA**
   - Provide URL for auth challenge

---

**Notes:**

- POST /acme/new-order
- POST /acme/authz
- POST /acme/authz/1234/0
- GET /stipa/cert.crt
**SHAKEN Certificate Acquisition Sequence of Events**

1. **KMS Creates ACME Account**
   - PKMS creates ACME account
   - PKMS sends order to STI-CA
   - PKMS sends challenge response to STI-CA
   - PKMS finalizes order with STI-CA
   - PKMS retrieves certificate in STI-CR & notifies STI-AS

2. **Create Key Pair for ACME Account**
   - 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": "H6ZXtGjT5yUnPeKn...wEA4Tk1Bdh3e454g"}
   )

3. **KMS Sends order to STI-CA**
   - POST /acme/order/order HTTP/1.1
   - Host: sti-ca.com
   - Content-Type: application/jose+json
   - "protected": base64url({...})
   - "payload": base64url({
     "csr": "MIIBFTCbxAIABDBFMQ....<TNAuthorizationList><DistributionPointName>...FS6aKdZeGsysoCo4H9P",
     ... FS6aKdZeGsysoCo4H9P",
   })
   - "signature": "uOvUfIk5RyQ...nw62Ay1c1eAB"

4. **KMS Sends challenge response to STI-CA**
   - SPC Token
     - X.509v3 extensions:
       - ASN.1 DER-encoded
       - <TNAuthorizationList>
       - <DistributionPointName>

5. **KMS finalizes order with STI-CA**
   - POST /acme/order/asdf/finalize HTTP/1.1
   - Host: sti-ca.com
   - Content-Type: application/jose+json
   - "protected": base64url({...})
   - "payload": base64url({
     "csr": "MIIBFTCbxAIABDBFMQ....<TNAuthorizationList><DistributionPointName>...FS6aKdZeGsysoCo4H9P",
     ... FS6aKdZeGsysoCo4H9P",
   })
   - "signature": "uOvUfIk5RyQ...nw62Ay1c1eAB"