SHAKEN Governance Model and Certificate Management Overview

ATIS-1000080
Service Provider Code Token

Private Key

HTTPS

HTTPS

HTTPS

HTTPS
| Interface used during Session Setup | 8/2/17 | 3 |
High level call/data flow for CM

1. The SP-KMS generates a STI public/private key pair for the service provider, for use by the STI-AS in signing the PASSporT. The SP-KMS securely distributes the SP STI private key to the SKS.

2. The SP-KMS sends a request for a token to the STI-PA. The token will be used for service provider validation during the process of acquiring a certificate.

3. The SP-KMS selects an STI-CA. If it has not already done so, the ACME client on the SP-KMS registers with the STI-CA using the ACME credentials.

4. The ACME client on the SP-KMS then establishes request for a new certificate to the ACME server hosted on the STI-CA. The response to the request includes a URL with the authorization challenge.

5. The SP that is requesting the certificate responds to that challenge by providing the current valid token acquired from the STI-PA.

6. If not already cached, the STI-CA sends a request for a public key certificate to the STI-PA in order to validate that the signature of the token has been signed by the STI-PA. Once the STI-CA receives the indication that the service provider is authorized, the STI-CA can issue the certificate.

7. In parallel with step 5, the ACME client starts polling for the status to determine if the service provider has been authorized to get a certificate and whether a certificate is available. Once the certificate has been issued, the ACME client downloads the certificate for use by the SP-KMS.

8. The SP-KMS notifies the STI-AS that the public key certificate is available through implementation specific means (e.g., SIP MESSAGE or WEBPUSH).

9. The SP-KMS puts the public key certificate in the STI-CR.
Service Provider Code Token (JWT)

JWT Header:
- **alg**: Defines the algorithm used in the signature of the token. For Service Provider Code tokens, the algorithm MUST be "ES256".
- **typ**: Set to standard "JWT" value.
- **x5u**: Defines the URL of the certificate of the STI-PA validating the Service Provider Code.

JWT Payload:
- **sub (*)**: Service Provider Code value being validated in the form of a JSON array of ASCII strings.
- **iat**: DateTime value of the time and date the token was issued.
- **nbf**: DateTime value of the starting time and date that the token is valid.
- **exp**: DateTime value of the ending time and date that the token expires.
- **fingerprint**: (Certificate) key fingerprint of the ACME credentials the Service Provider used to create an account with the CA.

- **“fingerprint”** is of the form:
  - base64url(JWK_Thumbprint(accountKey))
  - * For ATIS-1000080, only a single Service Provider Code is required in the “sub” field.
Certificate format

• X.509 v3 certificate (RFC 5280) syntax with STIR extensions (draft-ietf-stir-certificates):

Certificate ::= SEQUENCE {
  tbsCertificate    TBSCertificate,
  signatureAlgorithm AlgorithmIdentifier,
  signatureValue     BIT STRING }

TBSCertificate ::= SEQUENCE
  version     Version
  serialNumber CertificateSerialNumber,
  signature    AlgorithmIdentifier,
  issuer       Name,
  validity     Validity,
  subject      Name,
  subjectPublicKeyInfo SubjectPublicKeyInfo,
  issuerUniqueID [1] IMPLICIT UniqueIdentifier OPTIONAL,
    -- If present, version MUST be v2 or v3
  subjectUniqueID [2] IMPLICIT UniqueIdentifier OPTIONAL,
    -- If present, version MUST be v2 or v3
  extensions   [3] EXPLICIT Extensions OPTIONAL
    -- If present, version MUST be v3
}

Distinguished Name optional fields:
• countryName (C=) (e.g. US)
• organizationName (O=) (e.g. company name)
• organizationalUnitName (OU=) (e.g. Residential Voice or Wholesale Services)
• stateOrProvinceName (ST=) (e.g. PA)
• localityName (L=) (e.g. Philadelphia)
• commonName (CN=)

Note: If any of these attributes are filled out, generally they SHOULD be validated as claims in the token provided by STI-PA as valid contact and address strings.
Certificate format (continued)

Version ::= INTEGER { v1(0), v2(1), v3(2) }
CertificateSerialNumber ::= INTEGER
Validity ::= SEQUENCE {
   notBefore
   notAfter
Time ::= CHOICE {
   utcTime
   generalTime
Time, Time }
UTCTime,
GeneralizedTime }
UniqueIdenIfier ::= BIT STRING
SubjectPublicKeyInfo ::= SEQUENCE {
   algorithm AlgorithmIdentifier,
   subjectPublicKey BIT STRING }
Extensions ::= SEQUENCE SIZE (1..MAX) OF Extension
Certificate format – STIR specific Extensions

TNAuthorizationList ::= SEQUENCE SIZE (1..MAX) OF TNAuthorization

TNAuthorization ::= SEQUENCE SIZE (1..MAX) OF TNEntry

TNEntry ::= CHOICE {
    spc [0] ServiceProviderlCodeList,
    range [1] TelephoneNumberRange,
    one   E164Number }

ServiceProviderCodeList ::= SEQUENCE SIZE (1..3) OF OCTET STRING

-- When all three are present: Service Provider Code, Alt Service Provider Code, and Last Alt Service Provider Code

TelephoneNumberRange ::= SEQUENCE {
    start E164Number,
    count INTEGER }

E164Number ::= IA5String (SIZE (1..15)) (FROM ("0123456789"))

Note: OID for TNAuthorization List is 26
Certificate Example

Data:
Version: 3 (0x2)
Serial Number: 6734468596164949790 (0x5d75a381e96f771e)
Signature Algorithm: sha256WithRSAEncryption
Issuer: CN=CallAuthnCA, O=STI-CA-xyz IOT Lab, C=US
Validity
Not Before: May 10 20:19:22 2017 GMT
Not After: May 10 20:19:22 2019 GMT
Subject: CN=SHAKEN, OU=VOIP, O=AcmeTelecom, Inc.,
L=Bridgewater, ST=NJ, C=US
Subject Public Key Info:
Public Key Algorithm: id-ecPublicKey
Public-Key: (256 bit)
  pub:
  93:ca:d7:3f:9e:b7:8e:c7:70:6b:e2:d2:0e:8e:79:
  0c:5a:38:b8:a5:fd:52:5d:db:43:bf:00:b1:cd:df:
  74:c2:96:9c:22
ASN1 OID: prime256v1
X509v3 extensions:
  1.3.6.1.5.5.7.1.26:
0.....1234
X509v3 Subject Key Identifier:
91:43:70:C5:C6
X509v3 Basic Constraints: critical
CA:FALSE
X509v3 Authority Key Identifier:
  83:31:54:D0:C8