

SHAKEN Governance Model and Certificate Management Overview

ATIS-1000080

mbarnes@iconectiv.com

Overview – Certificate Management

- ATIS-1000080 Introduces new functional elements to the SHAKEN architecture (ATIS-1000074) to support the Certificate Provisioning logical function:
 - STI-CA: Secure Telephone Identity Certification Authority
 - *STI-CR: Secure Telephony Authority Certificate Repository (defined in ATIS-1000074)*
 - STI-PA: Secure Telephony Authority Policy Administrator
 - SP-KMS: Service Provider Key Management Server
 - SKS: (Service Provider) Secure Key Store

STI-CA

- Roles and responsibilities of the STI-CA align with those of traditional PKI (RFC 5280).
- New extension (TNAuthList, OID 26) added to CSR/certificate to support unique STI identifier requirements (draft-ietf-stir-certificates)
- Interface to STI-CA from SP-KMS uses an automated certificate management protocol (ACME) (draft-ietf-acme-acme)
 - New “challenge” Identifier and Type defined to support authorization of service providers to obtain certificates (draft-ietf-acme-service-provider)
- Interfaces with STI-PA to get public key certificate for validating Service Provider Code token.

STI-CR

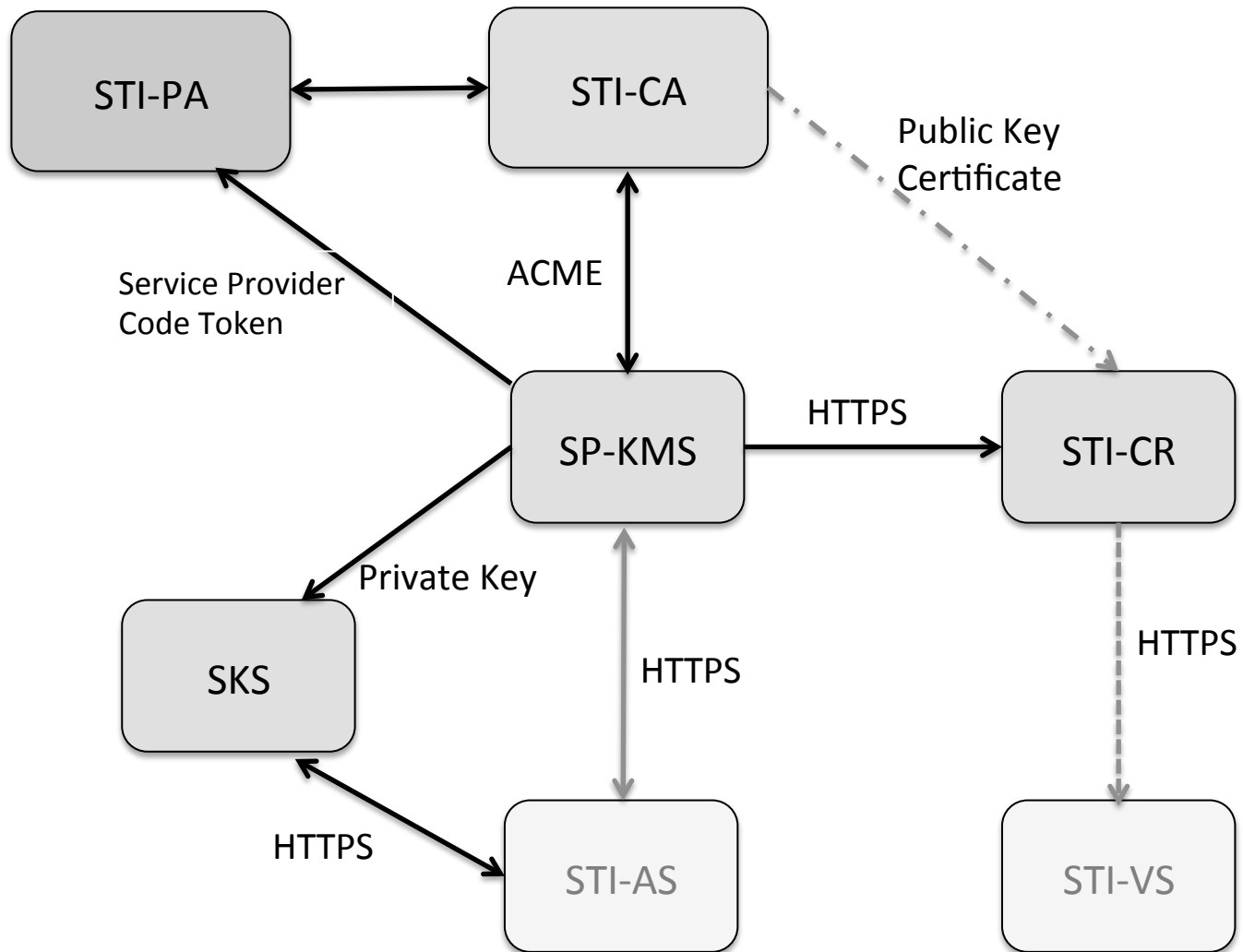
- STI-VS gets public key certificate used to sign the Identity header field from the STI-CR during the verification process.
- No new functionality or interfaces required.
- Follows existing procedures as defined in RFC 5280.

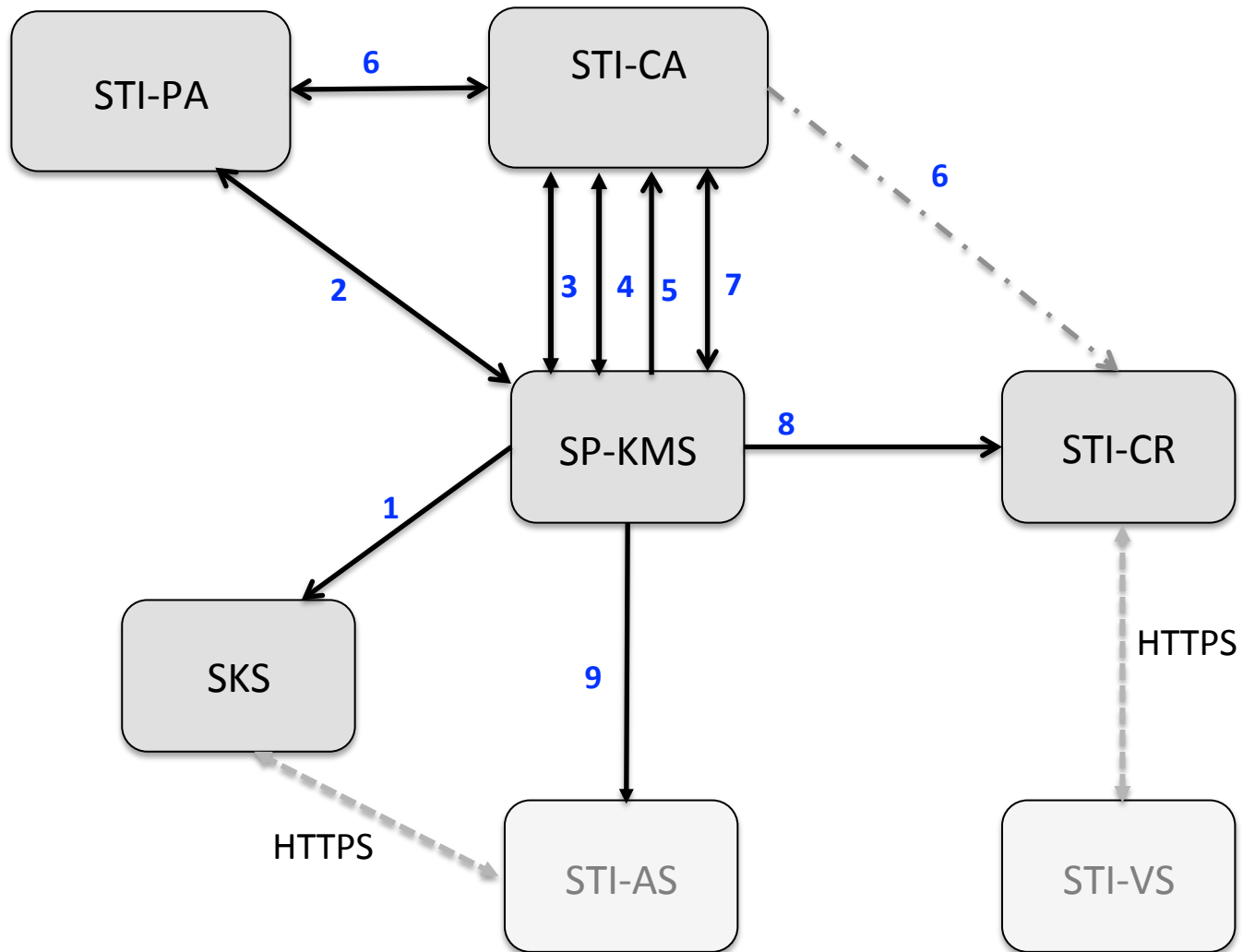
STI-PA

- Supports unique requirements of managing PKI infrastructure for STI and Service Provider's interactions with the PKI:
 - Serves as the Trust Authority for the PKI
 - Maintains a Trust List of approved STI-Cas
 - Serves as a Trust Anchor providing valid service providers with a unique token for authorization to get STI certificates
- Serves no active role in the issuance or validation of certificates:
 - Traditional PKI mechanisms for certificate validation are followed during the verification process

SP-KMS

- Provides the Service Provider's interface to the PKI.
- Hosts the ACME client which maintains an account with the ACME server hosted by the STI-CA
- Distributes private key to a Secure Key Store for access by the STI Authentication Service when signing the PASSporT in the Identity header field.
- Ensures the STI-AS has access to the public key certificate for inclusion in the Identity header field.



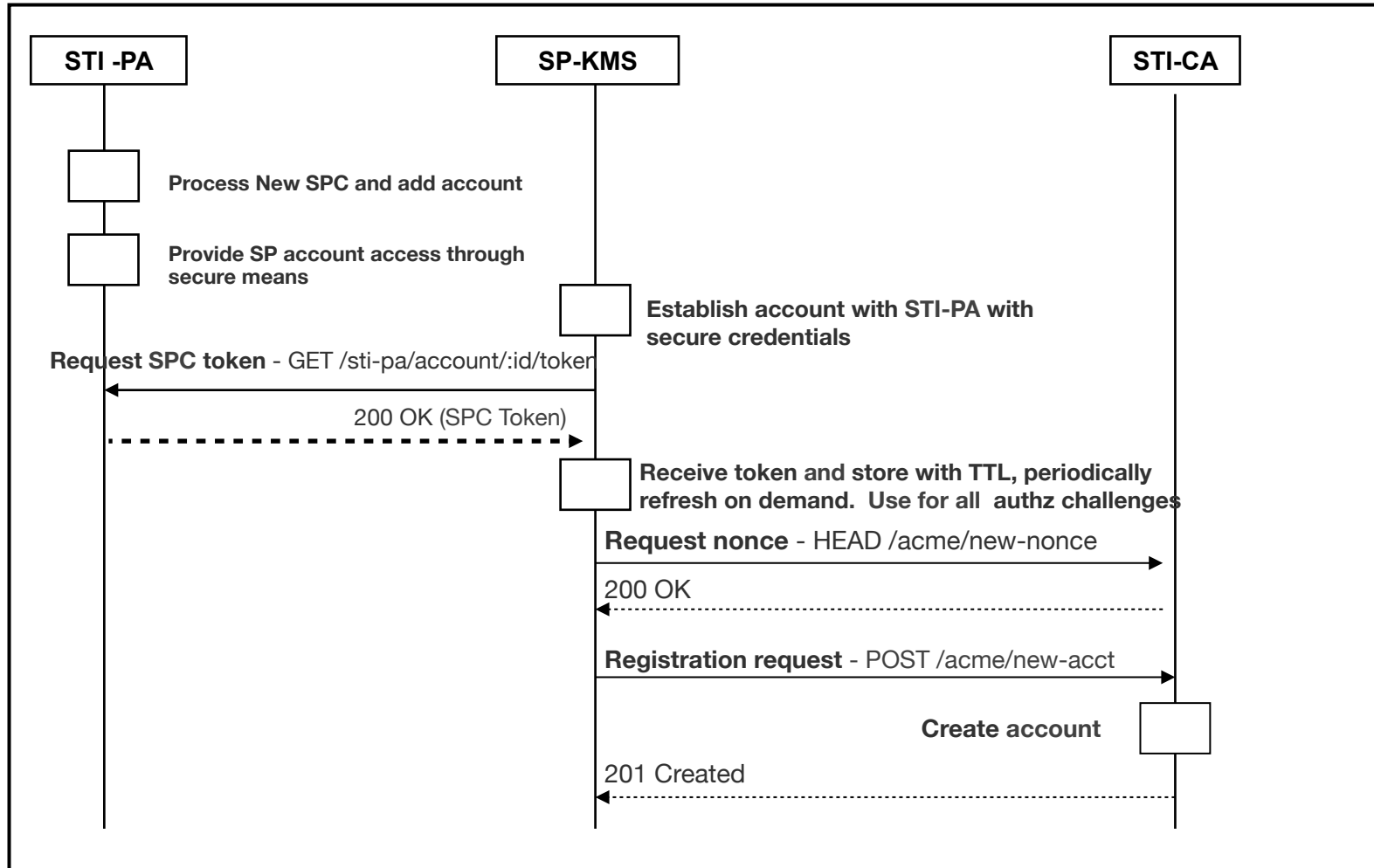


⬅️.....➡️ Interface used during Session Setup
- . . . ➡️ Interface not detailed in ATIS-1000080

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, storing the certificate in the **STI-CR**.
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.
8. *Once the certificate has been issued, the **ACME** client downloads the certificate for use by the **SP-KMS**. (Note: slight difference from ATIS-1000080, which leaves out interaction with STI-CR)*
9. The **SP-KMS** notifies the **STI-AS** that the public key certificate is available through implementation specific means (e.g., SIP MESSAGE or WEBPUSH).

STI-PA Account Registration & token creation



Service Provider Code Token (JWT)

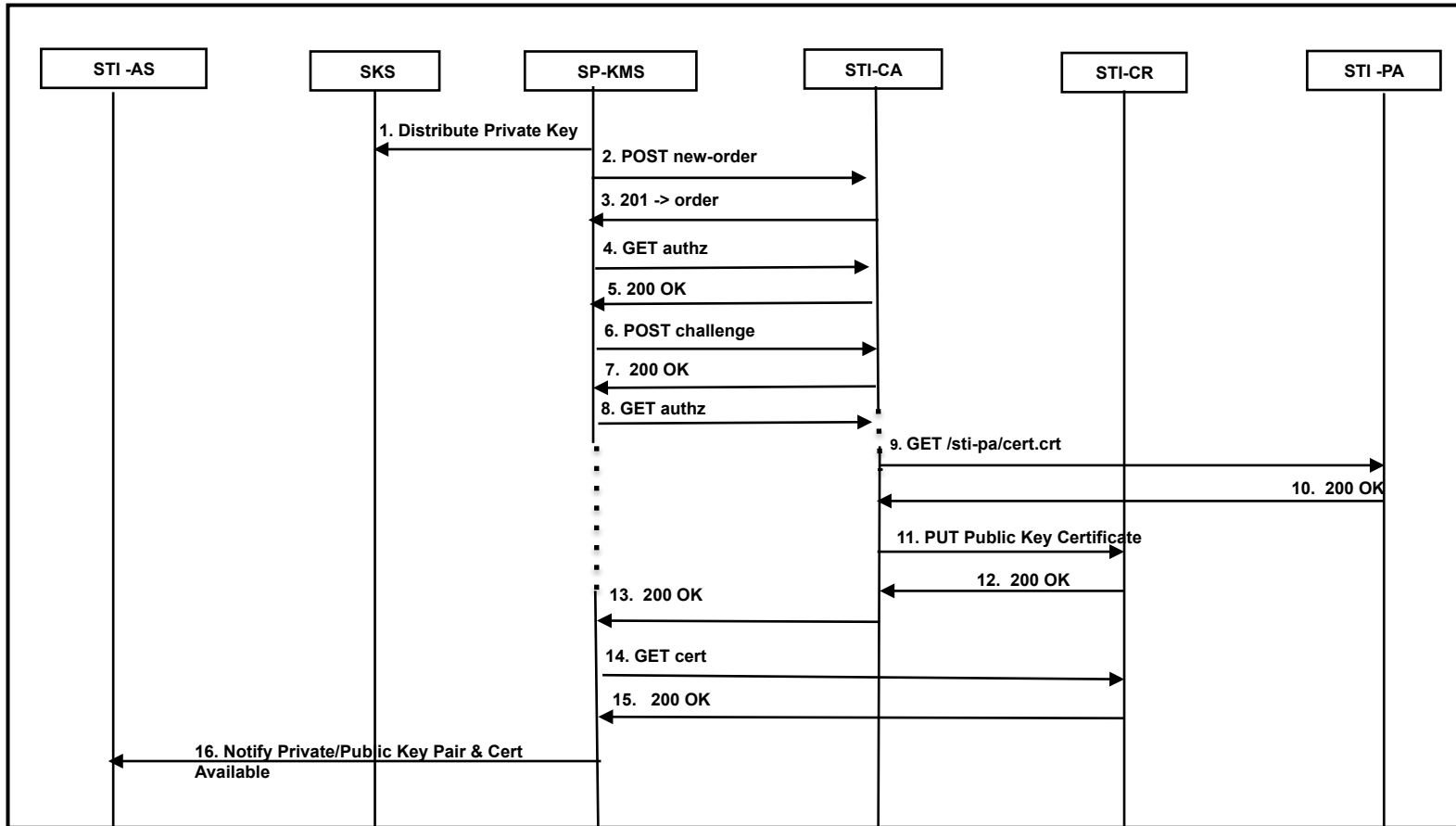
JWT Header:

- 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.

Detailed flow for CM



Information/control flow for CM

1. The **SP-KMS** uses normal PKI mechanisms to generate a public/private key pair for use by the STI-AS. **The SP-KMS** distributes the private key to the **SKS**.
2. The **ACME** client then sends an HTTP POST with a new order for a certificate.
3. The **ACME** server sends a 201 response with the certificate order object, with a status of “pending” that also includes (a pointer to) the challenges to be answered in order to authorize the SP-KMS to request certificates for the specific service provider represented by the SPID.
4. The **ACME client** sends an HTTP GET for the authorization object containing the challenges.
5. The **ACME server** sends a 200 response with the challenges.
6. The **ACME client** sends an HTTP POST, including the Service Provider Code token, in response to the challenges.
7. The **ACME server** sends a 200 response indicating the issuance is “pending”.
8. After the **ACME** client responds to the challenges, it starts polling for the status of the authorization of the service provider by send an HTTP GET for the authorization object.

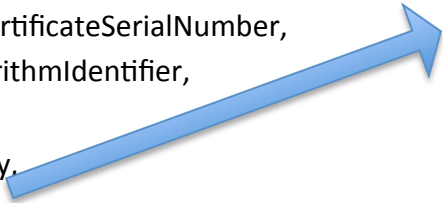
Information/control flow for CM

9. Once the **STI-CA** receives the response to the challenges from the **ACME** client, the **STI-CA** determines whether the service provider requesting the certificate is authorized by sending a request to get the public key for the service provider from the **STI-PA** to validate the signature of the Service Provider Code token.
10. If the service provider is authorized, the **STI-CA** changes the status in the authorization object to “valid”.
11. The **STI-CA** puts the issued public key certificate in the **STI-CR**.
12. The **STI-CR** sends a 200 response to the **STI-CA** indicating the public key certificate is available for use.
13. Thus, the next 200 response to the HTTP GET from the **ACME** client contains this authorization object indicating that the service provider has been authorized, as well as a URL to the certificate in the **STI-CR**.
14. The **ACME** client uses the URL in the authorization object and sends a request to get the certificate from the **STI-CR**.
15. The **STI-CR** sends the certificate to the **ACME** client in the 200 response.
16. The **SP-KMS** notifies the **STI-AS** that the public/private key pair is available along with the certificate.

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 }

UniqueIdentifier ::= 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] ServiceProviderCodeList,

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:

04:77:c6:b0:d6:df:fd:1f:0a:23:dc:40:24:a4:ea:

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:

d4:cf:cb:69:35:13:d1:52:9a:e3:10:fe:1b:51:5b:

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:

ED:87:91:08:DA:FC:82:A8:8A:CD:56:F5:A1:D6:7A:

91:43:70:C5:C6

X509v3 Basic Constraints: critical

CA:FALSE

X509v3 Authority Key Identifier:

keyid:03:93:A5:3B:9B:2E:8B:14:D6:C4:CF:58:CF:46:DB:
83:31:54:D0:C8