ATIS-1000080

ATIS Standard on

Signature-based Handling of Asserted information using toKENs (SHAKEN): Governance Model and Certificate Management

Alliance for Telecommunications Industry Solutions

Approved July 11, 2017

Abstract

Signature-based Handling of Asserted information using toKENs (SHAKEN) is an industry framework for managing and deploying Secure Telephone Identity (STI) technologies with the purpose of providing end-to-end cryptographic authentication and verification of the telephone identity and other information in an IP-based service provider voice network. This specification expands the SHAKEN framework, introducing a governance model and defining X.509 certificate management procedures. Certificate management provides mechanisms for validation of a certificate and verification of the associated digital signature, allowing for the identification of illegitimate use of national telecommunications infrastructure.

Foreword

The Alliance for Telecommunication Industry Solutions (ATIS) serves the public through improved understanding between providers, customers, and manufacturers. The Packet Technologies and Systems Committee (PTSC) develops and recommends standards and technical reports related to services, architectures, and signaling, in addition to related subjects under consideration in other North American and international standards bodies. PTSC coordinates and develops standards and technical reports relevant to telecommunications networks in the U.S., reviews and prepares contributions on such matters for submission to U.S. International Telecommunication Union Telecommunication Sector (ITU-T) and U.S. ITU Radiocommunication Sector (ITU-R) Study Groups or other standards organizations, and reviews for acceptability or per contra the positions of other countries in related standards development and takes or recommends appropriate actions.

The SIP Forum is an IP communications industry association that engages in numerous activities that promote and advance SIP-based technology, such as the development of industry recommendations, the SIPit, SIPconnect-IT, and RTCWeb-it interoperability testing events, special workshops, educational seminars, and general promotion of SIP in the industry. The SIP Forum is also the producer of the annual SIP Network Operators Conference (SIPNOC), focused on the technical requirements of the service provider community. One of the Forum's notable technical activities is the development of the SIPconnect Technical Recommendation – a standards-based SIP trunking recommendation for direct IP peering and interoperability between IP Private Branch Exchanges (PBXs) and SIP-based service provider networks. Other important Forum initiatives include work in Video Relay Service (VRS) interoperability, security, Network-to-Network Interoperability (NNI), and SIP and IPv6.

Suggestions for improvement of this document are welcome. They should be sent to the Alliance for Telecommunications Industry Solutions, PTSC, 1200 G Street NW, Suite 500, Washington, DC 20005, and/or to the SIP Forum, 733 Turnpike Street, Suite 192, North Andover, MA, 01845.

The mandatory requirements are designated by the word *shall* and recommendations by the word *should*. Where both a mandatory requirement and a recommendation are specified for the same criterion, the recommendation represents a goal currently identifiable as having distinct compatibility or performance advantages. The word *may* denotes an optional capability that could augment the standard. The standard is fully functional without the incorporation of this optional capability.

The **ATIS/SIP Forum IP-NNI Task Force** under the **ATIS Packet Technologies and Systems Committee (PTSC)** and the **SIP Forum Technical Working Group (TWG)** was responsible for the development of this document.

Table of Contents

Table of Figures

ATIS Standard on –

SHAKEN: Governance Model and Certificate Management

1 Scope & Purpose

1.1 Scope

This document expands the Signature-based Handling of Asserted Information using Tokens (SHAKEN) [ATIS-1000074] framework, introducing a governance model and defining certificate management procedures for Secure Telephone Identity (STI) technologies. The certificate management procedures identify the functional entities and protocols involved in the distribution and management of STI Certificates. The governance model identifies functional entities that have the responsibility to establish policies and procedures to ensure that only authorized entities are allowed to administer digital certificates within Voice over Internet Protocol (VoIP) networks. However, the details of these functional entities in terms of regulatory control and who establishes and manages those entities are outside the scope of this document.

1.2 Purpose

This document introduces a governance model, certificate management architecture, and related protocols to the SHAKEN framework [ATIS-1000074]. The governance model defines recommended roles and relationships, such that the determination of who is authorized to administer and use digital certificates in VoIP networks can be established. This model includes sufficient flexibility to allow specific regulatory requirements to be implemented and evolved over time, minimizing dependencies on the underlying mechanisms for certificate management. The certificate management architecture is based on the definition of roles similar to those defined in "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", Internet Engineering Task Force (IETF) [RFC 5280]. Per the SHAKEN framework, the certificates themselves are based on X.509 with specific policy extensions based on draft-ietf-stir-certificates. The objective of this document is to provide recommendations and requirements for implementing the protocols and procedures for certificate management within the SHAKEN framework.

2 Normative References

The following standards contain provisions which, through reference in this text, constitute provisions of this Standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this Standard are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below.

ATIS-1000074, *Signature-based Handling of Asserted Information using Tokens (SHAKEN).*[1]

ATIS-0300251, *Codes for Identification of Service Providers for Information Exchange.*[2]

ATIS-1000054*, ATIS Technical Report on Next Generation Network Certificate Management.*[3]

draft-ietf-stir-passport, *Personal Assertion Token (PASSporT)*.[4]

draft-ietf-stir-rfc4474bis, *Authenticated Identity Management in the Session Initiation Protocol*.[4]

---

[1] This document is available from the Alliance for Telecommunications Industry Solutions (ATIS) at: < https://www.atis.org/docstore/product.aspx?id=28297 >.

[2] This document is available from ATIS at: < https://www.atis.org/docstore/product.aspx?id=26148 >.

[3] This document is available from ATIS at: < https://www.atis.org/docstore/product.aspx?id=27962 >.

[4] This document is available from the Internet Engineering Task Force (IETF) at: < https://tools.ietf.org/ >.

draft-ietf-stir-certificates, *Secure Telephone Identity Credentials: Certificates*[4]

IETF RFC 5280, *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile.*[4]

draft-ietf-acme-acme, *Automatic Certificate Management Environment (ACME).*[4]

draft-barnes-acme-service-provider, *ACME Identifiers and Challenges for VoIP Service Providers.*[4]

RFC 2986, *PKCS #10: Certification Request Syntax Specification Version 1.7.*[4]

RFC 3261, *SIP: Session Initiation Protocol.*[4]

RFC 3966, *The tel URI for Telephone Numbers.*[4]

RFC 4949, *Internet Security Glossary, Version 2.*[4]

RFC 5246, *The Transport Layer Security (TLS) Protocol Version 1.2.*[4]

RFC 5958, *Asymmetric Key Package.*[4]

RFC 6749, *The OAuth 2.0 Authorization Framework.*[4]

RFC 6960, *Online Certificate Status Protocol (OSCP).*[4]

RFC 7159, *The JavaScript Object Notation (JSON).*[4]

RFC 7231, *Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content.*[4]

RFC 7375, *Secure Telephone Identity Threat Model.*[4]

RFC 7515, *JSON Web Signatures (JWS).*[4]

RFC 7516, *JSON Web Algorithms (JWA).*[4]

RFC 7517, *JSON Web Key (JWK).*[4]

RFC 7519, *JSON Web Token (JWT).*[4]

3 Definitions, Acronyms, & Abbreviations

For a list of common communications terms and definitions, please visit the *ATIS Telecom Glossary*, which is located at < http://www.atis.org/glossary >.

3.1 Definitions

The following provides some key definitions used in this document. Refer to IETF RFC 4949 for a complete Internet Security Glossary, as well as tutorial material for many of these terms.

Caller ID: The originating or calling party's telephone number used to identify the caller carried either in the P-Asserted-Identity or From header fields in the Session Initiation Protocol (SIP) [RFC 3261] messages.

(Digital) Certificate: Binds a public key to a Subject (e.g., the end-entity). A certificate document in the form of a digital data object (a data object used by a computer) to which is appended a computed digital signature value that depends on the data object. [RFC 4949]. See also STI Certificate.

Certification Authority (CA): An entity that issues digital certificates (especially X.509 certificates) and vouches for the binding between the data items in a certificate. [RFC 4949].

Certificate Validation: An act or process by which a certificate user established that the assertions made by a certificate can be trusted. [RFC 4949].

Certificate Revocation List (CRL): A data structure that enumerates digital certificates that have been invalidated by their issuer prior to when they were scheduled to expire. [RFC 4949]

Chain of Trust: Deprecated term referring to the chain of certificates to a Trust Anchor. Synonym for Certification Path or Certificate Chain. [RFC 4949].

Certificate Chain: See Certification Path.

Certification Path: A linked sequence of one or more public-key certificates, or one or more public-key certificates and one attribute certificate, that enables a certificate user to verify the signature on the last certificate in the path, and thus enables the user to obtain (from that last certificate) a certified public key, or certified attributes, of the system entity that is the subject of that last certificate. Synonym for Certificate Chain. [RFC 4949].

Certificate Signing Request (CSR): A CSR is sent to a CA to request a certificate. A CSR contains a Public Key of the end-entity that is requesting the certificate.

Company Code: A unique four-character alphanumeric code (NXXX) assigned to all Service Providers [ATIS-0300251].

End-Entity: An entity that participates in the Public Key Infrastructure (PKI). Usually a Server, Service, Router, or a Person. In the context of SHAKEN, it is the Service Provider on behalf of the originating endpoint.

Fingerprint: A hash result ("key fingerprint") used to authenticate a public key or other data [RFC 4949].

Identity: Either a canonical Address-of-Record (AoR) SIP Uniform Resource Identifier (URI) employed to reach a user (such as 'sip:alice@atlanta.example.com'), or a telephone number, which commonly appears in either a TEL URI [RFC 3966] or as the user portion of a SIP URI. See also Caller ID [draft-ietf-stir-4474bis].

National/Regional Regulatory Authority (NRRA): A governmental entity responsible for the oversight/regulation of the telecommunication networks within a specific country or region.

NOTE: Region is not intended to be a region within a country (e.g., a region is not a state within the US).

Online Certificate Status Protocol (OCSP): An Internet protocol used by a client to obtain the revocation status of a certificate from a server.

Private Key: In asymmetric cryptography, the private key is kept secret by the end-entity. The private key can be used for both encryption and decryption [RFC 4949].

Public Key: The publicly disclosable component of a pair of cryptographic keys used for asymmetric cryptography [RFC 4949].

Public Key Infrastructure (PKI): The set of hardware, software, personnel, policy, and procedures used by a CA to issue and manage certificates [RFC 4949].

Root CA: A CA that is directly trusted by an end-entity. See also Trust Anchor CA and Trusted CA [RFC 4949].

Secure Telephone Identity (STI) Certificate: A public key certificate used by a service provider to sign and verify the PASSporT.

Service Provider Code: In the context of this document, this term refers to any unique identifier that is allocated by a Regulatory and/or administrative entity to a service provider. In the US and Canada this would be a Company Code as defined in [ATIS-0300251].

Signature: Created by signing the message using the private key. It ensures the identity of the sender and the integrity of the data [RFC 4949].

Telephone Identity: An identifier associated with an originator of a telephone call. In the context of the SHAKEN framework, this is a SIP identity (e.g., a SIP URI or a TEL URI) from which a telephone number can be derived.

Trust Anchor: An established point of trust (usually based on the authority of some person, office, or organization) from which a certificate user begins the validation of a certification path. The combination of a trusted public key and the name of the entity to which the corresponding private key belongs. [RFC 4949].

Trust Anchor CA: A CA that is the subject of a trust anchor certificate or otherwise establishes a trust anchor key. See also Root CA and Trusted CA [RFC 4949].

Trusted CA: A CA upon which a certificate user relies for issuing valid certificates; especially a CA that is used as a trust anchor CA [RFC 4949].

Trust Model: Describes how trust is distributed from Trust Anchors.

3.2 Acronyms & Abbreviations

ACME	Automated Certificate Management Environment (Protocol)
ASCII	American Standard Code for Information Interchange
AoR	Address-of-Record
ATIS	Alliance for Telecommunications Industry Solutions
CA	Certification Authority
CORS	Cross-Origin Resource Sharing
CRL	Certificate Revocation List
CSR	Certificate Signing Request
DER	Distinguished Encoding Rules
DN	Distinguished Name
DNS	Domain Name System
ECDSA	Elliptic Curve Digital Signature Algorithm
HTTPS	Hypertext Transfer Protocol Secure
IETF	Internet Engineering Task Force
JDK	Java Development Kit
JSON	JavaScript Object Notation
JWA	JSON Web Algorithms
JWK	JSON Web Key
JWS	JSON Web Signature
JWT	JSON Web Token
NECA	National Exchange Carrier Association
NNI	Network-to-Network Interface
NRRA	National/Regional Regulatory Authority
OAuth	Open Authentication (Protocol)
OCN	Operating Company Number
OCSP	Online Certificate Status Protocol
PASSporT	Personal Assertion Token
PKI	Public Key Infrastructure
PKIX	Public Key Infrastructure for X.509 Certificates
PSTN	Public Switched Telephone Network
SHAKEN	Signature-based Handling of Asserted information using toKENs

SIP	Session Initiation Protocol
REST	Representational State Transfer
SKS	Secure Key Store
SMI	Structure of Management Information
SP	Service Provider
SP-KMS	SP Key Management Server
STI	Secure Telephone Identity
STI-AS	Secure Telephone Identity Authentication Service
STI-CA	Secure Telephone Identity Certification Authority
STI-CR	Secure Telephone Identity Certificate Repository
STI-GA	Secure Telephone Identity Governance Authority
STI-PA	Secure Telephone Identity Policy Administrator
STI-VS	Secure Telephone Identity Verification Service
STIR	Secure Telephone Identity Revisited
TLS	Transport Layer Security
TN	Telephone Number
URI	Uniform Resource Identifier
VoIP	Voice over Internet Protocol

4 Overview

This document introduces a governance model and defines certificate management procedures for the SHAKEN framework [ATIS-1000074]. The SHAKEN framework establishes an end-to-end architecture that allows an originating Service Provider to authenticate and assert a telephone identity and provides for the verification of this telephone identity by a terminating service provider. The SHAKEN framework defines a profile, using protocols standardized in the IETF Secure Telephone Identity Revisited (STIR) Working Group (WG). This document provides recommendations and requirements for implementing these IETF specifications, draft-ietf-stir-passport, draft-ietf-stir-rfc4474bis, and draft-ietf-stir-certificates, to support management of Service Provider-level certificates within the SHAKEN framework.

The SHAKEN framework uses X.509 certificates, as defined in “Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile”, IETF [RFC 5280], to verify the digital signatures associated with SIP identifiers. The governance model is described in clause 5 of this document. Clause 6 then defines the protocols and procedures used to create and manage STI certificates using the recommended governance model where there is a central policy administrator who authorizes Service Providers to acquire certificates from trusted Certification Authorities (CAs).

5 SHAKEN Governance Model

This clause introduces a governance model to support STI, defining two new functional entities: an STI Governance Authority (STI-GA) and an STI Policy Administrator (STI-PA). Clause 5.1 defines baseline requirements that lead to this model, and clause 5.2 defines the roles and responsibilities of these functional elements and the relationship of the STI-PA to the STI Certification Authority (STI-CA) and Service Provider.

5.1 Requirements for Governance of STI Certificate Management

The governance, creation, and management of certificates to support STI introduce the following requirements:

1) A PKI infrastructure to manage and issue the STI certificates, including a trust model.
2) A mechanism to authorize Service Providers to be issued STI certificates.
3) An entity to define the policies and procedures around who can acquire STI certificates.
4) An entity to establish policies around who can manage the PKI and issue STI certificates.
5) An entity to apply the policies and procedures established for STI certificate management.

Clause 5.2 defines a certificate governance model to support these requirements.

5.2 Certificate Governance: Roles & Responsibilities

The SHAKEN governance model for STI certificate management is illustrated in the following diagram.

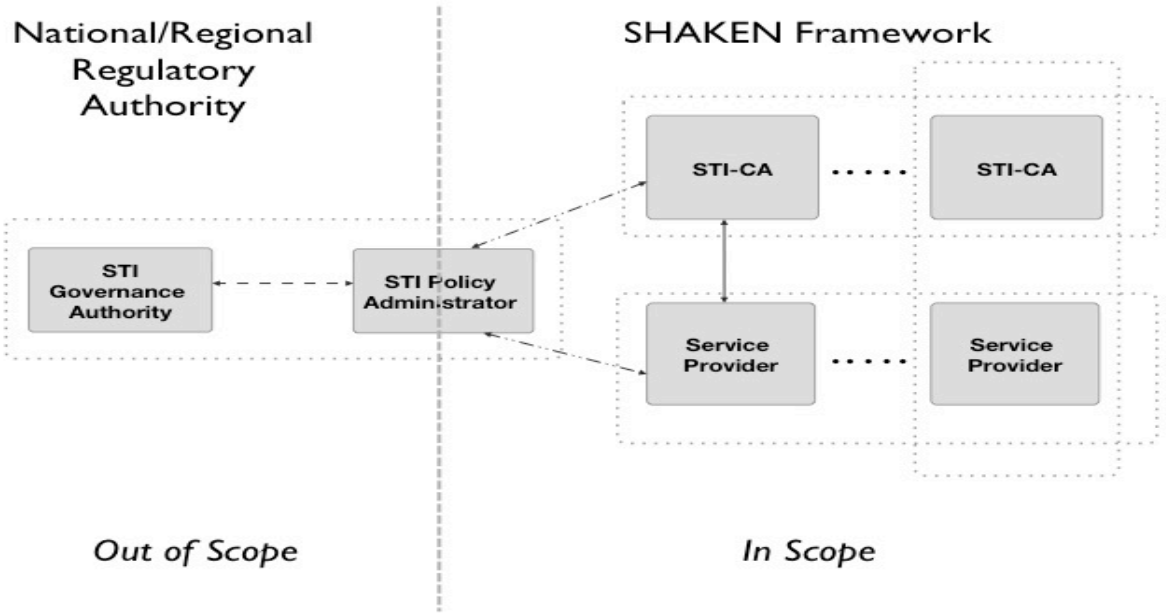

Figure 5.1 – Governance Model for Certificate Management

This diagram identifies the following roles associated with governance and STI certificate management:

- Secure Telephone Identity Governance Authority (STI-GA).
- Secure Telephone Identity Policy Administrator (STI-PA).
- Secure Telephone Identity Certification Authority (STI-CA).
- Service Provider (SP).

The STI-GA serves in an oversight role for the policies established or endorsed by a National/Regional Regulatory Authority (NRRA). The SHAKEN governance model assumes there is only one STI-GA for a given country or region.

The STI-GA is responsible for:

- Defining the policies and procedures governing which entities can acquire STI certificates.
- Establishing policies governing which entities can manage the PKI and issue STI certificates.

There is a relationship required between the STI-GA and the STI-PA as the latter serves in a policy enforcement role for the policies defined by the former. The STI-GA role satisfies requirements 3 and 4 in clause 5.1. The STI-PA role satisfies requirement 5 in clause 5.1. The STI-GA and the STI-PA are defined as distinct roles in this model, though in practice both roles could be performed by a single entity.

> NOTE: The details of the policies and procedures defined by the STI-GA and enforced by the STI-PA are outside the scope of this document.

This document specifies the protocols and message flows between the STI-PA, the Service Providers, and STI-CAs to support the issuance and management of certificates to support STI, satisfying the first two requirements identified in clause 5.1. The following clauses summarize the roles and responsibilities of these functional elements within the SHAKEN framework.

5.2.1 Secure Telephone Identity Policy Administrator (STI-PA)

The STI-PA serves in a policy enforcement role and is entrusted by the STI-GA to apply the defined rules and policies to confirm that Service Providers are authorized to request STI certificates and to authorize STI-CAs to issue STI certificates.

The STI-PA manages an active, secure list of approved STI-CAs in the form of their public key certificates. The STI-PA provides this list of approved STI-CAs to the service providers via a Hypertext Transfer Protocol Secure (HTTPS) interface. The SHAKEN-defined Secure Telephone Identity Verification Service (STI-VS) can then use a public key certificate to validate the root of the digital signature in the STI certificate by determining whether the STI-CA that issued the STI certificate is in the list of approved STI-CAs. Note that the details associated with the structure and management of this list require further specification, the details of which are outside the scope of this document.

The STI-PA also maintains a distinct X.509-based PKI for digitally signing Service Provider Code tokens, which represent the credentials and validation of SPs. An SP uses this Service Code token, which is a signed JSON Web Token (JWT), for validation when requesting issuance of STI certificates from an approved STI-CA. The mechanism by which the SP acquires the Service Provider Code token is described in clause 6.3.4.

The trust model for SHAKEN defines the STI-PA as the Trust Anchor for this token-based mechanism for validation of Service Providers within a national/regional administrative domain. For example, all STI certificates for the SP tokens in the United States would be associated with a single STI-PA Trust Anchor. Other countries could have a different Trust Anchor.

5.2.2 Secure Telephone Identity Certification Authority (STI-CA)

In the X.509 model, the STI-CA serves as the Root CA for the STI certificates used to digitally sign and verify telephone calls. The STI-CA provides the service of issuing valid STI certificates to the validated SPs. There will likely be a number of STI-CAs, supporting specific or multiple SPs, depending upon the SP. It is also worth noting that although the STI-CA and Service Provider are distinct roles, it would also be possible for a Service Provider to establish an internal STI-CA for its own use under the authority of the STI-PA.

In the North American telephone network, it is anticipated that the number of entities that would serve as STI-CAs is relatively small. However, this framework and architecture does not impose a specific limit.

5.2.3 Service Provider (SP)

The Service Provider obtains STI certificates from the STI-CA to create signatures authenticating the identity of originators of Session Initiation Protocol (SIP) requests. The SP can obtain STI certificates from any approved STI-CA in the list of approved CAs, which is received from the STI-PA. During account registration with the STI-PA, as detailed in clause 6.3.3, the SP selects the preferred STI-CA(s). During the verification process by the STI-VS, the

SP checks that the STI-CA that issued the STI certificate is in the list of approved STI-CAs received from the STI-PA.

In the context of the SHAKEN framework, STI certificates are not required for each originating telephone identity but rather, the same STI certificates can be used by a given SP to sign requests associated with multiple originators and SIP requests. The key aspect is that the identity-related information in the SIP requests is authenticated by the originating Service Provider and can be verified by the terminating Service Provider. Information contained within the Personal Assertion Token (PASSporT) in the SIP messages, attesting to a Service Provider's knowledge of specific telephone identities that the terminating SP can use to determine specific handling for a call. Details for the attestation are provided in [ATIS-1000074].

The SHAKEN certificate management framework is based on using a signed Service Provider Code token for validation when requesting an STI certificate. Prior to requesting a certificate, the SP requests a Service Provider Code token from the STI-PA as described in clause 6.3.4. When an SP initiates a certificate signing request, the SP proves that it has been validated and is eligible to receive an STI certificate via the use of the Service Provider Code token that is received from the STI-PA. Clause 6.3.5.2, steps 3 and 4, provide the details of the SP validation mechanism.

6 SHAKEN Certificate Management

Management of certificates for Transport Layer Security (TLS) [RFC 5246] and HTTPS [RFC 7231] based transactions on the Internet is a fairly well-defined and common practice for website and Internet applications. Generally, there are recognized certification authorities that can "vouch" for the authenticity of a domain owner based on out-of-band validation techniques such as e-mail and unique codes in the Domain Name System (DNS).

The certificate management model for SHAKEN is based on Internet best practices for PKI [ATIS-1000054] to the extent possible. The model is modified where appropriate to reflect unique characteristics of the service provider-based telephone network. STI certificates are initially expected to take advantage of service providers' recognized ability to legitimately assert telephone identities on a VoIP network. The fundamental requirements for SHAKEN certificate management are identified in clause 6.1. Clause 6.2 describes the functional elements added to the SHAKEN framework architecture to support certificate management. Clause 6.3 details the steps and procedures for the issuance of STI certificates.

6.1 Requirements for SHAKEN Certificate Management

This clause details the fundamental functionality required for SHAKEN certificate management. An automated mechanism for certificate management is preferred and includes the following fundamental functional requirements:

1) A mechanism to determine the STI-Certification Authorities (STI-CAs) that can be used when requesting STI certificates.
2) A procedure for registering with the STI-CA.
3) A process to request issuance of STI certificates.
4) A mechanism to validate the requesting Service Provider.
5) A process for adding public key STI certificates to a Certificate Repository.
6) A mechanism to renew/update STI certificates.
7) A mechanism to revoke STI certificates.

In terms of certificate issuance, the primary difference between Web PKI and the requirements for STI is the procedure to validate that the entity requesting a certificate is authorized to acquire STI certificates. Existing mechanisms for Web PKI, including the Automated Certificate Management Environment (ACME) protocol, rely on DNS or e-mail. STI uses a Service Provider Code token mechanism as described in clause 6.3.4.

6.2 SHAKEN Certificate Management Architecture

The following figure represents the recommended certificate management architecture for SHAKEN.

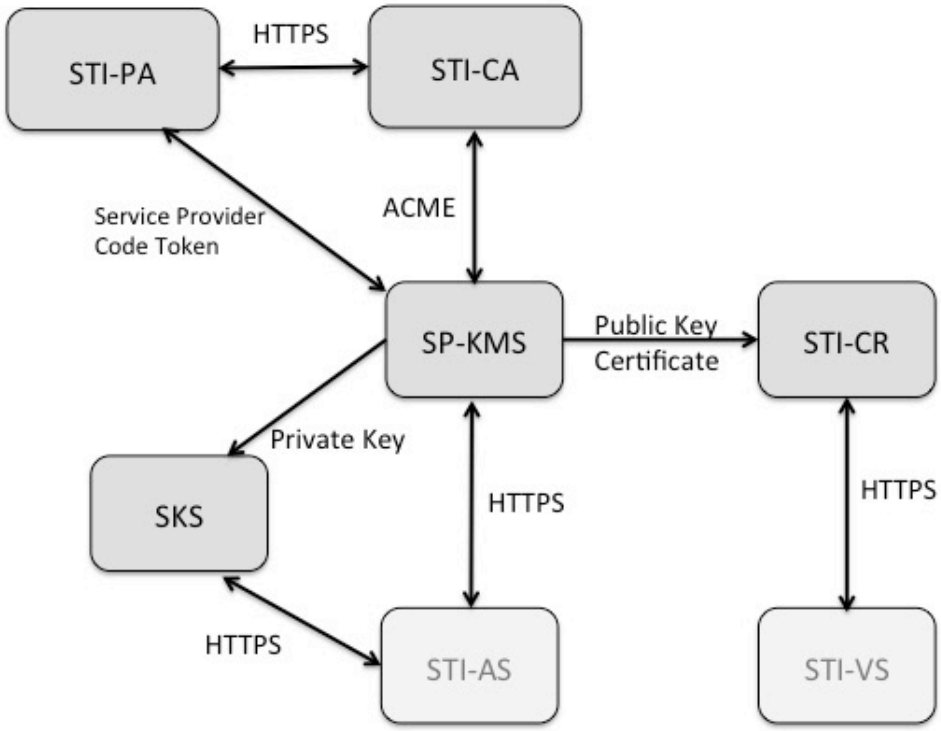

Figure 6.1 – SHAKEN Certificate Management Architecture

The above SHAKEN certificate management architecture introduces the following additional elements:

- Service Provider Key Management Server (SP-KMS) – The service provider's server that generates private/public key pair for signing, requests an STI certificate from the STI-CA, and receives the STI-CA signed public key certificate.
- Secure Key Store (SKS) – The store for private keys used by the originating service provider Authentication Service.
- Secure Telephone Identity Certificate Repository (STI-CR) – The HTTPS server that hosts the public key certificates used by the destination service provider's Verification Service to validate signatures.

NOTE: The STI-PA functional element introduced in clause 5.2.1 also plays a key role in the certificate management architecture and related procedures.

6.3 SHAKEN Certificate Management Process

This clause describes the detailed process for acquiring a signed public key certificate. It is based on an automated approach using the ACME protocol. Readers can also refer to Appendix A which illustrates an example of the steps for certificate creation and validation using openSSL.

Clause 6.3.1 lists the necessary functions in the process and provides a high level flow. Subsequent clauses describe the specific details for using the ACME protocol for each of the STI certificate management functions.

6.3.1 SHAKEN Certificate Management Flow

This clause describes the detailed STI certificate management process and the interaction model between the Service Provider, the STI-PA, and the STI-CA for acquiring STI certificates.

The SHAKEN certificate management process encompasses the following high level process functions that will be performed by the Service Provider and are detailed in the subsequent clauses of the document:

- STI-PA Account Registration and Service Provider Authorization.
- STI-CA Account Creation.
- Service Provider Code token acquisition.
- Application for a Public Key Certificate.
- STI certificate acquisition.
- Lifecycle Management of STI certificates (including Revocation).

The certificate management process follows two main flows:

1. The STI-PA has a two-party Open Authentication (Protocol) (OAuth) [RFC 6749]-style HTTP interface with the Service Provider in order to provide a token the Service Provider can use for authorization by the STI-CA when requesting a certificate.

   NOTE: Per clause 5.2.1, the STI-PA maintains a list of approved STI-CAs that are authorized to create STI certificates.

2. The Service Provider uses the ACME [draft-ietf-acme-acme] protocol for interfacing to the STI-CA for the acquisition of STI certificates. ACME is a Representational State Transfer (REST) services-based request and response protocol that uses HTTPS as a transport.

Typical HTTP caching of resources with long lives (e.g., certificates, tokens, etc.) is recommended, although not required, to minimize overall transaction delays whenever possible. Another consideration for the HTTP interface is the requirement for a secure interface using TLS [RFC 5246] (i.e., HTTPS). HTTP redirects shall not be allowed. Additional considerations on the use of HTTPS for ACME are provided in section 5.1 of draft-ietf-acme-acme. Since an ACME server supporting SHAKEN is not intended to be generally accessible, Cross-Origin Resource Sharing (CORS) shall not be used.

The processing flow for certificate management is as follows:

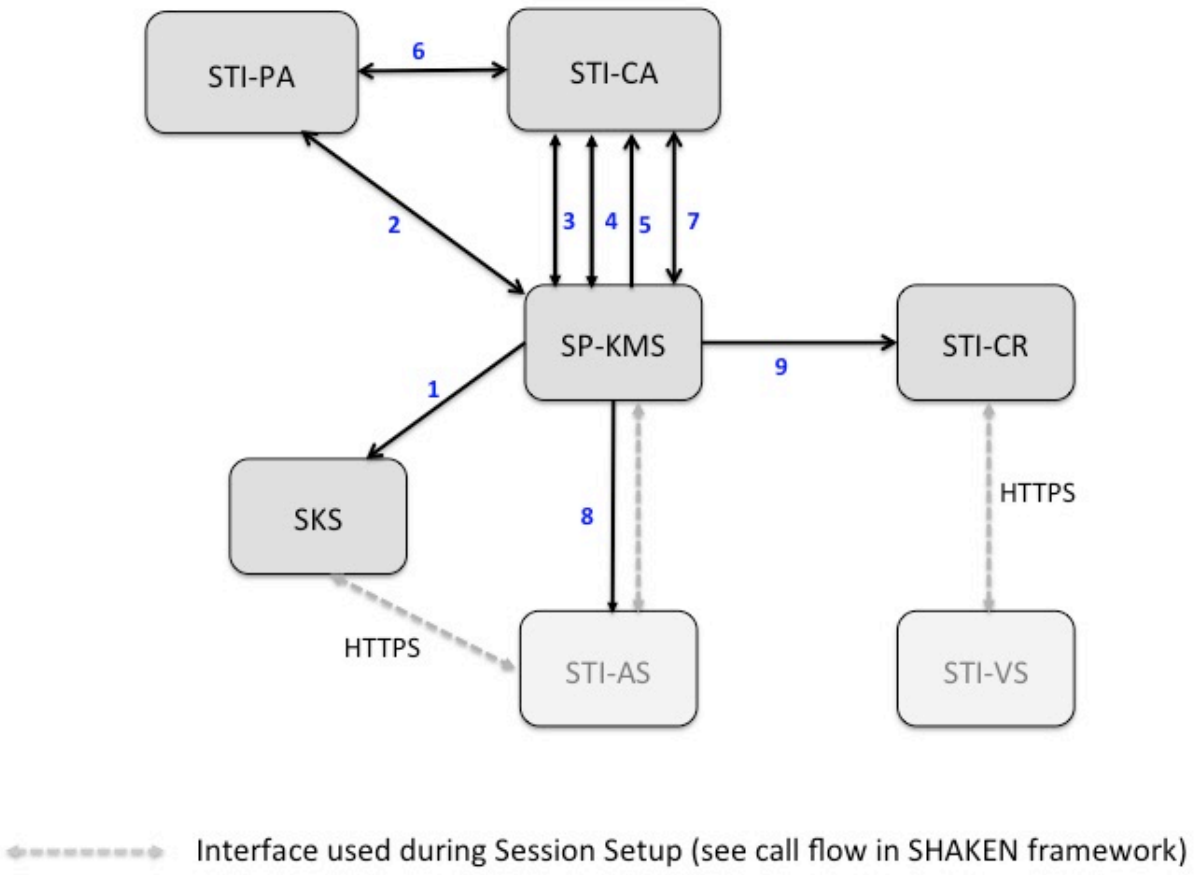

Figure 6.2 – SHAKEN Certificate Management High Level Call Flow

Prior to requesting STI certificates from the STI-CA, the SP-KMS generates a public/private key pair per standard PKI. The private key is used by the STI-AS in signing the PASSporT in the SIP Identity header field. The public key will be included in the public key certificate being requested.

1. The SP-KMS securely distributes the SP STIR private key to its SKS.

The ACME client on the Key Management Server presents a list of STI-CAs from which it could get a certificate. The Service Provider selects the preferred STI-CA and initiates the following steps:

2. The SP generates or chooses a set of public/private key ACME credentials for all transactions with the STI-CA. Assuming a first-time transaction or if the Service Provider Code token is either expired or not cached, the SP-KMS sends a request for a Service Provider Code token to the STI-PA with a fingerprint of the ACME account public key. This Service Provider Code token is used for service provider validation during the process of acquiring a certificate.
3. If it has not already done so, the ACME client on the SP-KMS registers with the STI-CA using the ACME key credentials prior to requesting an STI certificate per the procedures in draft-ietf-acme-acme.
4. Once the ACME client on the SP-KMS has registered with the STI-CA, the ACME client can send a request for a new STI certificate to the ACME server hosted on the STI-CA. The response to that request includes a URL for the authorization challenge.
5. The service provider that is requesting a signed STI certificate responds to that challenge by providing the current valid token acquired from the STI-PA.

6. 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 4, the ACME client starts polling for the "valid" status to determine if the service provider has been authorized to get an STI certificate and whether an STI certificate is available. Once the STI certificate has been issued, the ACME client downloads the STI 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, WEBPUSH, etc.).
9. The SP-KMS puts the public key certificate in the STI-CR.

After initially retrieving the certificate, the ACME client periodically contacts the STI-CA to get updated public key certificates, CRLs, or anything else required to keep the server functional and its credentials up-to-date as described in clause 6.3.10.

6.3.2 STI-PA Account Registration & Service Provider Authorization

The authorization model for SHAKEN assumes there is a single authorized STI-PA chosen by the STI-GA.

As identified in clause 5.2.3, while the criteria by which a Service Provider is authorized to serve in the role is out of scope of this document, an interface to the STI-PA from the SP is required to determine if a specific Service Provider is allowed to assert and digitally sign the Caller ID associated with the originating telephone number of telephone calls initiated on the VoIP telephone network. A verification and validation process shall be followed by the STI-PA to provide a secure set of credentials (e.g., username and password combined with other secure two-factor access security techniques) to allow the SP to access a management portal for the STI-PA set of services.

This management portal will be specified by the STI-PA, but should allow Service Providers to input Service Provider-specific configuration details such as the following:

- Login password management.
- SP-KMS instance(s) configuration.
- API security client id/secret information.
- Preferred STI-CA selection.

The STI-PA shall provide secure API protection for the Service Provider that follows the procedures in [RFC 6749] Section 2.3 client credentials to access its HTTP-based APIs. This includes the use of an STI-PA-defined client id/secret that is used in the HTTP Authorization header of each request from the Service Provider to the STI-PA. This authorization will allow an SP to acquire the Service Provider Code token as described in clause 6.3.5 and determine the preferred STI-CA to use when requesting STI certificates.

6.3.3 STI-CA Account Creation

When a Service Provider selects a particular STI-CA to service STI certificate requests, the Service Provider shall use the ACME account creation process defined in [draft-ietf-acme-acme].

In order to initiate the account creation process, the requesting Service Provider shall create a key pair using the ES256 algorithm. This key pair represents the Service Provider's ACME account credentials.

> NOTE: This account key pair is also used for the STI-PA Service Provider Code Token fingerprint value to tie the ACME account credentials to the validation of the Service Provider Code token by the STI-CA, as detailed in Clause 6.3.4.1.

The Service Provider's ACME account is created with the STI-CA using the following HTTP POST request:

> NOTE: Unless explicitly stated otherwise, the ACME examples in clause 6 are included for illustrative purposes only and not intended to profile the referenced ACME specifications.

```
POST /acme/new-account HTTP/1.1
Host: sti-ca.com
Content-Type: application/jose+json
```

```
{
  "protected": base64url({
    "alg": "ES256",
    "jwk": {...},
    "nonce": "6S8IqOGY7eL2lsGoTZYifg",
    "url": "https://sti-ca.com/acme/new-reg"
  })
  "payload": base64url({
   "contact": [
     "mailto:cert-admin-sp-kms01@sp.com",
      "tel:+12155551212"
    ]
  }),
  "signature": "RZPOnYoPs1PhjszF...-nh6X1qtOFPB519I"
}
```

Per ACME, the requesting Service Provider shall sign this request with the ACME account private key. The public key shall be passed in the JavaScript Object Notation (JSON) Web Key ("jwk" header parameter) [RFC 7515] as a JSON Web Key (JWK) [RFC 7517]. An example JWK is as follows:

```
{
  "kty":"EC",
  "crv":"P-256",
  "x":"f83OJ3D2xF1Bg8vub9tLe1gHMzV76e8Tus9uPHvRVEU",
  "y":"x_FEzRu9m36HLN_tue659LNpXW6pCyStikYjKIWI5a0",
  "kid":"sp.com Reg Public key 123XYZ"
}
```

If the account already exists with the key, then the response shall be 200 OK. Otherwise, if the account creation succeeds and is created at the STI-CA, the response shall be 201 OK in the following form:

```
HTTP/1.1 201 Created
Content-Type: application/json
Replay-Nonce: D8s4D2mLs8Vn-goWuPQeKA
Location: https://sti-ca.com/acme/acct/1
Link: <https://sti-ca.com/acme/some-directory>;rel="index"

{
  "status": "valid",

  "contact": [
   "mailto:cert-admin-sp-kms01@sp.com",
    "tel:+12155551212"
  ]
}
```

In the case where the Service Provider wants to change the account's public/private key pair used for the particular STI-CA, it can use the following request with both the old key and signature, and updated key and signature as follows:

```
POST /acme/key-change HTTP/1.1
Host: sti-ca.com
Content-Type: application/jose+json

{
  "protected": base64url({
    "alg": "ES256",
    "jwk": /* old key */,
```

```
      "nonce": "K60BWPrMQG9SDxBDS_xtSw",
      "url": “https://sti-ca.com/acme/key-change"
    }),
    "payload": base64url({
      "protected": base64url({
        "alg": "ES256",
        "jwk": /* new key */,
        "url": "https://sti-ca.com/acme/key-change"
      }),
      "payload": base64url({
        "account": "https://sti-ca.com/acme/acct/1",
        "newKey": /* new key */
      })
      "signature": "Xe8B94RD30Azj2ea...8BmZIRtcSKPSd8gU"
    }),
    "signature": "5TWiqIYQfIDfALQv...x9C2mg8JGPxl5bI4"
  }
```

6.3.4 Service Provider Code Token Acquisition

Before a Service Provider can create a Certificate Signing Request (CSR) as part of the ACME request to the STI-CA, it shall get a valid and up-to-date Service Provider Code token. The Service Provider Code and Service Provider Code token are used for two things.

First, the Service Provider Code token is used as a way to authenticate the Service Provider to the STI-CA as part of the authorization process defined in ACME and below as part of the application for an STI Certificate in clause 6.3.6.

Second, the Service Provider Code is used as part of the CSR so that the Service Provider Code is included in the STI certificate and can be validated by the STI-VS receiving a call with a signed Identity header field as defined in the SHAKEN Framework [ATIS-1000074].

6.3.4.1 STI-PA Service Provider Code Token Definition

The following is a standard JSON Web Token (JWT) [RFC 7519].

JWT Protected Header

```
{
  "alg": "ES256",
  "typ": "JWT",
  "x5u": "https://sti-pa.com/sti-pa/cert.crt"
}
```

The “alg” value defines the algorithm used in the signature of the token. For Service Provider Code tokens, the algorithm shall be “ES256”.

The “typ” is set to standard “JWT” value.

The “x5u” value defines the URL of the STI certificate of the STI-PA administrator validating the Service Provider Code.

JWT Payload

```
{
  "sub": ["1234"]
  "iat": 14589234802,
  "nbf": 14782347239,
  "exp": 15832948298
  "fingerprint":"SHA256
56:3E:CF:AE:83:CA:4D:15:B0:29:FF:1B:71:D3:BA:B9:19:81:F8:50:9B:DF:4A:D4:39:72:E2:B1
:F0:B9:38:E3"
}
```

The required values for the token are as follows:

- The “sub” value is the Service Provider Code value being validated in the form of an American Standard Code for Information Interchange (ASCII) string. This should be in the form of a JSON array for future extension, however, only a single SPC value is required or will be used for SHAKEN.
- The “iat” value is the DateTime value of the time and date the token was issued.
- The “nbf” value is the DateTime value of the starting time and date that the token is valid.
- The “exp” value is the DateTime value of the ending time and date that the token expires.
- The “fingerprint” value is the certificate fingerprint of the ACME credentials the SP used to create an account with the STI-CA, as defined in clause 6.3.3. This shall be in the form as shown in the above example with the algorithm first followed by a space followed by the fingerprint value. A certificate fingerprint is a secure one-way hash of the Distinguished Encoding Rules (DER) form of the certificate. The fingerprint value consists of the name of the hash function, which shall be ‘SHA256’ for this specification, followed by the hash value itself. The hash value is represented as a sequence of uppercase hexadecimal bytes, separated by colons. The number of bytes is defined by the hash function.

JSON Web Token Signature

The JSON Web token signature follows the standard JSON Web Signature (JWS)-defined signature string.

6.3.4.2 Service Provider Code Token API Request Definition

The following is the HTTP-based POST request that the STI-PA shall provide to a service provider to make the request.

POST /sti-pa/account/:id/token

Description

A request to get a current Service Provider Code token for a Service Provider to use in a CSR to the STI-CA.

Request

Pass the following information in the request parameter.

Filter	Description
id	A unique account id provided to Service Provider

Pass the following information in JSON body.

Property	Type	Description
fingerprint	string	The fingerprint of the public key certificate used for STI-CA ACME account creation

Example JSON body with fingerprint:

```
   {
     "fingerprint":"SHA256
56:3E:CF:AE:83:CA:4D:15:B0:29:FF:1B:71:D3:BA:B9:19:81:F8:50:9B:DF:4A:D4:39:72:E2:B1
:F0:B9:38:E3"
   }
```

Response

200 OK

Filter	Type	Description
token	string	A signed Service Provider Code token using the STI-PA certificate with a TTL of the token set by policy

403 - Forbidden

Authorization header credentials are invalid.

404 - Invalid account ID

Account ID provided does not exist or does not match credentials in Authorization header.

6.3.5 Application for a Certificate

Assuming the Service Provider has a current and up-to-date signed Service Provider Code token, as detailed in the previous clause of this document, it can immediately initiate an application for a new STI certificate to the STI-CA.

This process includes two main steps, creation of the CSR and the ACME-based certificate application process as defined in [draft-ietf-acme-acme].

6.3.5.1 CSR Construction

The general creation of a CSR is defined in [RFC 5280] with a format defined as PKCS #10 and defined in [RFC 2986]. For the SHAKEN certificate framework and ACME-based protocols the overall process and definitions do not change, however there are a few specific uses of and guidelines for CSR attributes defined as part of the SHAKEN Certificate Framework.

Following [draft-ietf-stir-certificates], a Telephone Number (TN) Authorization List certificate extension shall be included in the CSR. In the case of SHAKEN, the TN Authorization List shall include only one Service Provider Code. A service provider can obtain multiple certificates for a given service provider code or for different service provider codes. The essential aspect is that the service provider code uniquely identifies a given service provider.

As defined in [draft-ietf-stir-certificates] the OID defined for the TN Authorization list extension will be defined in Structure of Management Information (SMI) Security for Public Key Infrastructure for X.509 Certificates (PKIX) Certificate Extension registry here: http://www.iana.org/assignments/smi-numbers/smi-numbers.xhtml#smi-numbers-1.3.6.1.5.5.7.1 and assigned the value 26.

6.3.5.2 ACME Based Steps for Application for an STI Certificate

Once a CSR has been generated, the steps in the ACME protocol flow are as follows. It should be noted that it is possible for the ACME client to do a pre-authorization prior to applying for a certificate, in which case processing equivalent to steps 2-5 is done prior to an application for a certificate and thus the polling period for step 7 is abbreviated. However, that is not the recommended approach for the SHAKEN certificate framework at this time.

1) The application is initiated by the ACME client with an HTTP POST as shown in the following 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({
      "csr": "5jNudRx6Ye4HzKEqT5...FS6aKdZeGsysoCo4H9P",
      "notBefore": "2016-01-01T00:00:00Z",
      "notAfter": "2016-01-08T00:00:00Z"
    }),
    "signature": "H6ZXtGjTZyUnPeKn...wEA4TklBdh3e454g"
  }
```

The CSR is inserted into the JWS payload along with the requested time frame of the certificate application. The request is signed using the private key used in the ACME registration with the STI-CA.

2) The STI-CA ACME server shall look into the CSR request as standard process. However, for the SHAKEN certificate management specifically, different from a typical domain validation, it shall use the specific “type” identifier of “TNAuthList” and include a key of “value” which is a Service Provider Code. An example of this identifier is:

```
    "identifier": {
      "type": "TNAuthList",
      "value":["1234"]
    }
```

This identifier will be used in the authorization challenge that will be shown incorporated into the authorization object below.

This service provider code shall correspond to the service provider code provided in the STI-PA token.

3) Upon successful processing of the application request, a challenge authorization response from the ACME server is sent back, as shown in the following example:

```
  HTTP/1.1 201 Created
  Replay-Nonce: MYAuvOpaoIiywTezizk5vw
  Location: https://sti-ca.com/acme/order/1234

  {
    "status": "pending",
```

```
  "expires": "2015-03-01T14:09:00Z",

  "csr": "jcRf4uXra7FGYW5ZMewvV...rhlnznwy8YbpMGqwidEXfE",
  "notBefore": "2016-01-01T00:00:00Z",
  "notAfter": "2016-01-08T00:00:00Z",

  "authorizations": [
    "https://sti-ca.com/acme/authz/1234"
  ]
}
```

4) The SP-KMS ACME client shall respond to the challenge before it expires, but for the SHAKEN framework, the ACME client shall be prepared to respond to the challenge using the current Service Provider Code token retrieved in preparation for the certificate application process.

The ACME client shall first retrieve the authorization challenge details with a HTTP GET, an example of which follows:

```
GET /acme/authz/1234 HTTP/1.1
Host: sti-ca.com

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":[ "1234"]
  },

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

NOTE: This includes the identifier specific to the SHAKEN certificate framework constructed as part of the certificate application request and CSR processing. The response shall also include the SHAKEN specific challenge type of “token”.

5) Using the URL of the challenge, the ACME client shall respond to this challenge with the Service Provider Code token to validate the Service Provider’s authority to request an STI certificate. An HTTP POST shall be sent back in the form as follows:

```
POST /acme/authz/1234/0 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": "Q_s3MWoqT05TrdkM2MTDcw",
    "url": "https://sti-ca.com/acme/authz/1234/0"
  }),
  "payload": base64url({
    "type": "spc-token",
    "keyAuthorization": "IlirfxKKXA...vb29HhjjLPSggwiE"
  }),
  "signature": "9cbg5JO1Gf5YLjjz...SpkUfcdPai9uVYYQ"
}
```

This challenge response JWS payload shall include the SHAKEN certificate framework specific challenge type of “spc-token” and the “keyAuthorization” field containing the “token” for the challenge concatenated with the value of the Service Provider Code token.

6) Once the challenge response is sent to the STI-CA ACME server, the server shall validate the “token” challenge by verifying the Service Provider Code token. As a part of that token validation, the STI-CA needs to retrieve the public key of the STI-PA, as identified in the x5u protected header value in the token. Once successful, the state of the challenge shall be changed from “pending” to “valid”.

7) Finally, the SHAKEN ACME client shall poll the status of the authorization until it verifies that the challenge is set to the “valid” status. This is performed with the following HTTP GET request:

```
GET /acme/authz/1234_HTTP/1.1
Host: sti-ca.com

HTTP/1.1 200 OK

{
  "status": "valid",
  "expires": "2015-03-01T14:09:00Z",

  "identifier": {
   "type": "TNAuthList",
    "value":[ "1234"]
  },

  "challenges": [
    {
      "type": "spc-token",
      "url": "https://sti-ca.com/authz/1234/0",
      "status": "valid",
      "validated": "2014-12-01T12:05:00Z",
      "token": "DGyRejmCefe7v4NfDGDKfA",

    }
  ]
}
```

8) Once the challenge is “valid” the STI-CA ACME server can then proceed with the creation of the STI certificate that was requested in the CSR using standard X.509 processing.

6.3.6 STI Certificate Acquisition

After the authorization process that validates the Service Provider and its ability to request an STI certificate is complete, the SP-KMS ACME client can then retrieve the STI certificate from the STI-CA ACME server. This is performed using an HTTP GET request and response as follows:

```
GET /acme/cert/1234 HTTP/1.1
Host: sti-ca.com
Accept: application/pkix-cert

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


-----BEGIN CERTIFICATE-----
[End-entity certificate contents]
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
[Issuer certificate contents]
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
[Other certificate contents]
-----END CERTIFICATE-----
```

This certificate response will include the “end-entity” STI certificate requested in the CSR. It will also include any of the issuer STI certificates as part of the certificate chain needed for validating intermediate or root certificates appropriate for the STI-CA specific certificate chain.

6.3.7 STI Certificate Management Sequence Diagrams

Figure 6.3 provides the sequence of processing for a service provider to set up an account with the STI-PA and then create an account with the STI-CA using the ACME protocol. Figure 6.4 provides the sequence of processing for the SP-KMS to acquire a certificate using the ACME protocol.

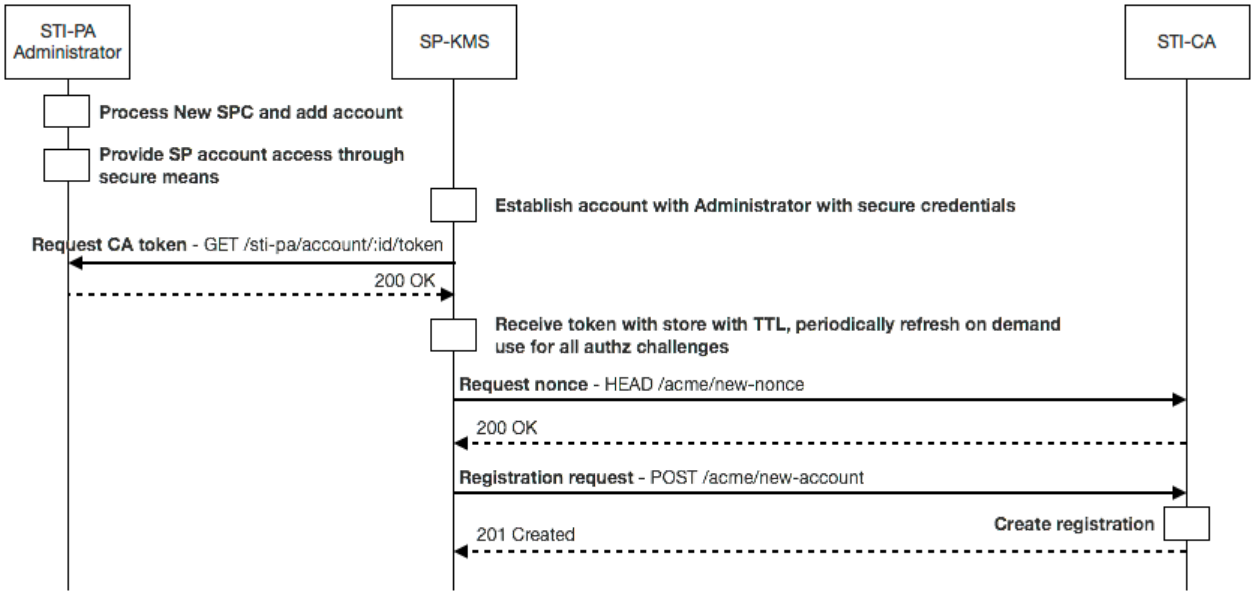

Figure 6.3 – STI-PA Account Setup and STI-CA (ACME) Account Creation

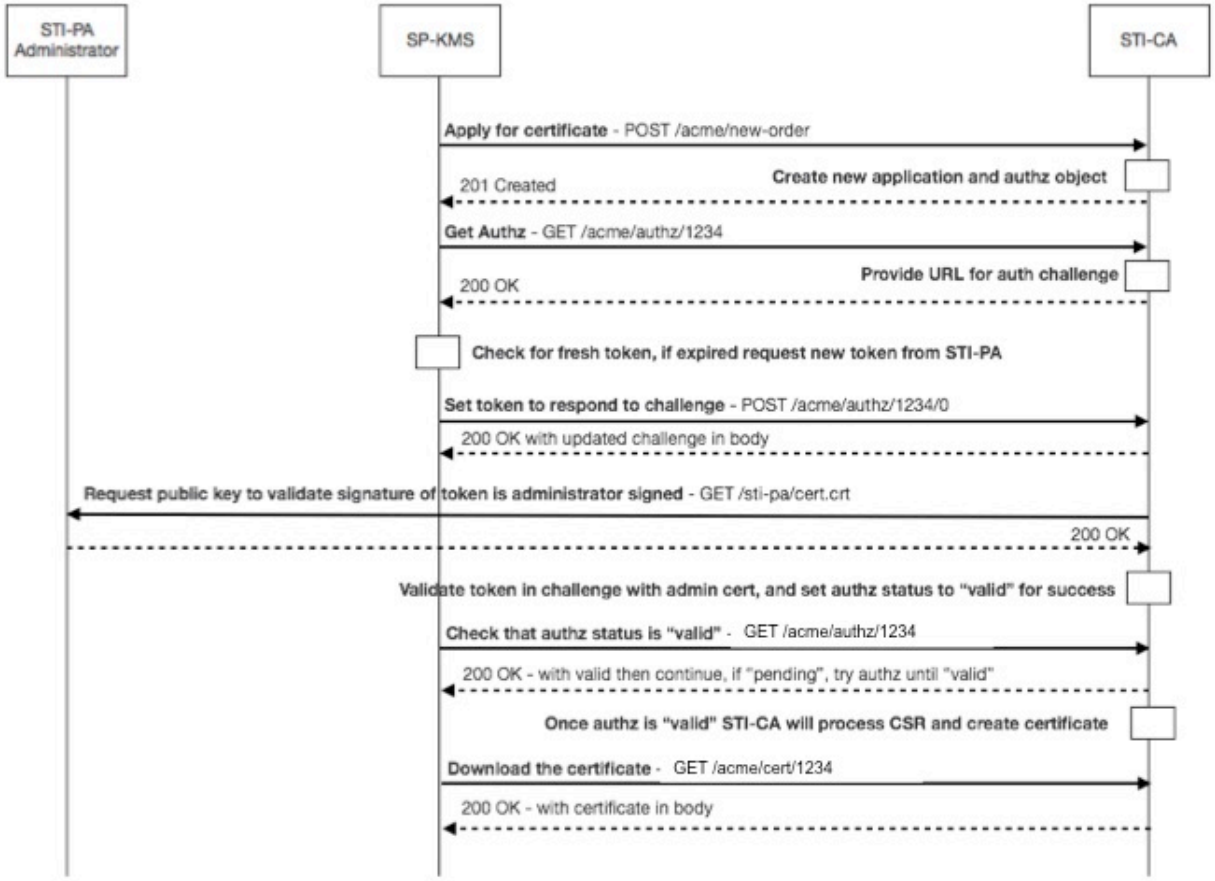

Figure 6.4 – STI Certificate Acquisition

6.3.8 Lifecycle Management of STI certificates

There are a number of lifecycle processes that can happen for each of the three main participants in the SHAKEN Certificate Framework lifecycle.

The STI-PA has a role in the management and upkeep of the verification of Service Providers and the potential need to revoke the STI-PA certificate used to sign the Service Provider Code token.

The STI-CA provides the capability to renew or update STI certificates for Service Providers through standard ACME interface capabilities. STI certificate renewal requests shall use the same authentication procedures that are applied to requests for a new STI certificate as described in clause 6.3.5.

The Service Provider has the ability to manage, renew, and update STI certificates and the ability to renew Service Provider Code tokens as credentials used to obtain STI certificates as part of the SHAKEN certificate framework.

6.3.9 STI Certificate Updates/Rotation Best Practices

Consideration of the impact of switching STI certificates and other certificate management impacts, while there are in-flight calls, should be considered. Standard CRL techniques should be considered the initial preferred way of signaling the revocation of a certificate. Techniques for short-lived certificates could be considered in the future.

6.3.10 Evolution of STI Certificates

SHAKEN proposes starting with Service Provider-level certificates. There are important use cases that may require finer granularity for STI certificates, including the possibility of telephone number level certificates (e.g., for School Districts, Police, government agencies, and financial institutions), where calls should be validated in order to guarantee delivery through the potential use of anti-spoofing mitigation techniques.

Future versions of this document and associated documents may provide the ability to validate telephone numbers and blocks of telephone numbers likely utilizing certificate details and practices defined in [draft-ietf-stir-certificates].

Appendix A – Certificate Creation & Validation with OpenSSL

Steps for Generating STI-CA CSR with OpenSSL

Check OpenSSL version and make sure it is at least 1.0.1e:

```
# openssl version
OpenSSL 1.0.1e-fips 11 Feb 2013
```

Check if 256-bit Elliptic Curve Digital Signature Algorithm (ECDSA) keys are supported, such as prime256v1:

```
# openssl ecparam -list_curves
secp384r1 : NIST/SECG curve over a 384 bit prime field
secp521r1 : NIST/SECG curve over a 521 bit prime field
prime256v1: X9.62/SECG curve over a 256 bit prime field
```

Prepare the configuration file for generating DER encoded value of the TNAuthorizationList extension. For example, for requesting a STI-CA certificate with Service Provider Code “1234”, the following configuration file, TNAuthList.conf, would be generated:

```
# cat > TNAuthList.conf << EOF
asn1=SEQUENCE:tn_auth_list
[tn_auth_list]
field1=EXP:0,IA5:1234
EOF
```

Generate the DER encoded value for the TNAuthorizationList extension; for example, by using the TNAuthList.conf file generated in the previous step. The TNAuthList.der file will be generated:

```
# openssl asn1parse -genconf TNAuthList.conf -out TNAuthList.der
  0:d=0  hl=2 l=   8 cons: SEQUENCE
  2:d=1  hl=2 l=   6 cons: cont [ 0 ]
  4:d=2  hl=2 l=   4 prim: IA5STRING         :1234
```

Construct the OpenSSL configuration file for including the TNAuthorizationList extension (OID 1.3.6.1.5.5.7.1.26) in generating CSR, by using the DER value generated from the previous step:

```
# cat > openssl.conf << EOF
[ req ]
distinguished_name = req_distinguished_name
req_extensions = v3_req
[ req_distinguished_name ]
commonName = "SHAKEN"
[ v3_req ]
EOF
# od -An -t x1 -w TNAuthList.der | sed -e 's/ /:/g' -e
's/^/1.3.6.1.5.5.7.1.26=DER/' >> openssl.conf

# cat openssl.conf
[ req ]
distinguished_name = req_distinguished_name
req_extensions = v3_req
[ req_distinguished_name ]
commonName = "SHAKEN"
[ v3_req ]
1.3.6.1.5.5.7.1.26=DER:30:08:a0:06:16:04:31:32:33:34
```

Generate 256-bit ECDSA key pairs, without explicitly encoding EC parameters for avoiding potential problems of PKI toolkits, such as standard JDK:

```
# openssl ecparam -noout -name prime256v1 -genkey -out private_key.pem -outform
PEM

# openssl ec -in private_key.pem -text
read EC key
Private-Key: (256 bit)
priv:
 15:6b:c5:b8:df:84:d8:e3:83:96:2f:18:db:39:e7:
 fe:8c:f7:10:68:49:01:75:87:90:2e:1f:57:14:3f:
 0a:75
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
writing EC key
-----BEGIN EC PRIVATE KEY-----
MHcCAQEEIBVrxbjfhNjjg5YvGNs55/6M9xBoSQF1h5AuH1cUPwp1oAoGCCqGSM49
AwEHoUQDQgAEd8aw1t/9Hwoj3EAkpOqTytc/nreOx3Br4tIOjnkMWji4pf1SXdtD
vwCxzd/Uz8tpNRPRUprjEP4bUVt0wpacIg==
-----END EC PRIVATE KEY-----
```

Generate the CSR file with a SHA256 signature, by using the openssl.conf file that includes the TNAuthorizationList extension:

```
# openssl req -new -nodes -key private_key.pem -keyform PEM \
 -out csr.pem -outform PEM \
 -subj '/C=US/ST=VA/L=Somewhere/O=AcmeTelecom, Inc./OU=VOIP/CN=SHAKEN' \
 -sha256 -config openssl.conf
```

Verify that the CSR file contains the TNAuthorizationList extension:

```
# openssl req -in csr.pem -text -noout
Certificate Request:
 Data:
 Version: 0 (0x0)
 Subject: C=US, ST=VA, L=Somewhere, O=AcmeTelecom, Inc., OU=VOIP, CN=SHAKEN
 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
 Attributes:
 Requested Extensions:
 1.3.6.1.5.5.7.1.26:
 0.....1234
 Signature Algorithm: ecdsa-with-SHA256
 30:45:02:20:5c:f0:4b:cd:16:a3:e7:66:d8:68:fe:65:e2:7b:
 8f:70:92:e6:4c:25:c9:41:bf:45:d1:e9:20:16:64:04:fc:cf:
 02:21:00:82:7c:24:9a:aa:22:c6:23:9d:6d:04:c2:e7:76:ed:
 44:d1:bc:bd:a2:1b:af:cb:97:71:9d:7b:bf:3a:4e:6a:59
```

Verify that the certificate obtained from a STI-CA contains the TNAuthorizationList extension:

```
# openssl x509 -in cert.pem -text -noout
Certificate:
 Data:
 Version: 3 (0x2)
 Serial Number: 6734468596164949790 (0x5d75a381e96f771e)
 Signature Algorithm: sha256WithRSAEncryption
 Issuer: CN=CallAuthnCA, O=Neustar 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=Somewhere, ST=VA, 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

X509v3 Key Usage: critical
 Digital Signature, Non Repudiation, Key Encipherment
 X509v3 Extended Key Usage: critical
 TLS Web Client Authentication, E-mail Protection
 Signature Algorithm: sha256WithRSAEncryption
 88:6b:1b:7a:7a:69:33:53:34:ca:53:a8:b6:87:7b:ed:ba:6d:
 f3:73:96:91:57:1c:ea:4e:e6:66:c7:fa:d3:6d:79:98:f9:7b:
 00:78:bb:19:fd:51:f5:c2:46:d8:ce:f1:7b:13:e3:e2:72:de:
 6e:e3:9d:37:8c:f9:41:9a:b6:89:82:64:6d:d9:e7:22:e3:4b:
 21:90:ad:ad:82:6f:d2:cc:2f:48:a8:46:da:b7:27:10:72:b8:
 97:9c:2b:8d:8a:67:4a:9e:1c:77:c4:32:8c:6e:a1:37:49:3a:
 d8:9c:9c:23:d8:1c:ce:58:d7:39:10:1f:7d:8c:e1:4f:c0:64:
 ef:b9:80:22:06:7f:59:6c:85:79:d4:86:f9:a1:87:75:0e:76:
 51:7b:c6:bf:7b:6b:c7:43:55:e2:a6:88:0f:f7:d7:37:02:b1:
 54:71:a5:3e:81:fc:68:b7:65:eb:de:89:8f:95:a6:c7:fe:84:
 a9:66:58:eb:a8:b3:70:ec:a0:93:2a:b1:01:5d:95:6e:be:49:
 7e:01:17:fe:5f:d4:55:a9:77:e5:51:67:33:ca:20:97:82:66:
 05:e3:59:60:24:25:93:89:46:90:5f:2f:cc:57:2a:b3:d4:a8:
 c4:5c:2a:23:82:6e:80:c2:cf:23:eb:65:39:4c:16:02:0f:bc:
 a3:17:65:6b
```