ATIS-1000080.v004

ATIS Standard on

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

Alliance for Telecommunications Industry Solutions

Approved October 5, 2021

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

1 Scope & Purpose 1
1.1 Scope 1
1.2 Purpose 1

2 References 1
2.1 Normative References 1

3 Definitions, Acronyms, & Abbreviations 2
3.1 Definitions 2
3.2 Acronyms & Abbreviations 4

4 Overview 6

5 SHAKEN Governance Model 6
5.1 Requirements for Governance of STI Certificate Management 6
5.2 Certificate Governance: Roles & Responsibilities 7
5.2.1 Secure Telephone Identity Policy Administrator (STI-PA) 8
5.2.2 Secure Telephone Identity Certification Authority (STI-CA) 8
5.2.3 Service Provider (SP) 8

6 SHAKEN Certificate Management 9
6.1 Requirements for SHAKEN Certificate Management 9
6.2 SHAKEN Certificate Management Architecture 10
6.3 SHAKEN Certificate Management Process 10
6.3.1 SHAKEN Certificate Management Flow 11
6.3.2 STI-PA Account Registration & Service Provider Authorization 13
6.3.3 STI-CA Account Creation 13
6.3.4 Service Provider Code Token 15
6.3.5 Application for a Certificate 19
6.3.6 STI Certificate Acquisition 25
6.3.7 STI Certificate Management Sequence Diagrams 26
6.3.8 Lifecycle Management of STI Certificates 27
6.3.9 STI Certificate Revocation 27
6.3.10 Evolution of STI Certificates 29
6.4 STI Certificate and Certificate Revocation List (CRL) Profile for SHAKEN 29
6.4.1 STI Certificate Requirements 29
6.4.2 SHAKEN CRL Requirements 30

Appendix A – SHAKEN Certificate Management Example with OpenSSL 32
A.1 TNAuthorizationList extension 32
A.2 Setup directories 33
A.3 Create private key and CSR 33
A.3.1. Create private key 33
A.3.2. Create CSR from private key 33
A.4 Signing certificate using root CA 33
A.4.1. Create file to be used as certificate database by openssl 35
A.4.2. Create file that contains the certificate serial number 35
A.4.3. Create directories to be used to store keys, certificates and signing requests 35
A.4.4. Create root key 35
A.4.5. Create root certificate 36
A.4.6. Verify root certificate 36
A.4.7. Sign CSR with root CA cert and create End-Entity certificate 37
A.4.8. Verify End-Entity certificate 37
A.4.9. Verify chain of trust 38

A.5 Signing certificate using intermediate CA 38
A.5.1. Create file to be used as certificate database by openssl 40
A.5.2. Create file that contains the certificate serial number 40
A.5.3. Create directories to be used to store keys, certificates and signing requests 40
A.5.4. Create intermediate key 40
A.5.5. Create CSR from intermediate key 40
A.5.6. Create intermediate certificate 41
A.5.7. Verify intermediate certificate 41
A.5.8. Sign CSR with intermediate cert and create End-Entity certificate 42
A.5.9. Verify End-Entity certificate 42
A.5.10. Verify chain of trust 43

Table of Figures

Figure 5.1 – Governance Model for Certificate Management 7
Figure 6.1 – SHAKEN Certificate Management Architecture 10
Figure 6.2 – SHAKEN Certificate Management High Level Call Flow 12
Figure 6.3 – STI-PA Account Setup and STI-CA (ACME) Account Creation 26
Figure 6.4 – STI Certificate Acquisition 27
Figure 6.5 – Distribution of the CRL 28
Figure 6.6 – Using the CRL 29

ATIS Standard on –

SHAKEN: Governance Model and Certificate Management

1 Scope & Purpose

1.1 Scope

This document expands the ATIS-1000074, *Signature-based Handling of Asserted Information using Tokens (SHAKEN)* [Ref 1], 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 [Ref 1]. 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 Engineering Task Force (IETF) RFC 5280, *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile* [Ref 11]. Per the SHAKEN framework, the certificates themselves are based on X.509 with specific policy extensions based on RFC 8226, *Secure Telephone Identity Credentials: Certificates* [Ref 20]. 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 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.

2.1 Normative References

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

[Ref 2] ATIS-1000084, *Technical Report on Operational and Management Considerations for SHAKEN STI Certification Authorities and Policy Administrators.*[1]

[Ref 3] ATIS-0300251, *Codes for Identification of Service Providers for Information Exchange.*[1]

[Ref 4] ATIS-1000054*, ATIS Technical Report on Next Generation Network Certificate Management.*[1]

[1] This document is available from the Alliance for Telecommunications Industry Solutions (ATIS) at: < https://www.atis.org >.

[Ref 5] draft-ietf-acme-authority-token-tnauthlist, *TNAuthList profile of ACME Authority Token.*[2]

[Ref 6] RFC 2986, *PKCS #10: Certification Request Syntax Specification Version 1.7.*[2]

[Ref 7] RFC 3261, *SIP: Session Initiation Protocol.*[2]

[Ref 8] RFC 3647, *Internet X.509 Public Key Infrastructure: Certificate Policy and Certification Practices Framework.*[2]

[Ref 9] RFC 4949, *Internet Security Glossary, Version 2.*[2]

[Ref 10] RFC 5246, *The Transport Layer Security (TLS) Protocol Version 1.2.*[2]

[Ref 11] RFC 5280, *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile.*[2]

[Ref 12] RFC 6749, *The OAuth 2.0 Authorization Framework.*[2]

[Ref 13] RFC 7231, *Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content.*[2]

[Ref 14] RFC 7468, *Textual Encodings of PKIX, PKCS, and CMS Structures.*[2]

[Ref 15] RFC 7515, *JSON Web Signatures (JWS).*[2]

[Ref 16] RFC 7517, *JSON Web Key (JWK).*[2]

[Ref 17] RFC 7519, *JSON Web Token (JWT).*[2]

[Ref 18] RFC 8224, *Authenticated Identity Management in the Session Initiation Protocol (SIP)*.[2]

[Ref 19] RFC 8225, *PASSporT: Personal Assertion Token.* [2]

[Ref 20] RFC 8226, *Secure Telephone Identity Credentials: Certificates.*[2]

[Ref 21] RFC 8555, *Automatic Certificate Management Environment (ACME).*[2]

[Ref 22] RFC 8588, *Personal Assertion Token (PASSporT) Extension for Signature-based Handling of Asserted information using toKENs (SHAKEN)*.[2]

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, *Internet Security Glossary, Version 2* [Ref 9], 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, *SIP: Session Initiation Protocol* [Ref 7]) 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 [Ref 9]. 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 [Ref 9].

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

Certificate Validation: An act or process by which a certificate user established that the assertions made by a certificate can be trusted [Ref 9].

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 [Ref 9].

Chain of Trust: Deprecated term referring to the chain of certificates to a Trust Anchor. Synonym for Certification Path or Certificate Chain [Ref 9].

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. [Ref 9].

Certificate Policy (CP): A named set of rules that indicates the applicability of a certificate to a particular community and/or class of application with common security requirements (RFC 3647, *Internet X.509 Public Key Infrastructure: Certificate Policy and Certification Practices Framework* [Ref 8]).

Certification Practice Statement (CPS): A statement of the practices that a certification authority employs in issuing, managing, revoking, and renewing or re-keying certificates [Ref 8].

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, *Codes for Identification of Service Providers for Information Exchange* [Ref 3]).

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 [Ref 9].

Identity: Unless otherwise qualified (see, for example, Telephone Identity below), an identifier that unambiguously distinguishes an entity for authentication and other security and policy application purposes. In this report, a Service Provider Code is an example of the identity of one kind of participant in the certificate management process.

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 United States).

POST-as-GET: An HTTP POST Request containing a JWS body as defined by RFC 8555, *Automatic Certificate Management Environment (ACME)* [Ref 21], where the payload of the JWS is a zero-length octet string.

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 [Ref 9].

Public Key: The publicly disclosable component of a pair of cryptographic keys used for asymmetric cryptography [Ref 9].

Public Key Infrastructure (PKI): The set of hardware, software, personnel, policy, and procedures used by a CA to issue and manage certificates [Ref 9].

Root CA: A CA that is directly trusted by an End-Entity. See also Trust Anchor CA and Trusted CA [Ref 9].

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 United States and Canada this would be a Company Code as defined in ATIS-0300251 [Ref 3].

Service Provider Code (SPC) Token: An authority token that can be used by a SHAKEN Service Provider during the STI Certificate ordering process to demonstrate to the STI-CA that the requesting SP has authority over the identity information contained in the TN Authorization List extension of the requested STI Certificate. The SPC Token complies with the structure of the TNAuthList Authority Token defined by draft-ietf-acme-authority-token-tnauthlist, *TNAuthList profile of ACME Authority Token* [Ref 5], but with the restriction for SHAKEN where the TNAuthList value contained in the token's value in the "atc" claim identifies a single Service Provider Code.

Signature: Created by signing the message using the private key. It ensures the identity of the sender and the integrity of the data [Ref 9].

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 [Ref 9].

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 [Ref 9].

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 [Ref 9].

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
CP	Certificate Policy
CPS	Certification Practice Statement
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
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 [Ref 1]. 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, RFC 8225, *Personal Assertion Token (PASSporT)* [Ref 19], RFC 8224, *Authenticated Identity Management in the Session Initiation Protocol* [Ref 18], and RFC 8226 [Ref 20], to support management of STI Certificates within the SHAKEN framework.

The SHAKEN framework uses X.509 certificates, as defined in IETF RFC 5280 [Ref 11], to verify the digital signatures associated with SIP identifiers. Specifically, SHAKEN uses STI Certificates that support the TN Authorization List extension defined in RFC 8226 [Ref 20].

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.

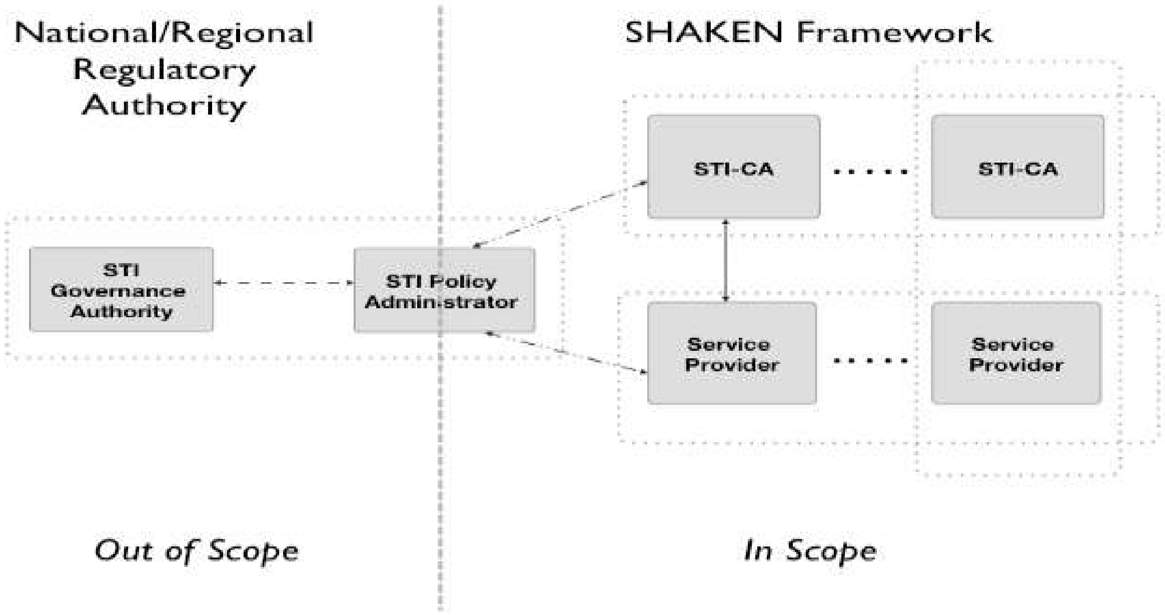

Figure 5.1 – Governance Model for Certificate Management

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 as specified in Clause 7 of ATIS-1000084, *Technical Report on Operational and Management Considerations for SHAKEN STI Certification Authorities and Policy Administrators* [Ref 2]. 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.

The STI-PA also issues Service Provider Code (SPC) Tokens to SHAKEN Service Providers. The STI-PA maintains a distinct X.509 based PKI for digitally signing these SPC Tokens. The SP uses the SPC Token during the recommended ACME certificate ordering process to demonstrate to the issuing STI-CA that the SP has authority over the scope of the requested STI Certificate. The mechanism by which the SP acquires the SPC Token from the STI-PA is described in Clause 6.3.4.2, while the structure of the SPC Token is described in Clause 6.3.4.1.

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 SPC Tokens in the United States would be associated with an 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 itself as the signing entity and protecting the integrity of the SIP Identity header field. 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 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) [Ref 19] in a SIP message attests to a Service Provider’s knowledge of a specific telephone identity that the terminating SP can use to determine specific handling for a call. Details for the attestation are provided in ATIS-1000074 [Ref 1].

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 6.3.4.2. When an SP applies to the STI-CA for issuance of a new STI Certificate, the SP proves to the STI-CA 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, 4 and 5, provide the details of the SP validation mechanism.

6 SHAKEN Certificate Management

Management of certificates for Transport Layer Security (TLS) (RFC 5246, *The Transport Layer Security (TLS) Protocol Version 1.2* [Ref 10]) and HTTPS (RFC 7231, *Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content* [Ref 13]) 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*, ATIS Technical Report on Next Generation Network Certificate Management* [Ref 4]) 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 creating an account 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. SHAKEN 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.

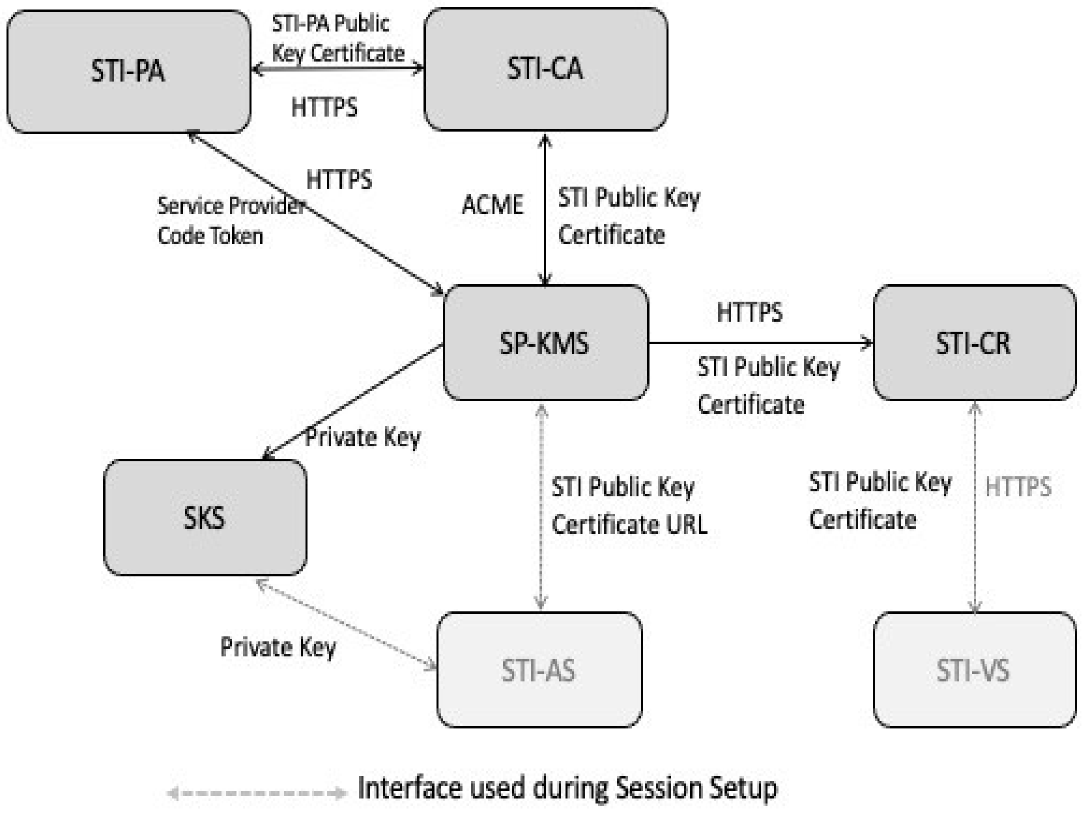
STI-PA
STI-PA Public Key Certificate
HTTPS
STI-CA
HTTPS
Service Provider Code Token
ACME
STI Public Key Certificate
SP-KMS
HTTPS
STI Public Key Certificate
STI-CR
Private Key
SKS
STI Public Key Certificate URL
STI Public Key Certificate
HTTPS
Private Key
STI-AS
STI-VS
Interface used during Session Setup

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 and receives an SPC Token from the STI-PA, 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 STI-AS.
- Secure Telephone Identity Certificate Repository (STI-CR) – The HTTPS server that hosts the public key certificates used by the terminating service provider's STI-VS to validate signatures.

6.3 SHAKEN Certificate Management Process

This clause describes the detailed process for acquiring a signed public key certificate. It is described with 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 as 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 (RFC 6749, *The OAuth 2.0 Authorization Framework* [Ref 12]) HTTPS interface with the Service Provider in order to provide an SPC Token the Service Provider can use for authorization by the STI-CA when requesting an STI 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 RFC 8555 [Ref 21] 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, access 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 [Ref 10] (i.e., HTTPS). HTTP redirects shall not be allowed. Additional considerations on the use of HTTPS for ACME are provided in section 6.1 of RFC 8555 [Ref 21]. 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:

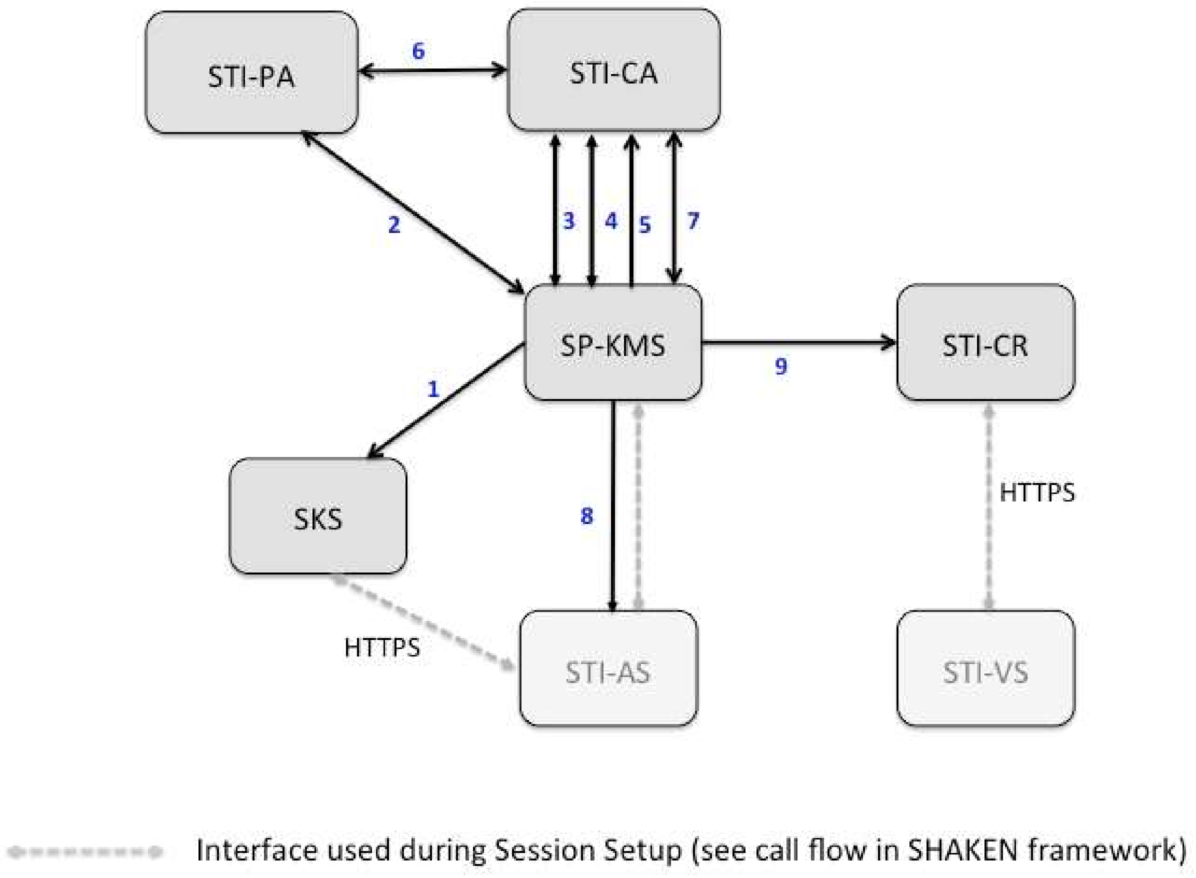
STI-PA
STI-CA
6
2
3
4
5
7
SP-KMS
STI-CR
9
1
SKS
8
HTTPS
HTTPS
STI-AS
STI-VS
Interface used during Session Setup (see call flow in SHAKEN framework)

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 private key to its SKS.

The ACME client on the Key Management Server presents a list of STI-CAs from which it could get an STI 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 an STI Certificate.
3. If it has not already done so, the ACME client on the SP-KMS registers with the STI-CA by creating an ACME account using the ACME key credentials from step 2, prior to requesting an STI Certificate per the procedures in RFC 8555 [Ref 21].

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 SPC Token acquired from the STI-PA.
6. If not previously retrieved, the STI-CA sends a request for the STI PA's public key certificate in order to validate that the signature of the SPC Token has been signed by the STI-PA. Once the STI-CA has verified that the SPC Token is valid, it can issue the STI 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. Upon successful authorization, additional steps are taken to complete the certificate acquisition process per Clause 6.3.5.2. Once the ACME client receives the status indicating 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 STI Certificate, the ACME client periodically contacts the STI-CA to get updated public key certificates to keep the server functional and its credentials up-to-date as described in Clause 6.3.8.

6.3.2 STI-PA Account Registration & Service Provider Authorization

The authorization model for SHAKEN assumes there is at least one authorized STI-PA chosen by the STI-GA.

As identified in Clause 5.2.3, while the criteria by which a Service Provider is eligible 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 calls initiated on the VoIP 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.

The STI-PA shall provide secure API protection for the Service Provider that follows the procedures in RFC 6749 [Ref 12] Section 2.3 on 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 6.3.4.2.

6.3.3 STI-CA Account Creation

Before ACME account creation, the SP-KMS ACME client shall be configured with an ACME directory object URL for each of the SP's preferred STI-CAs. The ACME client can use the directory object URL of the selected STI-CA to discover the URLs of the ACME server resources that the ACME client will use to create and manage its ACME accounts, and to obtain STI Certificates.

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 RFC 8555 [Ref 21].

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: The public key of 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-account"
 })
 "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) defined in RFC 7515, *JSON Web Signatures (JWS)* [Ref 15], as a JSON Web Key (JWK) defined in RFC 7517, *JSON Web Key (JWK)* [Ref 16]. 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"
 ]

 "orders": "https://sti-ca.com/acme/acct/1/orders"

}
```

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

Before a Service Provider can apply for issuance of an STI Certificate from the STI-CA, it shall get a valid and up-to-date SPC Token from the STI-PA.

6.3.4.1 SPC Token Definition

An SP uses an SPC Token during the STI Certificate ordering process to demonstrate to the issuing STI-CA that the SP has control over the scope of the requested certificate. The scope of an STI Certificate is determined by the SPC and TN identity information contained in the TN Authorization List extension defined in RFC 8226 [Ref 20]. SHAKEN shall restrict the scope of STI Certificates to a single Service Provider Code assigned to the SP holding the certificate. Therefore, the scope of an SPC Token shall identify the single SPC value of the certificate it authorizes.

An SPC Token shall comply with the TNAuthList Authority Token structure defined in draft-ietf-acme-authority-token-tnauthlist [Ref 5] per the following example:

JWT Protected Header

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

The “alg” value defines the algorithm used in the signature of the SPC 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-PA certificate that contains the public key corresponding to the private key that was used to sign the SPC Token.

JWT Payload

```
{
  "exp":1300819380,
  "jti":"id6098364921",
  "atc":{"tktype":"TNAuthList",
   "tkvalue":"F83n2a...avn27DN3==",
   "ca":false,
   "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 “exp” claim contains the DateTime value of the ending date and time that the SPC Token expires. The time value is expressed in the NumericDate format in units of seconds, as defined in RFC 7519, *JSON Web Token (JWT)* [Ref 17].
- The “jti” claim contains a universally unique identifier for this TNAuthlist Authority Token transaction.
- The “atc” claim is comprised of four elements, as defined in draft-ietf-acme-authority-token-tnauthlist [Ref 5]. In the context of SHAKEN, the contents of the elements are as follows:
  - The "tktype" key shall contain the string value "TNAuthList".
  - The "tkvalue" key shall be equal to the TNAuthList identifier “value” string, which shall contain the base 64 encoding of the TN Authorization List extension ASN.1 object with explicit tagging, as defined in RFC 8226 [Ref 20]. This object shall contain a single SPC assigned to the requesting Service Provider.
  - The “ca” key shall be set to false, indicating that the SPC Token is being used to authorize the request for an End-Entity certificate.
  - The “fingerprint” key shall be equal to the fingerprint of the ACME account credentials. 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 SPC Token Request API

The following is the HTTPS-based POST request that the STI-PA shall provide to a service provider to make the request for an SPC Token. A Service Provider can obtain multiple active SPC Tokens for the same SPC value, or for different SPC values. As a convenience, the STI-PA shall also include the URL to the Certificate Revocation List (Clause 6.3.9) in the response, since it is also required when the service provider applies for an End-Entity certificate.

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

Description

A request to get a current Service Provider Code Token from the STI-PA that a Service Provider can use during the ACME certificate ordering process to demonstrate to the issuing STI-CA that the SP has authority over the identity information contained in the TN Authorization List of the requested STI Certificate.

Request

The following information is included in the request parameter.

Filter	Description
id	A unique account id provided to Service Provider

And the following information is included in the JSON body of the request.

Property	Type	Description
atc	JSON Object	The “atc” object as defined in Clause 6.3.4.1.

Response

A 200 OK response shall be sent in the case that an SPC Token has been allocated and in the case of specific errors that do not directly map to HTTPS error responses.

200 OK Response

Field	Type	Description
status	string	The status of the request. Initial values are: “success” and “error”
message	string	Text to indicate success or describe the exception encountered. Initial values for error codes are defined in the table below. In the case of a successful transaction, the message claim is set to “SPC Token Granted”
token	string	A signed Service Provider Code Token, using the STI-PA certificate with a TTL of the token set by policy
crl	string	A URL to the Certificate Revocation List maintained by the STI-PA

iss	string	An optional field that identifies the DN of the issuer of the CRL. This field can be omitted if the STI-PA provides an alternate mechanism for conveying the DN of the issuer of the CRL to SPs.
errorCode	integer	An optional field included in the response in the case of a status value of "error".

In the case of a status of "error" in the "status" field in the 200 OK response, the message and errorCode claims shall include one of the following:

message Value	**Description**	**errorCode Value**
Invalid ATC	The "atc" claim is not properly formatted or has invalid content (e.g., "ca" claim shall be false for SHAKEN).	701
Invalid SPC	SPC value in the "tkvalue" element of the "atc" claim does not match the SPC values associated with the account.	702
Missing ATC	The request did not contain an "atc" claim.	703

If there is an error, the "token" field shall be set to "null".

HTTP Error Responses

In the case of an error, an appropriate HTTP response code, as defined in RFC 7231 [Ref 13] shall be returned. The following provides two examples of possible HTTP error responses with semantics specific to the SPC Token request:

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.

6.3.4.3 SPC Token Request Example

This section provides an example showing how an SP would use the SPC Token API to obtain a fresh SPC Token.

First, the SP sends a POST request to the STI-PA with a body containing an "atc" element as defined in draft-ietf-authority-token-tnauthlist [Ref 5]. In this case, the "atc" element identifies a single SPC value.

```
POST /sti-pa/account/3141/token HTTP/1.1
 Host: sti-pa.com
 Content-Type: application/json


{
 "atc":{"tktype":"TNAuthList",
  "tkvalue":"F83n2a...avn27DN3==",
  "ca":false,
    "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"}
  }
```

Once it has determined that the SP is authorized to use the requested the SPC value, the STI-PA responds with the SPC Token, plus the CRL URL, and status information about the request, as follows:

```
HTTP/1.1 200 OK
 Content-Type:application/json
 {
  "status":"success",
  "message":"SPC Token Granted",
  "token":"DGyRejmCefe7v4N...vb29HhjjLPSggwiE",
  "crl":"https://sti-pa.com/sti-pa/crl",
  "iss":"C=US, ST=NJ, L=Bridgewater, O=Example PA, CN=SHAKEN CRL"
 }
```

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 RFC 8555 [Ref 21].

6.3.5.1 CSR Construction

The general creation of a CSR is defined in RFC 5280 [Ref 11] with a format defined as PKCS #10 and defined in RFC 2986, *PKCS #10: Certification Request Syntax Specification Version 1.7* [Ref 6]. 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. The following summarizes the attributes that are described in further detail in this document:

- Following RFC 8226 [Ref 20], 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 STI 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. The Service Provider Code shall be the same SPC as that included in the "tkvalue" in the SPC Token (Clause 6.3.4) included in the ACME challenge response.
- As defined in RFC 8226 [Ref 20], the Object Identifier (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.
- The SP shall include a CRL Distribution Points extension in the CSR, populated as follows:
    - o The distributionPoint field shall contain the HTTP URL reference to the CRL (Clause 6.3.9) obtained from the "crl" field of the SPC Token response received from the STI-PA, as shown in Clause 6.3.4.3.

- The cRLIssuer field shall contain the Distinguished Name of the issuer of the CRL obtained either from the "iss" field of the SPC Token response (if this optional field is present in the response), or via an alternate mechanism outside the scope of this document.

A comprehensive description of the other required attributes in the CSR is provided in Clause 6.4.1.

6.3.5.2 ACME Based Steps for Application for an STI Certificate

Once the ACME account has been created, 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 an STI Certificate, in which case processing equivalent to steps 3-6 is done prior to an application for an STI 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({
  "identifiers": [{"type:"TNAuthList","value":"F83n2a...avn27DN3=="}],
  "notBefore": "2016-01-01T00:00:00Z",
  "notAfter": "2016-01-08T00:00:00Z"
 }),
 "signature": "H6ZXtGjTZyUnPeKn...wEA4TklBdh3e454g"
}
```

The TNAuthList identifier is inserted into the JWS payload along with the requested time frame of the STI Certificate application. The TNAuthList identifier, as defined in draft-ietf-acme-authority-token-tnauthlist [Ref 5], consists of a type field set to "TNAuthList", and a value field containing the base64 encoding of the TN Authorization List ASN.1 object defined in RFC 8226 [Ref 20]. The request is signed using the private key that was used during the STI-CA account creation procedure (Clause 6.3.3).

2) Upon successful processing of the application request, the STI-CA sends a 201 (Created) response containing the newly created order object, 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",

 "notBefore": "2016-01-01T00:00:00Z",
 "notAfter": "2016-01-08T00:00:00Z",
 "identifiers": [{"type:"TNAuthList","value":"F83n2a...avn27DN3=="}],

 "authorizations": [
  "https://sti-ca.com/acme/authz/1234"
 ],
 "finalize": "https://sti-ca.com/acme/order/1234/finalize"

}
```

The order object has a status of "pending" indicating that the order authorizations have not yet been satisfied. The "authorizations" field URL references the authorization object containing the challenges the ACME client shall satisfy in order to demonstrate authority over the TNAuthList identifier listed in the "identifiers" field. The "finalize" field contains the URL that the ACME client will use to finalize the order once the outstanding authorizations have been satisfied.

3) The ACME client shall retrieve the authorization challenge details by sending a POST-as-GET request to the order object "authorizations" URL, an example of which follows:

```
POST /acme/authz/1234 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": "uQpSjlRb4vQVCjVYAyyUWg",
  "url": "https://sti-ca.com/acme/authz/1234",
 }),
 "payload": "",
 "signature": "nuSDISbWG8mMgE7H...QyVUL68yzf3Zawps"
}
```

4) The STI-CA shall respond to the POST-as-GET with a 200 OK response containing an authorization object. The authorization object identifies the challenges that the ACME client must respond to in order to demonstrate authority over the TNAuthList identifier requested in step 1. In the case of SHAKEN, the STI-CA shall return a challenge "type" of "tkauth-01" and a "tkauth-type" of "atc", as specified in draft-ietf-acme-authority-token-tnauthlist [Ref 5]. The authorization object has a "status" of "pending", indicating that there are outstanding challenges that have not been satisfied.

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

{
 "status": "pending",

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

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

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 whose scope is indicated by the Service Provider Code value contained in the TNAuthList identifier from step 1. 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({
  "atc": "evaGxfADs...62jcerQ"
 }),
 "signature": "9cbg5JO1Gf5YLjjz...SpkUfcdPai9uVYYQ"
}
```

This challenge response JWS payload shall include an "atc" field containing the SPC Token described in Clause 6.3.4.1.

6) On receiving the challenge response from the ACME client, the STI-CA ACME server shall transition the challenge object "status" field to the "processing" state while it verifies the received Service Provider Code Token. As a part of that SPC 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 JWT. Once the SPC Token has been verified, the "status" of both the challenge and authorization objects shall be changed to "valid", and the "status" of the order object shall be changed to "ready".

7) While the challenge response is being verified by the STI-CA in step 6, the SHAKEN ACME client shall poll the status of the authorization object, waiting for the "status" to transition to the "valid" state. This is performed with the following POST-as-GET request:

```
POST /acme/authz/1234 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": "uQpSjlRb4vQVCjVYAyyUWg",
  "url": "https://sti-ca.com/acme/authz/1234"
}),
"payload": "",
 "signature": "nuSDISbWG8mMgE7H...QyVUL68yzf3Zawps"
}
```

8) The STI-CA responds to the POST-as-GET request with a 200 OK response containing the authorization object. Once the challenge response has been verified, the STI-CA shall update the status of the authorization object to “valid”. The STI-CA responds to the next POST-as-GET request from the ACME client as follows:

```
  HTTP/1.1 200 OK

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

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

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

As an alternative (or in addition) to polling the authorization object, the ACME client may poll the order object with a POST-as-GET request, waiting for the “status” to transition to the “ready” state.

9) Once the challenge is “valid”, and the order object has transitioned to the “ready” state, the ACME client shall finalize the order by sending an HTTP POST request to the order object “finalize” URL that was returned by the ACME server in step 2. The body of the POST request shall contain the CSR described in Clause 6.3.5.1, as follows:

```
POST /acme/order/asdf/finalize 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": "MSF2j2nawWHPxxkE3ZJtKQ",
    "url": "https://sti-ca.com/acme/order/asdf/finalize"
   }),
   "payload": base64url({
    "csr": "MIIBPTCBxAIBADBFMQ...FS6aKdZeGsysoCo4H9P",
   }),
   "signature": "uOrUfIIk5RyQ...nw62Ay1cl6AB"
  }
```

10) On receiving the request to finalize the order, the STI-CA shall update the order object status to “processing” while finalizing the order, and respond with a 200 OK response containing the order object, as follows:

```
HTTP/1.1 200 OK
Replay-Nonce: CGf81JWBsq8QyIgPCi9Q9X
Location: https://sti-ca.com/acme/order/asdf

{
 "status": "processing",
 "expires": "2015-12-31T00:17:00.00-09:00",

 "notBefore": "2015-12-31T00:17:00.00-09:00",
 "notAfter": "2015-12-31T00:17:00.00-09:00",

 "identifiers": [{"type:"TNAuthList","value":"F83n2a...avn27DN3=="}],

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

 "finalize": "https://sti-ca.com/acme/order/asdf/finalize",

}
```

11) While the order is being finalized, the ACME client shall poll the order object with a POST-as-GET request, waiting for the “status” to transition from “processing” to the “valid” state.

```
POST /acme/order/1234 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": "uQpSjlRb4vQVCjVYAyyUWg",
  "url": "https://sti-ca.com/acme/order/1234",
}),
"payload": "",
 "signature": "nuSDISbWG8mMgE7H...QyVUL68yzf3Zawps"
}
```

12) Once the order has been finalized and the STI Certificate is available, the STI-CA shall update the order object status from “processing” to “valid”. The STI-CA responds to the next POST-as-GET poll request from the ACME client as follows:

```
HTTP/1.1 200 OK
Replay-Nonce: CGf81JWBsq8QyIgPCi9Q9X
Location: https://sti-ca.com/acme/order/asdf

{
 "status": "valid",
 "expires": "2015-12-31T00:17:00.00-09:00",

 "notBefore": "2015-12-31T00:17:00.00-09:00",
 "notAfter": "2015-12-31T00:17:00.00-09:00",
 "identifiers": [{"type:"TNAuthList","value":"F83n2a...avn27DN3=="}],

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

 "finalize": "https://sti-ca.com/acme/order/asdf/finalize",

 "certificate": "https://sti-ca.com/acme/cert/mAt3xBGaobw"
}
```

The “certificate” field contains the URL to the STI Certificate that has been issued in response to this order.

6.3.6 STI Certificate Acquisition

Once the authorization process that validates the Service Provider and its ability to request an STI Certificate is complete, and the STI-CA has issued the certificate, the SP-KMS ACME client can retrieve the STI PEM certificate chain from the STI-CA ACME server using the URL in the “certificate” field of the order object. This is performed using a POST-as-GET request and response as follows:

```
  POST /acme/cert/mAt3xBGaobw HTTP/1.1
  Host: sti-ca.com
  Accept: application/pem-certificate-chain
  Content-Type: application/jose+json

  {
   "protected": base64url({
    "alg": "ES256",
    "kid": " https://sti-ca.com/acme/acct/1",
    "nonce": "uQpSjlRb4vQVCjVYAyyUWg",
    "url": "https://sti-ca.com/acme/cert/mAt3xBGaobw",
  }),
  "payload": "",
  "signature": "nuSDISbWG8mMgE7H...QyVUL68yzf3Zawps"
  }

  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 STI Certificate requested in the CSR. It will also include the complete certificate chain. The certificates are encoded with the PEM textual encoding according to RFC 7468, *Textual Encodings of PKIX, PKCS, and CMS Structures* [Ref 14].

The SP-KMS shall store the certificate in the STI-CR and make the URL available to the STI-AS.

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 an STI Certificate using the ACME protocol.

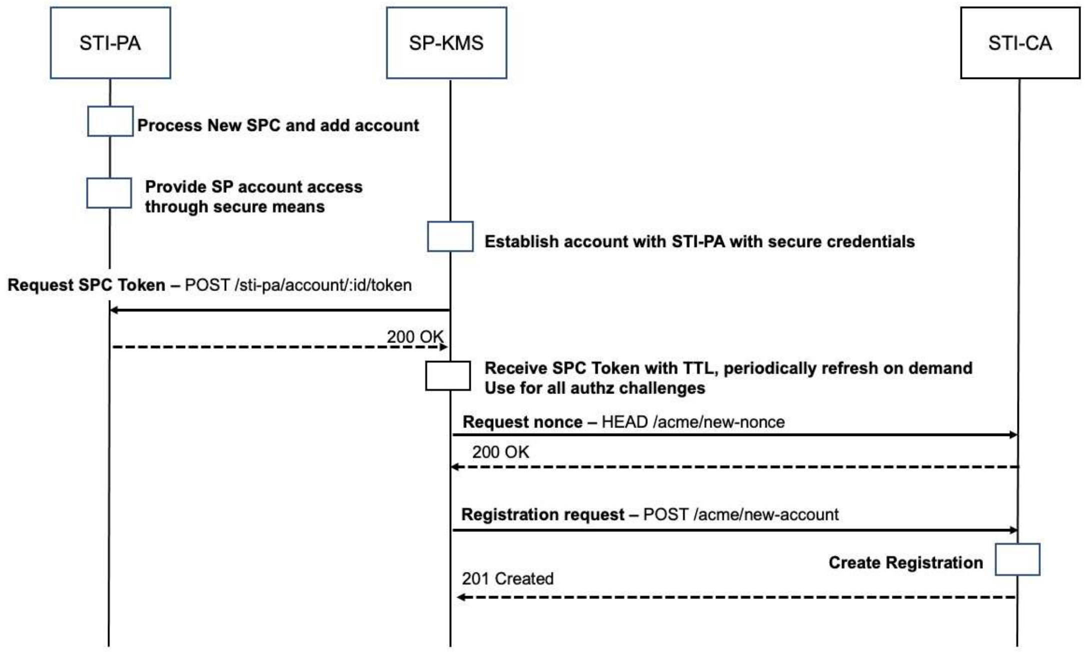
STI-PA
SP-KMS
STI-CA
Process New SPC and add account
Provide SP account access through secure means
Establish account with STI-PA with secure credentials
Request SPC Token – POST /sti-pa/account/:id/token
200 OK
Receive SPC Token with TTL, periodically refresh on demand
Use for all authz challenges
Request nonce – HEAD /acme/new-nonce
200 OK
Registration request – POST /acme/new-account
Create Registration
201 Created

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

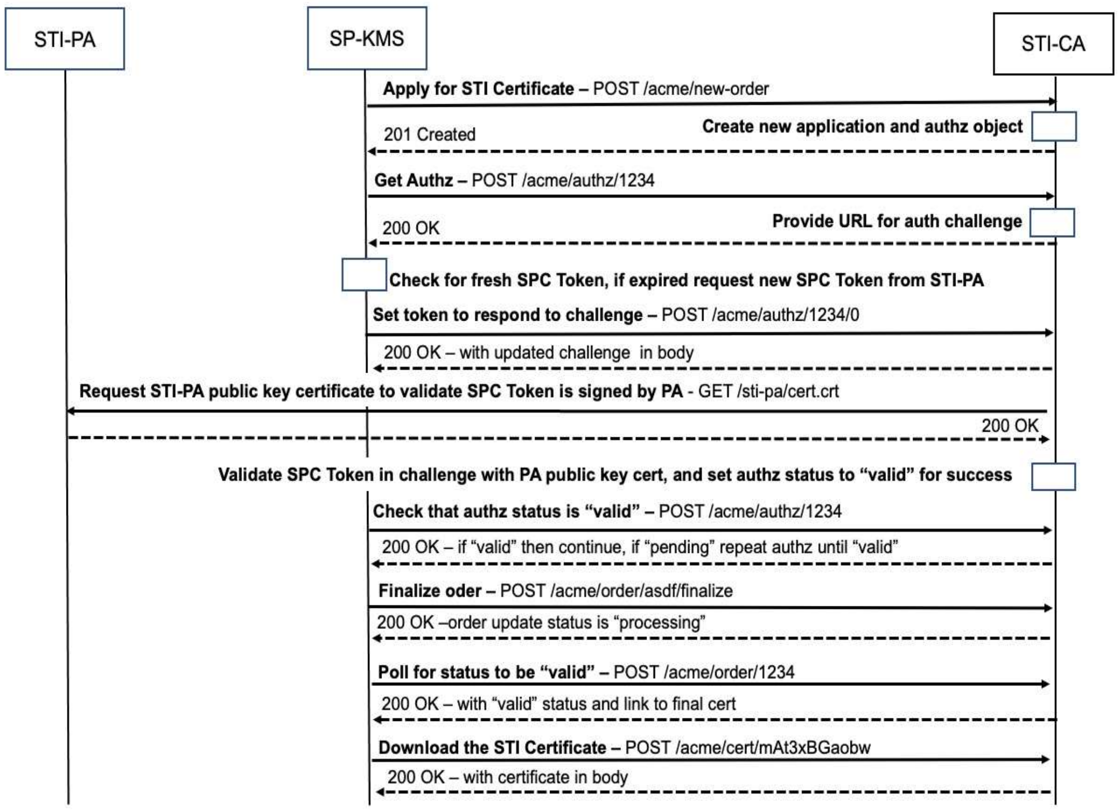
STI-PA
SP-KMS
STI-CA
Apply for STI Certificate – POST /acme/new-order
201 Created
Create new application and authz object
Get Authz – POST /acme/authz/1234
200 OK
Provide URL for auth challenge
Check for fresh SPC Token, if expired request new SPC Token from STI-PA
Set token to respond to challenge – POST /acme/authz/1234/0
200 OK – with updated challenge in body
Request STI-PA public key certificate to validate SPC Token is signed by PA - GET /sti-pa/cert.crt
200 OK
Validate SPC Token in challenge with PA public key cert, and set authz status to "valid" for success
Check that authz status is "valid" – POST /acme/authz/1234
200 OK – if "valid" then continue, if "pending" repeat authz until "valid"
Finalize oder – POST /acme/order/asdf/finalize
200 OK –order update status is "processing"
Poll for status to be "valid" – POST /acme/order/1234
200 OK – with "valid" status and link to final cert
Download the STI Certificate – POST /acme/cert/mAt3xBGaobw
200 OK – with certificate in body

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 recommended 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 Revocation

It is anticipated that initially many service providers will not support short-lived certificates; thus, a mechanism to handle certificate revocation is required. Rather than each STI-CA publishing a Certificate Revocation List (CRL), an indirect CRL published by the STI-PA shall be used, following the model outlined in RFC 5280 [Ref 11]. The CRL shall be an X.509 V2 CRL format as detailed in RFC 5280 [Ref 11] and Clause 6.4.2.

It is anticipated that the list will not be large given that service providers are not expected to be using a large number of certificates initially and some service providers will choose to use short-lived certificates. The Certification Practice Statement (CPS) shall outline the criteria under which a specific STI-CA would revoke a certificate. Service providers likely will establish their own criterion as well, thus an STI-CA shall provide a mechanism that allows an SP to revoke a certificate. The STI-CA or Service Provider shall notify the STI-PA, when a certificate is revoked via a mechanism as defined by the Certificate Policy (CP) established by the STI-PA. Initially, an out-of-band mechanism is deemed sufficient, until operational experience indicates otherwise.

The URL to the STI-PA CRL shall be provided to the service providers for inclusion in the CSR. Given the static nature of this URL, it does not need to be frequently updated. Rather than defining a separate API, this URL shall be included as a field in the response to the SPC Token Request (Clause 6.3.4.2), per the following diagram:

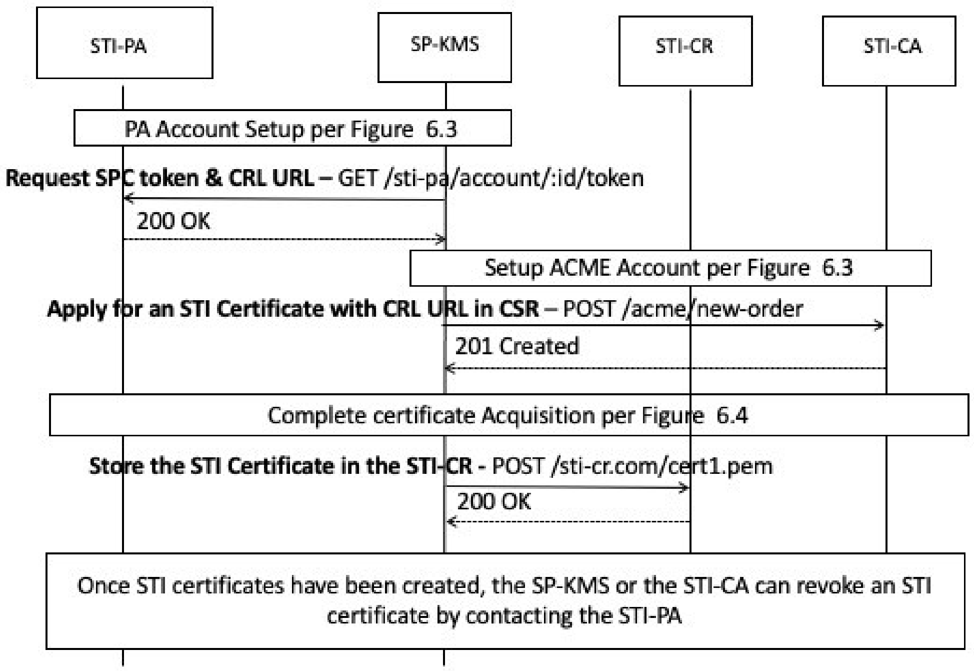
STI-PA
SP-KMS
STI-CR
STI-CA
PA Account Setup per Figure 6.3
Request SPC token & CRL URL – GET /sti-pa/account/:id/token
200 OK
Setup ACME Account per Figure 6.3
Apply for an STI Certificate with CRL URL in CSR – POST /acme/new-order
201 Created
Complete certificate Acquisition per Figure 6.4
Store the STI Certificate in the STI-CR - POST /sti-cr.com/cert1.pem
200 OK
Once STI certificates have been created, the SP-KMS or the STI-CA can revoke an STI certificate by contacting the STI-PA

Figure 6.5 – Distribution of the CRL

The inclusion of the STI-PA CRL in the STI Certificates follows standard practices per RFC 5280 [Ref 11] for inclusion of a CRL distribution point in a certificate. In the case of SHAKEN, the STI-VS uses this field to ensure that the STI Certificate used to sign the PASSporT in the SIP Identity header field has not been revoked, per the following diagram:

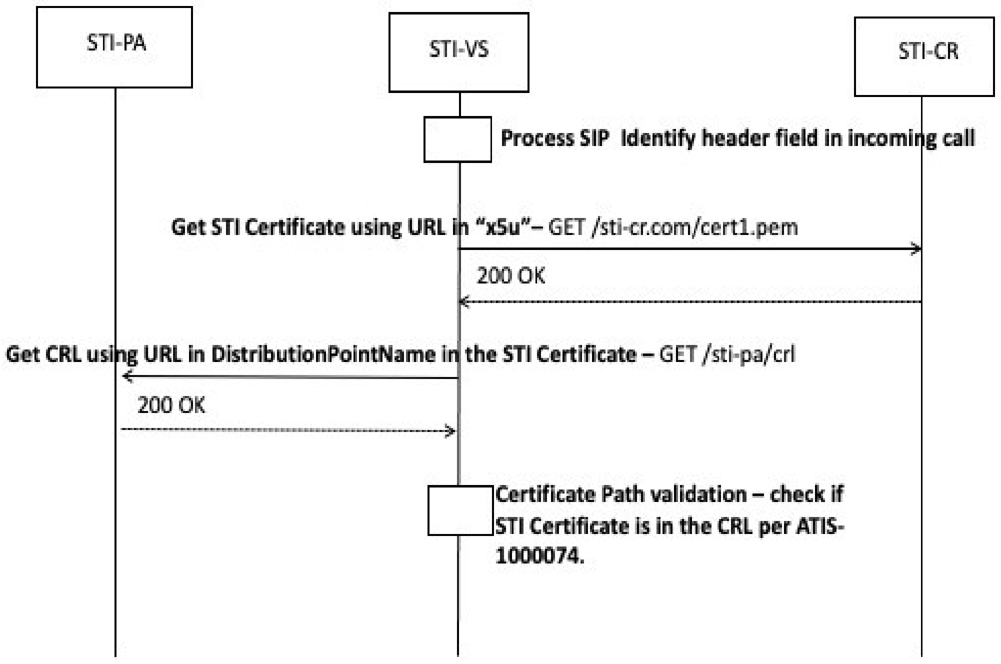
STI-PA
STI-VS
STI-CR
Process SIP Identify header field in incoming call
Get STI Certificate using URL in "x5u"– GET /sti-cr.com/cert1.pem
200 OK
Get CRL using URL in DistributionPointName in the STI Certificate – GET /sti-pa/crl
200 OK
Certificate Path validation – check if STI Certificate is in the CRL per ATIS-1000074.

Figure 6.6 – Using the CRL

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 RFC 8226 [Ref 20].

6.4 STI Certificate and Certificate Revocation List (CRL) Profile for SHAKEN

This section provides the detailed requirements for the attributes that shall be included in the STI Certificate and Certificate Revocation List.

6.4.1 STI Certificate Requirements

This section defines the STI Certificate profile that shall be supported by SHAKEN-compliant STI-CAs and Service Providers.

NOTE: The term "STI Certificates" in this section refers to End-Entity certificates containing a TNAuthList extension as defined in Clause 6.3.5.1 of this document, plus any intermediate/root certificate in the certification path of an STI End-Entity certificate.

STI certificates shall contain Version field specifying version 3 (value 2).

STI certificates shall include a Serial Number field containing a serial number that is unique within the scope of the issuing STI-CA.

STI certificates shall contain a Signature Algorithm field with the value "ecdsa-with-SHA256".

STI certificates shall include a Subject field containing a Distinguished Name (DN), which is unique for each subject entity certified under one CA issuer identity, as specified in RFC 5280 [Ref 11]. The DN shall contain a Country (C=) attribute, a Common Name (CN=) attribute and an Organization (O=) attribute. Other DN attributes are optional. For non-End-Entity CA certificates (Basic Constraints CA boolean = TRUE), the Common Name attribute shall include the text string "SHAKEN" and also indicate whether the certificate is a root or intermediate certificate (e.g., CN=SHAKEN root). The Common Name attribute of an End-Entity certificate shall contain the text string “SHAKEN”, followed by a single space, followed by the SPC value identified in the TNAuthList of the End-Entity certificate (e.g., "CN=SHAKEN 1234"). The Organization (O=) attribute shall include a legal name of the service provider in order to facilitate traceback and operations.

STI certificates shall include an Issuer field. For root certificates, the Issuer field shall match the certificate’s Subject field. For intermediate and End-Entity certificates, the Issuer field shall match the Subject field of the parent certificate.

STI certificates shall contain a Subject Public Key Info field specifying a Public Key Algorithm of "id-ecPublicKey” and containing a 256-bit public key.

STI certificates shall contain a BasicConstraints extension marked critical. For root and intermediate certificates, the BasicConstraints CA boolean shall be set to TRUE, while for End-Entity certificates, the CA boolean shall be set to FALSE.

STI certificates shall contain a Subject Key Identifier extension which is unique for each certificate. The value for the Subject Key Identifier is recommended to be derived from the public key of the certificate (e.g., a 160-bit SHA-1 hash of the public key, as described in RFC 5280 [Ref 11]). The value for the Subject Key Identifier for a root or intermediate certificate shall be the value placed in the Key Identifier field of the Authority Key Identifier extension of certificates issued by the subject of the root or intermediate certificate.

STI intermediate and End-Entity certificates shall contain an Authority Key Identifier extension (this extension is optional for root certificates). For root certificates that contain an Authority Key Identifier extension, the Authority Key Identifier shall contain a keyIdentifier field with a value that matches the Subject Key Identifier value of the same root certificate. For intermediate and End-Entity certificates, the Authority Key Identifier extension shall contain a keyIdentifier field with a value that matches the Subject Key Identifier value of the parent certificate.

STI certificates shall contain a Key Usage extension marked as critical. For root and intermediate certificates, the Key Usage extension shall contain the key usage value keyCertSign (5), and may contain the key usage values digitalSignature (0) and/or cRLSign (6). For End-Entity certificates, the Key Usage extension shall contain a single key usage value of digitalSignature (0).

STI intermediate and End-Entity certificates shall contain a CRL Distribution Points extension containing a single DistributionPoint entry. The DistributionPoint entry shall contain a distributionPoint field identifying the HTTP URL reference to the file containing the SHAKEN CRL hosted by the STI-PA, and a cRLIssuer field that contains the DN of the issuer of the CRL.

STI intermediate and End-Entity certificates shall include a Certificate Policies extension containing a single OID value that identifies the SHAKEN Certificate Policy established by the STI-PA. The OID value is specified in the SHAKEN Certificate Policy document.

STI End-Entity certificates shall contain a TNAuthList extension as specified in RFC 8226 [Ref 20]. The TNAuthList shall contain a single SPC value.

The private key of an STI root or intermediate certificate shall be used to sign STI certificates, and may be used to sign other items that are used internally by the STI-CA (i.e., internal CRLs). Likewise, the private key of an STI End-Entity certificate shall only be used to sign PASSporTs, supported by SHAKEN-compliant authentication services (e.g., PASSporTs supporting the "shaken" [Ref 22], "rph" or "div" extensions).

STI certificate examples are provided in Appendix A.

6.4.2 SHAKEN CRL Requirements

Per RFC 5280 [Ref 11], the STI-PA shall populate the CRL with the following fields and values:

1) The tbsCertList element shall be constructed as specified in Clause 6.4.2.1.
2) The Authority Key Identifier extension shall contain a Key Identifier field populated with the Subject Key Identifier value of the STI-PA certificate used to sign the CRL.
3) CRL Number extension shall contain a sequence number that is monotonically incremented each time a new CRL is issued (i.e., each time the tbsCertList "This Update" field is updated).
4) The Issuing Distribution Point extension shall contain an indirectCRL boolean set to TRUE. All other Issuing Distribution Point extension booleans shall be set to FALSE. If a distributionPoint field is included in the Issuing Distribution Point, then it shall match the distributionPoint field of the CRL Distribution Points extension of every certificate identified by the Revoked Certificates list of the CRL. The Issuing Distribution Point extension shall not contain an onlySomeReasons field.
5) The Signature Algorithm shall contain the value "ecdsa-with-SHA256".
6) The Signature Value shall be populated with a digital signature computed using the algorithm identified by the Signature Algorithm field in conjunction with the private key of the STI-PA certificate identified by the Authority Key Identifier field.

6.4.2.1 CRL tbsCertList Requirements

The tbsCertList element in the CRL contains the (possibly empty) list of revoked certificates. The scope of the STI-PA CRL is STI Certificates that have been revoked by one of the STI-CAs in the list of trusted STI-CAs or by a Service Provider. The tbsCertList shall not include expired certificates.

The tbsCertList shall be populated as follows:

1) The Signature field shall contain the algorithm identified by the CRL Signature Algorithm field.
2) The Issuer field shall contain the Subject field value of the STI-PA certificate that was used to sign this CRL.
3) The “This Update” field shall contain the issue date of the CRL encoded as UTCTime.
4) The “Next Update” field shall indicate the issue date of the next CRL, encoded as UTCTime. The next CRL may be issued before and shall be issued no later than the “Next Update” date. The STI-PA shall set the “Next Update” field value to the “This Update” field value plus 24 hours, in order to ensure that verifiers download the CRL on a timely basis.
5) The Authority Information Access extension shall contain an accessMethod of id-ad-caIssuers and an accessLocation with an HTTPS URL referencing the file that contains the STI-PA certificate that can be used to verify the signature of the CRL (i.e., a certificate whose Subject name matches the DN of the issuer of the CRL).
6) The Revoked Certificates list shall be included only if there are one or more revoked STI-CA certificates. When included in the CRL, each Revoked Certificates list entry shall identify a revoked certificate and provide information about its revocation by including the following fields and values:
   - The User Certificate field shall contain the Serial Number of the revoked STI Certificate.
   - The Certificate Issuer field shall contain a GeneralName identifying the STI-CA that issued the revoked STI Certificate.

   NOTE: Since the Serial Number of an STI certificate is unique within the scope of an STI-CA (see Clause 6.4.1), the combination of Serial Number and STI-CA identity uniquely identify the revoked certificate.

   - The Revocation Date shall contain the date that the STI-CA revoked the STI Certificate, encoded as UTCTime.
   - The Reason Code shall identify the reason that the STI Certificate was revoked.

Appendix A – SHAKEN Certificate Management Example with OpenSSL

(Informative)

A.1 TNAuthorizationList extension

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
```

Add output of the following command to the End-Entity section in OpenSSL configuration file:

```
# od -An -t x1 -w TNAuthList.der | sed -e 's/ /:/g' -e
's/^/1.3.6.1.5.5.7.1.26=DER/'
```

A.2 Setup directories

Assuming $HOME is /home/ubuntu/certs

```
# cd $HOME
# mkdir -p root intermediate private
```

A.3 Create private key and CSR

A.3.1. Create private key

```
# pwd
/home/ubuntu/certs/private
# openssl ecparam -name prime256v1 -genkey -noout -out private.key.pem
```

A.3.2. Create CSR from private key

```
# pwd
/home/ubuntu/certs/private
# openssl req -key private.key.pem -new -sha256 -out private.csr.pem -subj
"/C=US/ST=Pennsylvania/L=Philadelphia/O=Example CA/CN=SHAKEN"
```

A.4 Signing certificate using root CA

This Clause illustrates creating an End-Entity certificate from a root CA.

```
# cd $HOME/root
# pwd
/home/ubuntu/certs/root
```

SAVE THIS OPENSSL CONFIG IN `$HOME/root/openssl.cnf FILE`

```
[ ca ]
default_ca = CA_default

[ CA_default ]
default_md    = sha256
name_opt     = ca_default
cert_opt     = ca_default
preserve     = no
policy      = policy_strict
# directories and files
dir        = ./
```

```
database     = $dir/db
serial      = $dir/srl
new_certs_dir   = $dir/newcerts
private_key    = $dir/rootca.key.pem
certificate    = $dir/rootca.crt.pem

[ policy_strict ]
countryName       = match
stateOrProvinceName   = match
organizationName    = match
organizationalUnitName = optional
commonName       = supplied
emailAddress      = optional

[ req ]
default_bits    = 2048
string_mask     = utf8only
prompt       = no
distinguished_name = ca_dn

[ ca_dn ]
countryName       = US
stateOrProvinceName   = Pennsylvania
localityName      = Philadelphia
0.organizationName   = Example CA
commonName       = SHAKEN Root CA

[ ca_ext ]
subjectKeyIdentifier = hash
basicConstraints = critical, CA:true
keyUsage = critical, keyCertSign

[ int_ext ]
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always,issuer
basicConstraints = critical, CA:true
keyUsage = critical, keyCertSign
crlDistributionPoints=crldp1_section
certificatePolicies = 2.16.840.1.114569.1.1.1

[ leaf_cert ]
1.3.6.1.5.5.7.1.26=DER:30:08:a0:06:16:04:31:32:33:34
basicConstraints = critical, CA:FALSE
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer:always
```

```
keyUsage = critical, digitalSignature
crlDistributionPoints=crldp1_section

[crldp1_section]

fullname=URI:https://sti-pa.com/shaken/crl
CRLissuer=dirName:issuer_sect

[issuer_sect]
C=US
O=STI-PA
CN=STI-PA CRL
```

NOTE THAT *leaf_cert* SECTION CONTAINS TNAuthorizationList EXTENSION DERIVED FROM THE DER VALUE

```
# od -An -t x1 -w TNAuthList.der | sed -e 's/ /:/g' -e
's/^/1.3.6.1.5.5.7.1.26=DER/'
```

A.4.1. Create file to be used as certificate database by openssl

```
# pwd
/home/ubuntu/certs/root
# touch db
```

A.4.2. Create file that contains the certificate serial number

```
# pwd
/home/ubuntu/certs/root
# echo 1000 > srl
```

A.4.3. Create directories to be used to store keys, certificates and signing requests

```
# pwd
/home/ubuntu/certs/root
# mkdir -p newcerts
```

A.4.4. Create root key

```
# pwd
/home/ubuntu/certs/root
```

```
# openssl ecparam -name prime256v1 -genkey -noout -out rootca.key.pem
```

A.4.5. Create root certificate

```
# pwd
/home/ubuntu/certs/root
# openssl req -config openssl.cnf -key rootca.key.pem -new -x509 -days 7300
-sha256 -extensions ca_ext -out rootca.crt.pem
```

A.4.6. Verify root certificate

```
# pwd
/home/ubuntu/certs/root
# openssl x509 -in rootca.crt.pem -text -noout
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number: 12496366116147440257 (0xad6c02c628322a81)
  Signature Algorithm: ecdsa-with-SHA256
    Issuer: C=US, ST=Pennsylvania, L=Philadelphia, O=Example CA, CN=SHAKEN
Root CA
    Validity
      Not Before: Dec 9 23:06:34 2019 GMT
      Not After : Dec 4 23:06:34 2039 GMT
    Subject: C=US, ST=Pennsylvania, L=Philadelphia, O=Example CA, CN=SHAKEN
Root CA
    Subject Public Key Info:
      Public Key Algorithm: id-ecPublicKey
        Public-Key: (256 bit)
        pub:
          04:94:b3:67:34:de:36:1c:68:bc:bb:72:c2:17:73:
          41:4d:74:f4:96:4b:91:cc:57:8c:15:7d:5c:1f:e3:
          81:fb:fd:ab:2f:59:25:f7:0f:ef:1f:5c:ae:34:9b:
          cc:1b:b5:f8:8a:06:eb:94:20:be:0e:45:1b:3e:56:
          e9:74:75:70:a2
        ASN1 OID: prime256v1
        NIST CURVE: P-256
    X509v3 extensions:
      X509v3 Subject Key Identifier:
        91:90:CA:B1:86:0E:4F:16:5E:BE:B5:37:51:3F:69:79:E5:23:1B:1C

      X509v3 Basic Constraints: critical
        CA:TRUE
      X509v3 Key Usage: critical
        Certificate Sign
  Signature Algorithm: ecdsa-with-SHA256
     30:45:02:20:3a:52:c8:2b:99:c9:ee:5a:38:04:1d:c0:db:2f:
     3a:a4:e8:0c:42:52:cb:dc:3d:bf:57:ec:18:b8:f6:03:2b:7a:
     02:21:00:d5:7b:36:19:af:86:44:8d:31:d7:a0:88:72:a8:45:
     7b:f3:5f:4a:5b:be:e5:3c:01:05:8b:45:e4:93:1d:0d:f3
```

A.4.7. Sign CSR with root CA cert and create End-Entity certificate

- CSR was created in Clause A.3.2.

```
# pwd
/home/ubuntu/certs/root
# openssl ca -config openssl.cnf -extensions leaf_cert -days 375 -notext -
md sha256 -in ../private/private.csr.pem –out ../newcerts/1000.crt.pem
```

A.4.8. Verify End-Entity certificate

```
# pwd
/home/ubuntu/certs/root
# openssl x509 -in ../newcerts/1000.crt.pem -text -noout
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number: 4097 (0x1001)
  Signature Algorithm: ecdsa-with-SHA256
    Issuer: C=US, ST=Pennsylvania, L=Philadelphia, O=Example CA, CN=SHAKEN
Root CA
    Validity
      Not Before: Dec 9 23:38:35 2019 GMT
      Not After : Dec 18 23:38:35 2020 GMT
    Subject: C=US, ST=Pennsylvania, O=Example SP, CN=SHAKEN 1234
    Subject Public Key Info:
      Public Key Algorithm: id-ecPublicKey
        Public-Key: (256 bit)
        pub:
          04:20:ee:f3:47:0f:b4:ab:fd:56:74:25:c4:cc:e9:
          8f:81:2b:ae:fb:5d:24:3d:72:d7:62:16:5e:91:f0:
          1a:62:1e:96:da:13:4d:72:3d:fb:f0:3e:47:cf:80:
          3c:a7:3d:fa:74:7b:eb:6d:9e:00:e7:98:cb:d5:79:
          1b:37:11:58:59
        ASN1 OID: prime256v1
        NIST CURVE: P-256
    X509v3 extensions:
      1.3.6.1.5.5.7.1.26:
        0.....1234
      X509v3 Basic Constraints: critical
        CA:FALSE
      X509v3 Subject Key Identifier:
        B6:26:4C:D2:45:81:87:08:6E:09:EA:F9:66:8C:0F:8D:05:C2:E6:46
      X509v3 Authority Key Identifier:
        keyid:91:90:CA:B1:86:0E:4F:16:5E:BE:B5:37:51:3F:69:79:E5:23:1B:1C


      X509v3 Key Usage: critical
        Digital Signature
      X509v3 CRL Distribution Points:

          Full Name:
            URI:https://sti-pa.com/shaken/crl
          CRL Issuer:
            DirName:C = US, O = STI-PA, CN = STI-PA CRL

       X509v3 Certificate Policies:
          Policy: 2.16.840.1.114569.1.1.1

  Signature Algorithm: ecdsa-with-SHA256
     30:46:02:21:00:fa:4c:fb:ad:97:5a:1e:46:09:13:9c:5b:ef:
     a4:7f:82:a6:9d:6c:d9:1e:f8:07:9b:ab:de:5e:64:52:77:2e:
     f8:02:21:00:d5:b2:bd:d7:84:ee:ce:e0:e4:69:e7:ea:f9:e9:
     cf:35:b3:56:37:85:f8:1f:f4:47:5b:bf:f5:5d:9c:4d:62:2c
```

A.4.9. Verify chain of trust

```
# pwd
/home/ubuntu/certs/root
# openssl verify -CAfile rootca.crt.pem ../newcerts/1000.crt.pem
../newcerts/1000.crt.pem: OK
```

A.5 Signing certificate using intermediate CA

This Clause illustrates creating an End-Entity certificate from an intermediate CA of a root CA.

```
# cd $HOME/intermediate
# pwd
/home/ubuntu/certs/intermediate
```

SAVE THIS OPENSSL CONFIG IN **`$HOME/intermediate/openssl.cnf FILE`**

```
[ ca ]
default_ca = CA_default

[ CA_default ]
default_md    = sha256
name_opt     = ca_default
cert_opt     = ca_default
preserve     = no
policy      = policy_strict
# directories and files
dir        = ./
database     = $dir/db
serial      = $dir/srl
new_certs_dir   = $dir/newcerts
private_key    = $dir/intermediate.key.pem
certificate    = $dir/intermediate.crt.pem

[ policy_strict ]
countryName       = match
stateOrProvinceName   = match
organizationName    = match
organizationalUnitName = optional
commonName       = supplied
emailAddress      = optional

[ req ]
default_bits    = 2048
string_mask     = utf8only
prompt       = no
distinguished_name = int_dn

[ int_dn ]
countryName       = US
stateOrProvinceName   = Pennsylvania
localityName      = Philadelphia
0.organizationName   = Example CA
commonName       = SHAKEN Intermediate CA

[ int_ext ]
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always,issuer
basicConstraints = critical, CA:true
keyUsage = critical, keyCertSign

[ leaf_cert ]
```

```
1.3.6.1.5.5.7.1.26=DER:30:08:a0:06:16:04:31:32:33:34
basicConstraints = critical, CA:FALSE
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer:always
keyUsage = critical, digitalSignature
```

NOTE THAT *leaf_cert* SECTION CONTAINS TNAuthorizationList EXTENSION DERIVED FROM THE DER VALUE

```
# od -An -t x1 -w TNAuthList.der | sed -e 's/ /:/g' -e
's/^/1.3.6.1.5.5.7.1.26=DER/'
```

A.5.1. Create file to be used as certificate database by openssl

```
# pwd
/home/ubuntu/certs/intermediate
# touch db
```

A.5.2. Create file that contains the certificate serial number

```
# pwd
/home/ubuntu/certs/intermediate
# echo 1000 > srl
```

A.5.3. Create directories to be used to store keys, certificates and signing requests

```
# pwd
/home/ubuntu/certs/intermediate
# mkdir -p newcerts
```

A.5.4. Create intermediate key

```
# pwd
/home/ubuntu/certs/intermediate
# openssl ecparam -name prime256v1 -genkey -noout -out intermediate.key.pem
```

A.5.5. Create CSR from intermediate key

```
# pwd
/home/ubuntu/certs/intermediate
# openssl req -config openssl.cnf -new -sha256 -key intermediate.key.pem -
out intermediate.csr.pem
```

A.5.6. Create intermediate certificate

```
#cd $HOME/root
# pwd
/home/ubuntu/certs/root
# openssl ca -config openssl.cnf -extensions int_ext -days 7000 -notext -md
sha256 -in ../intermediate/intermediate.csr.pem -out
../intermediate/intermediate.crt.pem
```

A.5.7. Verify intermediate certificate

```
#cd $HOME/intermediate
# pwd
/home/ubuntu/certs/intermediate
# openssl x509 -in intermediate.crt.pem -text -noout
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number: 4099 (0x1003)
  Signature Algorithm: ecdsa-with-SHA256
    Issuer: C=US, ST=Pennsylvania, L=Philadelphia, O=Example CA, CN=SHAKEN
Root CA
    Validity
      Not Before: Dec 10 02:20:30 2019 GMT
      Not After : Feb 8 02:20:30 2039 GMT
    Subject: C=US, ST=Pennsylvania, O=Example CA, CN=SHAKEN Intermediate CA
    Subject Public Key Info:
      Public Key Algorithm: id-ecPublicKey
        Public-Key: (256 bit)
        pub:
          04:17:74:4c:94:75:2c:f4:d7:cf:c0:8e:5a:50:17:
          0b:4a:0a:84:84:ba:71:c8:5a:23:49:d3:7e:24:3e:
          4b:b6:2e:59:9d:03:f1:60:ae:0f:6b:10:f7:65:d7:
          a5:41:66:66:16:27:41:5c:12:a7:61:6c:a0:82:e7:
          f6:2c:bb:89:b3
        ASN1 OID: prime256v1
        NIST CURVE: P-256
    X509v3 extensions:
      X509v3 Subject Key Identifier:
        E0:15:BC:55:D7:9A:7A:0D:18:67:D8:7E:82:1D:AD:35:D9:54:DD:60
      X509v3 Authority Key Identifier:
        keyid:91:90:CA:B1:86:0E:4F:16:5E:BE:B5:37:51:3F:69:79:E5:23:1B:1C

      X509v3 Basic Constraints: critical
        CA:TRUE
      X509v3 Key Usage: critical
        Certificate Sign
      X509v3 CRL Distribution Points:

        Full Name:
          URI:https://sti-pa.com/shaken/crl
        CRL Issuer:
          DirName:C = US, O = STI-PA, CN = STI-PA CRL

      X509v3 Certificate Policies:
        Policy Identifier = 2.16.840.1.114569.1.1.1

  Signature Algorithm: ecdsa-with-SHA256
     30:45:02:20:75:28:f9:51:25:ba:5f:65:71:de:b8:bc:72:51:
     d1:75:34:ef:be:3c:7a:39:a5:42:ef:46:81:90:c7:16:b6:46:
     02:21:00:f7:b6:c6:78:86:df:40:4d:71:fc:41:3a:83:c6:a0:
     2c:52:c3:c6:47:9f:6a:bb:20:be:69:5e:18:71:e0:09:b5
```

A.5.8. Sign CSR with intermediate cert and create End-Entity certificate

- CSR was created in section A.3.2.

```
# pwd
/home/ubuntu/certs/intermediate
# openssl ca -config openssl.cnf -extensions leaf_cert -days 375 -notext -
md sha256 -in ../private/private.csr.pem -out ../private/private.crt.pem
```

A.5.9. Verify End-Entity certificate

```
# pwd
/home/ubuntu/certs/private
# openssl x509 -in private.crt.pem -text -noout
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number: 4097 (0x1001)
  Signature Algorithm: ecdsa-with-SHA256
    Issuer: C=US, ST=Pennsylvania, O=Example CA, CN=SHAKEN
 Intermediate CA
    Validity
      Not Before: Dec 10 02:42:14 2019 GMT
      Not After : Dec 19 02:42:14 2020 GMT
    Subject: C=US, ST=Pennsylvania, O=Example SP, CN=SHAKEN 1234
    Subject Public Key Info:
      Public Key Algorithm: id-ecPublicKey
        Public-Key: (256 bit)
        pub:
          04:f6:7d:10:e0:3f:15:08:a5:f6:6d:6a:e6:4f:98:
          51:30:c5:8e:9c:a3:d3:4c:1f:a8:fa:af:c6:c3:38:
          1c:82:df:7a:19:f1:59:d1:81:42:5a:8d:35:22:3c:
          0f:56:82:ad:d0:49:38:f8:d9:65:0d:99:d8:74:62:
          78:b7:7a:ab:e4
        ASN1 OID: prime256v1
        NIST CURVE: P-256
    X509v3 extensions:
      1.3.6.1.5.5.7.1.26:
        0.....1234
      X509v3 Basic Constraints: critical
        CA:FALSE
      X509v3 Subject Key Identifier:
        07:D5:04:6D:F0:52:1F:EE:FD:B9:BD:0C:97:45:45:B0:33:D1:C1:CD
      X509v3 Authority Key Identifier:
        keyid:E0:15:BC:55:D7:9A:7A:0D:18:67:D8:7E:82:1D:AD:35:D9:54:DD:60


      X509v3 Key Usage: critical
        Digital Signature
      X509v3 CRL Distribution Points:

        Full Name:
          URI:https://sti-pa.com/shaken/crl
        CRL Issuer:
          DirName:C = US, O = STI-PA, CN = STI-PA CRL

       X509v3 Certificate Policies:
         Policy: 2.16.840.1.114569.1.1.1

  Signature Algorithm: ecdsa-with-SHA256
     30:45:02:21:00:83:b8:d6:f4:3b:20:f6:90:40:98:88:eb:97:
     84:4a:b2:e6:d7:a5:a1:e9:3a:95:8b:2c:81:7a:3e:cc:b4:86:
     4d:02:20:10:04:2b:0e:1c:42:fa:1e:37:4b:78:12:27:81:6e:
     b1:ac:f4:1c:61:68:17:18:ed:f8:78:96:b6:37:76:e5:ca
```

A.5.10. Verify chain of trust

```
# pwd
/home/ubuntu/certs/intermediate
# cat intermediate.crt.pem ../root/rootca.crt.pem > chain.crt.pem
# openssl verify -CAfile chain.crt.pem ../private/private.crt.pem
../private/private.crt.pem: OK
```