**ATIS-0x0000x**

ATIS Standard on

**Technical Report on SHAKEN API for a Centralized Signing and Signature Validation Server**

**Alliance for Telecommunications Industry Solutions**

Approved Month DD, YYYY

**Abstract**

This document provides a Technical Report on Originating Party Spoofing in IP Communication Networks. It describes problems associated with originating party spoofing in IP communication networks, identifies potential mitigation options, analyze pros and cons of mitigation options.

**Foreword**

The Alliance for Telecommunications Industry Solutions (ATIS) serves the public through improved understanding between carriers, customers, and manufacturers. The [**COMMITTEE NAME**] Committee [**INSERT MISSION**]. [**INSERT SCOPE**].

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 a optional capability that could augment the standard. The standard is fully functional without the incorporation of this optional capability.

Suggestions for improvement of this document are welcome. They should be sent to the Alliance for Telecommunications Industry Solutions, [**COMMITTEE NAME**], 1200 G Street NW, Suite 500, Washington, DC 20005.

At the time of consensus on this document, [**COMMITTEE NAME**], which was responsible for its development, had the following leadership:

[**LEADERSHIP LIST**]

The **[SUBCOMMITTEE NAME]** Subcommittee was responsible for the development of this document.

**Revision History**

| **Date** | **Version** | **Description** | **Author** |
| --- | --- | --- | --- |
|  |  |  |  |

**Table of Contents**

[INSERT]

**Table of Figures**

[INSERT]

**Table of Tables**

[INSERT]

# Introduction

This technical report provides

# Normative References

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

# 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> >.

## Definitions

**Caller identity:** The originating phone number included in call signalling used to identify the caller for call screening purposes.In some cases this may be the Calling Line Identification or Public User Identity. For the purposes of this study, the caller identity may be set to an identity other than the caller’s Calling Line Identification or Public User Identity.

## Acronyms & Abbreviations

|  |  |
| --- | --- |
| Acronym | Term |
| STI  | Secure Telephone Identity |
| STI-AS | STI Authentication Service |
| STI-VS | STI Verification Service |
| SHAKEN  | Signature based HAndling of Asserted Information using toKENs  |
| STIR | Secure Telephone Identity Revisited |
| UUID  | Universally Unique Identifier |
| PASSporT | Persona Assertion Token |

# Architecture



Figure 4.1 – SHAKEN Reference Architecture

Xxxxx



# API Interface

## Signing API

###  Functional Behavior

Used to create the PASSporT signature with private key certificate.

1. Validate the incoming signing request parameters in terms of parameter’s type and format.

2. Validate the “iat” parameter value in terms of “freshness”: the request with “iat” value with time different by more than one minute from the current time will be rejected.

3. Normalize to the canonical form the received telephony numbers if needed (remove visual separators and leading “+”).

3. Build SHAKEN PASSport protected header (with “ppt” SHAKEN extension).

4. Build SHAKEN PASSporT header and payload by keeping lexicographic order and removing space and line breaking characters.

6. Generate PASSporT signature with appropriate certificate private key.

7. Build Full Form of PASSporT.

8. Build SIP “Identity” header value by using identity digest from the previous step and add “info” parameter with angle bracketed URI to acquire the public key of certificate used during PASSporT signing

9. In case of successfully signing build and send “siginingResponse”, otherwise send error.

###  Call Flow



* + 1. **Request (POST)**

The used resource is: https://{serverRoot}/stir/v1/signing

|  |  |
| --- | --- |
| Name  | Description |
| serverRoot | Server base URL : hostname+port+base pathHostname shall contain the Global FQDN of Signing Service |

* + - 1. **Request Body**

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Data Type | Required? | Brief description |
| Signing Request | signingRequest | Yes | Contains the JSON structure of the signing request (PASSporT payload claims)  |

* + - 1. **Request Sample**
1. POST /stir/v1/signing HTTP/1.1
2. Host : stir.att.com
3. Accept : application/json
4. X-RequestID: AA97B177-9383-4934-8543-0F91A7A02836
5. Content-Type: application/json
6. Content-Length : …

{

 "signingRequest”: {

 "attest": “A”,

 "orig”: {

 “tn”: “12155551212”

 },

 “dest”: {

 “tn” : [

 “12355551212”

 ]

 },

 "iat”: 1443208345,

 “origid”: “de305d54-75b4-431b-adb2-eb6b9e546014”

 }

}

* + 1. **Response**
			1. **Response Body**

Response body is returned as JSON object (Content-Type: application/son).

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Data Type | Required? | Brief description |
| Signing Response | signingResponse | Yes | Contains the JSON structure of the signing response (SIP Identity header value). |

* + - 1. **Response Sample (Success)**
1. HTTP/1.1 200 Ok
2. X-RequestID: AA97B177-9383-4934-8543-0F91A7A02836
3. Content-Type : application/json
4. Content-Length : …

{

 "signingResponse": {

 "identity" : “eyJhbGciOiJFUzI1NiIsInR5cCI6InBhc3Nwb3J0IiwicHB0Ijoic2hha2VuIiwieDV1IjoiaHR0cDov L2NlcnQtYXV0aC5wb2Muc3lzLmNvbWNhc3QubmV0L2V4YW1wbGUuY2VydCJ9eyJhdHRlc3QiOiJBIiwiZGVzdCI6eyJ0biI6IisxMjE1NTU1MTIxMyJ9LCJpYXQiOiIxNDcxMzc1NDE4Iiwib3JpZyI6eyJ0biI64oCdKzEyMTU1NTUxMjEyIn0sIm9yaWdpZCI6IjEyM2U0NTY3LWU4OWItMTJkMy1hNDU2LTQyNjY1NTQ0MDAwMCJ9.\_28kAwRWnheXyA6nY4MvmK5JKHZH9hSYkWI4g75mnq9Tj2lW4WPm0PlvudoGaj7wM5XujZUTb\_3MA4modoDtCA;info=<<http://cert.example2.net/example.cert>>”

 }

}

* + - 1. **Response Sample (Failure)**
1. HTTP/1.1 400 Bad Request
2. X-RequestID: AA97B177-9383-4934-8543-0F91A7A02836
3. Content-Type : application/json
4. Content-Length : …

{

 “requestError”: {

 “serviceException”: {

 “messageId”: “SVC4501”

 “text”: “Error: Invalid Content. Missing mandatory parameter ‘%1’”,

 “variables”: [“iat”]

 }

 }

}

* + - 1. **HTTP Response Codes**

|  |  |  |
| --- | --- | --- |
| Response code | Service/PolicyException | Reason /Description |
| 200 | N/A | Successful signing |
| 400 | SVC4000 | Missing JSON body in the request |
| 400 | SVC4001 | Missing mandatory parameter |
| 406 | SVC4002 | Not supported body type is specified in Accept HTTP header |
| 415 | SVC4004 | Received unsupported message body type in Content-Type HTTP header |
| 400 | SVC4005 | Invalid parameter value |
| 400 | SVC4006 | Failed to parse JSON body |
| 411 | SVC4007 | Missing mandatory Content-Length header |
| 405 | POL4050 | Method Not Allowed : Invalid HTTP method used ( all methods except POST will be rejected for the specific resource URL)  |
| 500  | POL5000 | The POST request failed either due to internal signing server problem. |

* 1. **Verification API**
		1. **Functional Behavior**

 Used to verify the signature provided in the Identity header and to determine that the signing service credentials demonstrate authority over the call originating identity. Please find below the validations steps. Each step is associated with appropriate error case specified in the section “Mapping of verification failure cases to the returned SIP Reason header parameters” The error case numbers **En** per each step is specified in parentheses.

1. Validate the incoming verification request parameters in terms of parameter’s type and format (E1 and E2).

2. Validate the “iat” parameter value in terms of “freshness”: the request with “iat” value with time different by more than one minute from the current time on will be rejected (E3)

3. Parse “identity” parameter value:

 - full form of PASSporT is required by SHAKEN : “identity-digest” parameter of Identity header has to be parsed to validate the full form format ( 3 data portions delimited with dot (“.”) ) .If the expected format is not matched 🡪 reject request on the Invalid PASSporT form (E4)

 - If “ppt” parameter is specified and its value is not “shaken” 🡪 reject request (E5)

 - If “info” parameter is not specified 🡪 reject request (E6)

 - If the URI specified in “info” parameter is not syntactically valid 🡪 reject request (E7)

4. Decode “identity-digest” parameter value to extract from the first portion (**PASSporT header** ) “ppt” , “typ”,”alg” and “x5u” claims :

 - If one of the mentioned claims is missing -> reject request ( E9)

 - if extracted “typ” value is not equal to “passport” 🡪 reject request (E11)

 - if extracted “alg” value is not equal to “ES256” 🡪 reject request ( E12)

 - if extracted “x5u” value is not equal to the URI specified in the “info” parameter of Identity header 🡪 reject request (E10)

 - If extracted “ppt” is not equal to “shaken” 🡪 reject request (E13)

5. Decode “identity-digest” parameter value to extract from the second portion (**PASSporT payload**) “dest” , “orig” , “attest”, “origid” and “iat” claims :

 - on missing mandatory claims reject request ( E14)

 - validate the extracted from payload “iat” claim value in terms of “freshness”: request with “expired” “iat” will be rejected🡪 reject request (E15)

 - Normalize to the canonical form the received in the “verificationRequest” “orig” and “dest” telephone numbers (remove visual separators and leading “+”) and compare them with ones extracted from the

“orig” and “dest” claims of PASSporT payload. If they are not identical 🡪 reject request (E16)

 6. Dereference “info” parameter URI to a resource that contains the public key of the certificate used by signing service to sign a request .On failure to dereference URI due to timeout/not existing resource the request should be rejected ( E8).

7. Authenticate the receieved CA . On the failure to authenticate the CA ( for example not valid, no root CA) request will be rejected (E17))

8. Validate the signature of “identity” digest parameter. On failure reject the request (E18).

9. No “origid” or/and “attest” claim/s in the decrypted PASSporT payload 🡪 reject (E19 , E20)).

10. Compare all PASSporT claims from decrypted identity digest ( except “origid” and “attest”) and claims from PASSporT header and payload validated at step 5 and 6 above. If one of the claim’s value is not matched the request should be be rejected ( E21 , E22) .

* + 1. **Call Flow**



* + 1. **Request (POST)**

The used resource is: https://{serverRoot}/stir/v1/verification

|  |  |
| --- | --- |
| Name  | Description |
| serverRoot | Server base URL : hostname+port+base pathHostname shall contain the Global FQDN of Verification Service |

* + - 1. **Request Body**

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Data Type | Required? | Brief description |
| Verification Request | verificationRequest | Yes | Contains the JSON structure of the verification request (PASSporT payload claims + identity header)  |

* + - 1. **Request Sample**
1. POST /stir/v1/verification HTTP/1.1
2. Host : stir.att.com
3. Accept : application/json
4. X-RequestID: AA97B177-9383-4934-8543-0F91A7A02836
5. Content-Type: application/json
6. Content-Length : …

{

 “verificationRequest”: {

 “orig”: {

 “tn”: “12155551212”

 },

 “dest”: {

 “tn” : [

 “12355551212”

 ]

 },

 “iat”: 1443208345,

 “identity” : “eyJhbGciOiJFUzI1NiIsInR5cCI6InBhc3Nwb3J0IiwicHB0Ijoic2hha2VuIiwieDV1IjoiaHR0cDov L2NlcnQtYXV0aC5wb2Muc3lzLmNvbWNhc3QubmV0L2V4YW1wbGUuY2VydCJ9eyJhdHRlc3QiOiJBIiwiZGVzdCI6eyJ0biI6IisxMjE1NTU1MTIxMyJ9LCJpYXQiOiIxNDcxMzc1NDE4Iiwib3JpZyI6eyJ0biI64oCdKzEyMTU1NTUxMjEyIn0sIm9yaWdpZCI6IjEyM2U0NTY3LWU4OWItMTJkMy1hNDU2LTQyNjY1NTQ0MDAwMCJ9.\_28kAwRWnheXyA6nY4MvmK5JKHZH9hSYkWI4g75mnq9Tj2lW4WPm0PlvudoGaj7wM5XujZUTb\_3MA4modoDtCA;info=<<http://cert.example2.net/example.cert>>”

 }

}

* + 1. **Response**
			1. **Response Body**

Response body is returned as JSON object (Content-Type: application/son).

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Data Type | Required? | Brief description |
| Verification Response | verificationResponse | Yes | Contains the JSON structure of the verification response. |

* + - 1. **Mapping of verification failure cases to the returned SIP Reason header parameters**

| Error Case Number | Error Case  | HTTP Status Code | SIP Reason Code | SIP Reason Text | “verstat”  |
| --- | --- | --- | --- | --- | --- |
| E1 | Missing mandatory parameters in the verification request ( “orig’, “dest” ,”iat” , “identity”) | 400 with service exception | - | - | Explicitly should be handled as TN-Validation-Failed |
| E2 | Received invalid parameters( invalid “tn” , “iat” value …) | 400 with service exception  | - | - | Explicitly should be handled as TN-Validation-Failed |
| E3 | Received “iat” value is not “fresh”  | 200  | 403 | Stale Date | TN-Validation-Failed |
| E4 | Identity header with “identity-digest” in compact form instead of required by SHAKEN spec full form. | 200 | 438 | InvalidIdentity Header | TN-Validation-Failed |
| E5 | Identity header is received with “ppt” parameter value that is not “shaken”  | 200 | 438 | InvalidIdentity Header | TN-Validation-Failed |
| E6 | Missing “info” parameter in the “identity”  | 200 | 436 | Bad identity Info | TN-Validation-Failed |
| E7 | “info” parameter from “identity” is invalid (syntactically invalid URI) | 200  | 436 | Bad identity Info | TN-Validation-Failed |
| E8 | Failed to dereference “info” URI  | 200  | 436 | Bad identity Info | TN-Validation-Failed |
| E9 | “ppt”/”typ”/”alg”/”x5u” claims missing in the PASSporT header  | 200 | 436 | Bad identity Info | TN-Validation-Failed |
| E10 | “x5u” from PASSporT header doesn’t match the “info” parameter of identity header value | 200  | 436 | Bad identity Info | TN-Validation-Failed |
| E11 | “typ” from PASSporT header is not “passport” | 200  | 437  | Unsupported credential | TN-Validation-Failed |
| E12 | “alg” from PASSporT header is not “ES256” | 200 | 437  | Unsupported credential | TN-Validation-Failed |
| E13 | “ppt” from PASSporT header is not “shaken” | 200 | 438 | InvalidIdentity Header | TN-Validation-Failed |
| E14 | Missing mandatory claims in PASSporT payload ( “dest” , “orig” , “attest” , “origid” ) | 200 | 438 | InvalidIdentity Header | TN-Validation-Failed |
| E15 | “iat” from in PASSporT payload is not “fresh” | 200  | 403 | Stale Date | TN-Validation-Failed |
| E16 | “orig” and “dest” claims from PASSporT payload don’t match the received in the verification request corresponding claims  | 200 | 438 | InvalidIdentity Header | TN-Validation-Failed |
| E17 | Failed to authenticate CA | 400 | 437 | Unsupported credential | TN-Validation-Failed |
| E18 | Signature validation failed  | 200  | 438 | InvalidIdentity Header | TN-Validation-Failed |
| E19 | Missing SHAKEN extension “attest” claim in the decrypted PASSporT | 200 | 438 | InvalidIdentity Header | TN-Validation-Failed |
| E20 | Missing SHAKEN extension “origid” claim in the decrypted PASSporT | 200 | 438 | InvalidIdentity Header | TN-Validation-Failed |
| E21 | “orig” /”dest” claims from decrypted payload don’t match the ones received in the INVITE  | 200 | 438  | Invalid Identity Header  | TN-Validation-Failed |
| E22 | “iat” claim from decrypted payload doesn’t match the “iat” from PASSporT payload. | 200 | 438  | Invalid Identity Header  | TN-Validation-Failed |
| E23 | Successful verification | 200 | - | - | TN-Validation-Passed |

* + - 1. **Response Sample (Success + Successful Validation)**
1. HTTP/1.1 200 Ok
2. X-RequestID: AA97B177-9383-4934-8543-0F91A7A02836
3. Content-Type : application/json
4. Content-Length : …

{

 "verificationResponse": {

 “verstat”: “TN-Validation-Passed”

 }

}

* + - 1. **Response Sample (Success + Failed Validation)**
1. HTTP/1.1 200 Ok
2. X-RequestID: AA97B177-9383-4934-8543-0F91A7A02836
3. Content-Type : application/json
4. Content-Length : …

{

 "verificationResponse": {

 “reasoncode”: 436200,

 “reasontext”: “Bad Identity Info”,

 “reasondesc”: “Info URI dereferencing failure”,

 “verstat”: “TN-Validation-Failed”

 }

}

* + - 1. **Response Sample (Failure)**
1. HTTP/1.1 400 Bad Request
2. X-RequestID: AA97B177-9383-4934-8543-0F91A7A02836
3. Content-Type : application/json
4. Content-Length : …

{

 “requestError”: {

 “serviceException”: {

 “messageId”: “SVC4501”

 “text”: “Error: Invalid Content. Missing mandatory parameter ‘%1’”,

 “variables”: [“iat”]

 }

 }

}

* + - 1. **HTTP Response Codes**

|  |  |  |
| --- | --- | --- |
| Response code | Service/PolicyException | Reason /Description |
| 200 | N/A | Successful signing |
| 400 | SVC4000 | Missing JSON body in the request |
| 400 | SVC4001 | Missing mandatory parameter |
| 406 | SVC4002 | Not supported body type is specified in Accept HTTP header |
| 415 | SVC4004 | Received unsupported message body type in Content-Type HTTP header |
| 400 | SVC4005 | Invalid parameter value |
| 400 | SVC4006 | Failed to parse JSON body |
| 411 | SVC4007 | Missing mandatory Content-Length header |
| 405 | POL4050 | Method Not Allowed : Invalid HTTP method used ( all methods except POST will be rejected for the specific resource URL)  |
| 500  | POL5000 | The POST request failed either due to internal signing server problem. |

# Conclusions

**Annex A**

(normative/informative)

# A XXXX

This annex will document supportive material