**3GPP TSG-SA3 Meeting #115AdHoc-e *draft\_S3-241257-r1***

Electronic meeting, online, 15 - 19 April 2024

**Source: Cisco Systems, Google**

**Title: Solution proposal for ACME challenge validation**

**Document for: Approval**

**Agenda Item: 5.4**

# 1 Decision/action requested

***Approval of a new solution to address key issues in TR 33.776.***

# 2 References

[1] [TR 33.776, v0.1.0](https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=4238), "Study of Automated Certificate Management Environment (ACME) for the Service Based Architecture (SBA)"

[2] [TS 33.310](https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2293), "Network Domain Security (NDS); Authentication Framework (AF)"

[3] [IETF RFC 8555](https://datatracker.ietf.org/doc/html/rfc8555): "Automatic Certificate Management Environment (ACME)"

[4] [IETF RFC 9447](https://datatracker.ietf.org/doc/html/rfc9447), "Automated Certificate Management Environment (ACME) Challenges Using an Authority Token"

[5] [IETF RFC 9448](https://datatracker.ietf.org/doc/html/rfc9448), "TNAuthList Profile of Automated Certificate Management Environment (ACME) Authority Token"

[6] [TS 23.502](https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3145), "Procedures for the 5G System (5GS)"

[7] [IETF RFC 7519](https://datatracker.ietf.org/doc/html/rfc7519), " JSON Web Token (JWT)"

[8] [TS 29.571](https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3347), "5G System; Common Data Types for Service Based Interfaces; Stage 3"

[9] [IETF RFC 9110](https://datatracker.ietf.org/doc/html/rfc9110), "HTTP Semantics"

[10] [IETF RFC 7515](https://datatracker.ietf.org/doc/html/rfc7515), "JSON Web Signature (JWS)"

# 3 Rationale

This contribution proposed a solution that addresses key issues identified in TR 33.776 [1], including Key Issue #1 - ACME initial trust framework and Key Issue #3 - Aspects of challenge validation.

# 4 Detailed proposal

Approve the following changes to TR 33.776 [1].

\*\*\* Start of changes (all new text) \*\*\*

# 6 Solutions

## 6.Y Solution #Y: Automated validation of certificate signing requests for network functions

### 6.Y.1 Introduction

This contribution proposed a solution that addresses the following key issues:

- Key Issue #1 - ACME initial trust framework, and

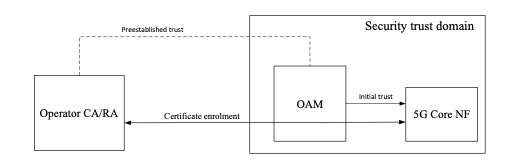
- Key Issue #3 - Aspects of challenge validation.

### 6.Y.2 Solution details

This solution enables a 5GC network function (NF) to use ACME [3] to obtain certificates it can use to establish secure connections within the Service Based Architecture (SBA).

#### 6.Y.2.1 Initial trust

Automated certificate management using ACME reuses the initial trust schema defined in TS 33.310 [2].



The Operations, Administration and Maintenance (OAM) system instantiates the NF, providing it with the initial trust needed for certificate enrollment with the operator CA/RA. The NF instance ID, which uniquely identifies the NF within the 5GC, is assigned to the NF by the OAM system as part of its NF profile, as specified in section 4.17 of TS 23.502 [6].

Initial trust for certificate management of 5GC NFs may be provided using any of the following:

a) OAM issued certificate,

b) Initial Authentication Key (IAK), or

c) OAM issued signature of certain NF profile parameters, at least including the NF instance ID.

When using ACME, option (c) is used. The NF acts as the ACME client, the Operator CA/RA acts as the ACME server, and the OAM system acts as a Token Authority.

An ACME client authenticates to the ACME server by means of an "account key pair", as defined in [3]. The client uses the private key of this key pair to sign all messages sent to the server. The server uses the public key to verify the authenticity and integrity of messages from the client. The NF can generate its own private/public key combination for use as an ACME client account key. Alternatively, this can be assigned by the OAM system.

The ACME challenge-type used is the ACME Authority Token Challenge type, "tkauth-01", as specified in RFC 9447 [4]. The architecture associated with this challenge-type assumes a trust relationship between a CA and a Token Authority, i.e., that a CA is willing to accept the attestation of a Token Authority for particular types of identifiers as sufficient proof to issue a credential. When using ACME, the OAM system acts as a Token Authority that is trusted by the Operator CA/RA. As such, the OAM is trusted to act as the authority for the NF Instance ID namespace within the 5GC.

#### 6.Y.2.2 New identifier type

A new ACME identifier type, "nf-instance-id", is defined in this document. A NF uses its NF instance ID as the value of the “nf-instance-id". The format of the value of the "nf-instance-id" is defined to match that of the NfInstanceId, as defined in TS 29.571 [8]:

- NfInstanceId: string: String uniquely identifying a NF instance. The format of the NF Instance ID shall be a Universally Unique Identifier (UUID) version 4, as described in [RFC 4122](https://datatracker.ietf.org/doc/html/rfc4122). The hexadecimal letters should be formatted as lower-case characters by the sender, and they shall be handled as case-insensitive by the receiver.

- Example: "4ace9d34-2c69-4f99-92d5-a73a3fe8e23b"

An example of an ACME order object "identifiers" field containing a "nf-instance-id" is as follows:

- "identifiers": [{"type":"nf-instance-id","value":"4ace9d34-2c69-4f99-92d5-a73a3fe8e23b"}]

This new ACME identifier type needs to be listed in a new registration in the ACME Validation Methods registry maintained by IANA, per RFC 9447, section 3.

#### 6.Y.2.3 Certificate issuance

The NF begins the certificate issuance process by sending a POST request to the CA's newOrder resource. The NF demonstrates control of its NF instance ID by including its signed NfInstanceId, as provided by the OAM, in the ACME challenge response.

In NF certificates, both client and server, subjectAltName contains the NfInstanceId as a "uniformResourceIdentifier" formatted as a URN as described in clause 5.3.2 of TS 29.571. For example, "urn:uuid: 4ace9d34-2c69-4f99-92d5-a73a3fe8e23b" is the string representation of the NF Instance ID "4ace9d34-2c69-4f99-92d5-a73a3fe8e23b" as a URN.

A full ACME new-order request would look as follows:

POST /acme/new-order HTTP/1.1

Host: example.com

Content-Type: application/jose+json

{

"protected": base64url({

"alg": "ES256",

"kid": "https://example.com/acme/acct/evOfKhNU60wg",

"nonce": "5XJ1L3lEkMG7tR6pA00clA",

"url": "https://example.com/acme/new-order"

}),

"payload": base64url({

"identifiers": [{"type":"nf-instance-id","value":"4ace9d34-2c69-4f99-92d5-a73a3fe8e23b"}],

"notBefore": "2024-05-01T00:00:00Z",

"notAfter": "2024-05-08T00:00:00Z"

}),

"signature": "H6ZXtGjTZyUnPeKn...wEA4TklBdh3e454g"

}

On receiving a valid new-order request, the CA's ACME server creates an authorization object, per RFC8555, Section 7.1.4 [3], containing the challenge that the NF's ACME client must satisfy to demonstrate authority for the identifiers specified by the new order (in this case, the "nf-instance-id"). The CA adds the authorization object URL to the "authorizations" field of the order object and returns the order object to the NF in the body of a 201 (Created) response.

HTTP/1.1 201 Created

Content-Type: application/json

Replay-Nonce: MYAuvOpaoIiywTezizk5vw

Location: https://example.com/acme/order/1234

{

"status": "pending",

"expires": "2024-05-08T00:00:00Z",

"notBefore": "2024-05-01T00:00:00Z",

"notAfter": "2024-05-08T00:00:00Z",

"identifiers": [{"type":"nf-instance-id","value":"4ace9d34-2c69-4f99-92d5-a73a3fe8e23b"}],

"authorizations": [

"https://example.com/acme/authz/1234"

],

"finalize": "https://example.com/acme/order/1234/finalize"

}

On receiving the new-order response, the NF queries the referenced authorization object to obtain the challenges for the identifier contained in the new-order request, as shown in the following example request and response.

POST /acme/authz/1234 HTTP/1.1

Host: example.com

Content-Type: application/jose+json

{

"protected": base64url({

"alg": "ES256",

"kid": " https://example.com/acme/acct/evOfKhNU60wg",

"nonce": "uQpSjlRb4vQVCjVYAyyUWg",

"url": "https://example.com/acme/authz/1234"

}),

"payload": "",

"signature": "nuSDISbWG8mMgE7H...QyVUL68yzf3Zawps"

}

HTTP/1.1 200 OK

Content-Type: application/json

Link: <https://example.com/acme/some-directory>;rel="index"

{

"status": "pending",

"expires": "2024-05-08T00:00:00Z",

"identifier": {

"type":"nf-instance-id",

"value":"4ace9d34-2c69-4f99-92d5-a73a3fe8e23b"

},

"challenges": [

{

"type": "tkauth-01",

"tkauth-type": "atc",

"token-authority": "https://authority.example.org",

"url": "https://example.com/acme/chall/prV\_B7yEyA4",

"token": "IlirfxKKXAsHtmzK29Pj8A"

}

]

}

When processing a certificate order containing an identifier of type "nf-instance-id", a CA uses the Authority Token challenge type of "tkauth-01" with a "tkauth-type" of "atc", as defined in RFC 9447, to verify that the requesting ACME client has authenticated and authorized control over the requested resources represented by the "nf-instance-id" value.

The NF's ACME client responds to the challenge by posting the Authority Token, as received from the OAM system, to the challenge URL identified in the returned ACME authorization object, an example of which follows:

POST /acme/chall/prV\_B7yEyA4 HTTP/1.1

Host: boulder.example.com

Content-Type: application/jose+json

{

"protected": base64url({

"alg": "ES256",

"kid": "https://example.com/acme/acct/evOfKhNU60wg",

"nonce": "Q\_s3MWoqT05TrdkM2MTDcw",

"url": "https://boulder.example.com/acme/authz/asdf/0"

}),

"payload": base64url({

"tkauth": "DGyRejmCefe7v4N...vb29HhjjLPSggwiE"

}),

"signature": "9cbg5JO1Gf5YLjjz...SpkUfcdPai9uVYYQ"

}

The "tkauth" field is, as defined in RFC 9448 [5], a field in the challenge object specific to the tkauth-01 challenge type that should contain an Authority Token as defined in the next section.

#### 6.Y.2.4 NFInstanceId Authority Token

A new Authority Token profile, NFInstanceId Authority Token, is defined in this document. The NFInstanceId Authority Token is a profile instance of the ACME Authority Token defined in RFC9447.

The NFInstanceId Authority Token protected header meets the requirements for "Request Authentication", as specified in RFC 8555, Section 6.2.

The NFInstanceId Authority Token payload includes the mandatory claims "exp", "jti", and "atc":

- "exp" claim, defined in RFC7519, [Section 4.1.4](https://rfc-editor.org/rfc/rfc7519#section-4.1.4) [7], is included and contains the DateTime value of the date and time that the NFInstanceId Authority Token expires.

- "jti" claim, defined in [RFC7519](https://datatracker.ietf.org/doc/html/rfc7519), [Section 4.1.7](https://rfc-editor.org/rfc/rfc7519#section-4.1.7), is included and contains a unique identifier for this NFInstanceId Authority Token transaction.

- "atc" claim, defined in RFC 9447, is included and contains a JSON object with the following elements:

- "tktype" key with a string value equal to "NFInstanceId" to represent a NFInstanceId profile of the Authority Token defined by this document.

- "tkvalue" key with a string value equal to value of the "nf-instance-id".

- "fingerprint" key constructed as defined in RFC8555, [Section 8.1](https://rfc-editor.org/rfc/rfc8555#section-8.1), corresponding to the computation of the "Thumbprint" step using the ACME account key credentials.

An example of the NFInstanceId Authority Token is as follows:

{

"protected": base64url({

"typ":"JWT",

"alg":"ES256",

"x5u":"https://authority.example.org/cert"

}),

"payload": base64url({

"exp":1640995200,

"jti":"id6098364921",

"atc":{"tktype":"NFInstanceId",

"tkvalue":"4ace9d34-2c69-4f99-92d5-a73a3fe8e23b",

"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"}

}),

"signature": "9cbg5JO1Gf5YLjjz...SpkUfcdPai9uVYYQ"

}

The Authority Token is acquired by the NF using a RESTful HTTP POST transaction as follows:

POST /at/account/:id/token HTTP/1.1

Host: authority.example.org

Content-Type: application/json

The request includes the account identifier as a string in the request parameter "id". This string is managed as an identifier specific to the Token Authority's relationship with an operator CA.

The body of the POST request contains a JSON object with key value pairs corresponding to values that are requested as the content of the claims in the issued token. An example is follows:

{

"tktype":"NFInstanceId",

"tkvalue":"4ace9d34-2c69-4f99-92d5-a73a3fe8e23b",

"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"

}

If successful, the response to the POST request returns a 200 (OK) with a JSON body that contains, at a minimum, the NFInstanceId Authority Token as a JSON object with a key of "token" and the base64url-encoded string representing the atc token. An example of a successful response is as follows:

HTTP/1.1 200 OK

Content-Type: application/json

{"token": "DGyRejmCefe7v4N...vb29HhjjLPSggwiE"}

If the request is not successful, the response indicates the error condition. Specifically, for the case that the authorization credentials are invalid or if the account identifier provided does not exist, the response code 403 (Forbidden) is returned. Other 4xx and 5xx responses follow standard HTTP error condition conventions [9].

When creating the NFInstanceId Authority Token, the Token Authority validates that the information contained in the NFInstanceId accurately represents the NF instance id the requesting party is authorized to represent based on their pre-established, verified, and secure relationship. Note that the fingerprint in the token request is not meant to be verified by the Token Authority but rather is meant to be signed as part of the token so that the party that requests the token can, as part of the challenge response, allow the ACME server to validate that the token requested and used came from the same party that controls the ACME client.

#### 6.Y.2.5 Validation of NFInstanceId Authority Token

Upon receiving a response to the challenge, the Operator CA's ACME server performs the following steps to determine the validity of the response.

- Verify that the value of the "atc" claim is a well-formed JSON object containing the mandatory key values.

- If there is an "x5u" parameter, verify the "x5u" parameter is an HTTPS URL with a reference to a certificate representing the trusted issuer of Authority Tokens for the ecosystem.

- If there is an "x5c" parameter, verify the certificate array contains a certificate representing the trusted issuer of Authority Tokens for the ecosystem.

- Verify the NFInstanceId Authority Token signature using the public key of the certificate referenced by the token's "x5u" or "x5c" parameter.

- Verify that "atc" claim contains a "tktype" identifier with the value "NFInstanceId".

- Verify that the "atc" claim "tkvalue" identifier contains the "nf-instance-id" value as the identifier specified in the original challenge.

- Verify that the "atc" claim "fingerprint" is valid and matches the account key of the client making the request.

- Verify that the remaining claims are valid (e.g., verify that token has not expired).

#### 6.Y.2.6 Use of JSON Web Signature

JSON Web Signature (JWS) [10] objects can include an "x5u" header parameter to refer to a certificate that is used to validate the JWS signature. The URLs used in "x5u" are expected to provide the required certificate in response to a GET request, not a POST-as-GET, as required for the "certificate" URL in the ACME order object. This generally requires the ACME client to download the certificate and host it on a public URL to make it accessible to relying parties. RFC 9448, Section 7, defines an optional mechanism for the certification authority (CA) to host the certificate directly and provide a URL that the ACME client owner can directly reference in the "x5u" of their signed nf-instance-id.

The following is an example of the use of "x5u" in the response when the certificate status is "valid".

HTTP/1.1 200 OK

Content-Type: application/json

Replay-Nonce: CGf81JWBsq8QyIgPCi9Q9X

Link: <https://example.com/acme/directory>;rel="index"

Location: https://example.com/acme/order/TOlocE8rfgo

{

"status": "valid",

"expires": "2024-05-20T14:09:07.99Z",

"notBefore": "2024-05-01T00:00:00Z",

"notAfter": "2024-05-08T00:00:00Z",

"identifiers": [

"type":"nf-instance-id",

"value":"4ace9d34-2c69-4f99-92d5-a73a3fe8e23b"

],

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

"finalize": "https://example.com/acme/order/TOlocE8rfgo/finalize",

"certificate": "https://example.com/acme/cert/mAt3xBGaobw",

"x5u": "https://example.com/cert-repo/giJI53km23.pem"

}

### 6.Y.3 Evaluation

Editor’s Note: Evaluation of this solution if FFS.

\*\*\* End of changes \*\*\*