**3GPP TSG-SA3 Meeting #116S3-242514**

**Jeju, South Korea, 20 – 24 May 2024**

**Source: Google, John Hopkins University APL**

**Title: Proposed solution for Certificate enrolment, Challenge validation and Certificate renewal**

**Document for: Approval**

**Agenda Item: 5.4**

# 1 Decision/action requested

***Approval of a new solution to address KI#3, KI#4 and KI#5 in TR 33.776.***

# 2 References

[1] 3GPP TR 33.776: “Study of ACME for Automated Certificate Management in SBA”

[2] 3GPP TS 33.310: “Network Domain Security (NDS); Authentication Framework (AF)”

[3] IETF RFC 8555: “Automatic Certificate Management Environment (ACME)”, 2019

[4] IETF RFC 9448: "TNAuthList Profile of Automated Certificate Management Environment (ACME) Authority Token", 2023

[5] IETF RFC 8739: “Support for Short-Term, Automatically Renewed (STAR) Certificates in the Automated Certificate Management Environment (ACME)”, 2020

[6] IETF RFC 8259: “The JavaScript Object Notation (JSON) Data Interchange Format”, 2017

[7] IETF RFC 2818: “HTTP Over TLS”, 2000

[8] IETF RFC 3629: “UTF-8, a transformation format of ISO 10646”, 2003

[9] IETF RFC 7515: “JSON Web Signature (JWS)”, 2015

[10] IETF RFC 7525: “Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)”, 2015

[11] IETF RFC 8737: “ Automated Certificate Management Environment (ACME) TLS Application‑Layer Protocol Negotiation (ALPN) Challenge Extension”, 2020

[12] IETF RFC 7301: “Transport Layer Security (TLS) Application-Layer Protocol Negotiation Extension”, 2014

# 3 Rationale

The certificate management process involves multiple events within a certificate management lifecycle, such as certificate request and generation, certificated enrolment, and validation of certificates and participating entities. As the 5G system enhances the security infrastructure with automated digital certificate issuance methods and seeks appropriate solutions for automated certificate lifecycles in the 5G SBA, ACME is among the best suited options.

# 4 Detailed proposal

\*\*\* BEGINNING OF CHANGES (all text new) \*\*\*

## 6.x Solution #x: Using ACME protocol for certificate enrolment, renewal and challenge validation

### 6.x.1 Introduction

Considering virtualization in 5G Service-Based Architecture (SBA) and the measure by which the services can expand, it is impractical to manage certificates manually. If there is no standardized use of an automated certificate management protocol, the certificate management needs to be done manually, which may lead to errors such as missing rotation of certificates and attempted use of expired certificates. The significance can increase further given that the 5G core operates on a cloud-native architecture, where elasticity is prominent. On the other hand, automated certificate management promotes scalability by handling a high volume of certificates without requiring extensive manual intervention, which includes certificate enrolment, challenge validation and certificate renewal procedures. Thus, having a modern standardized automated certificate management protocol for 5G SBA will ensure secure interoperability among systems and accessibility of trusted valid certificates at all times.

This solution proposes to use the ACME protocol to address the requirements in key issues, KI#3, KI#4, and KI #5 of the study, TR 33.776 [1].

ACME provides the following benefits:

- Enabling automatic acquisition and management of certificates

- Simplifying deployment of TLS across devices

- Cryptographic key material can be changed automatically, replacement keys can be rolled out automatically

- Reducing chance of global outages

- ACME reduces the chance of the environment becoming calcified on a single certificate on accident because they change regularly

- It makes monitoring certificate lifecycles easier because changes happen on a predictable basis, which enables monitoring and proactive issue resolution

- Enabling quicker responses to global outages when they happen

- Reduces the chance that a vendor will divest itself of the obligation for the CNF’s implementation’s requirement to manage its own certificates in favor of an “optional” certificate life cycle management operational support system

- Reducing the need for third party vendor ‘plugins’ to actively manage the configuration files on a running CNF

- Reduces the chance that third party management of certificates violates a containers’ immutability property, which in an increasingly cloud native and containerized environment is a pillar of integrity

- Removes vendor lock in on certificate issuance enabling migration from one solution/vendor without code changes

- Using certificates uniformly across this ecosystem helps secure the network from tampering and other malicious acts

- Usable with public trust anchors or in a private PKI

### 6.x.2 Solution details

#### 6.x.2.1 Solution overview

ACME can be implemented in multiple ways depending on the trust model network participants wish to use and the aspects of the 5G infrastructure that use ACME based certificates. Additionally, IETF RFC 8555 [3] is extensible to many use cases [4].

6.x.2.1.2 Simplified Architecture

Figure 6.x.2.1.2-1 below illustrates the simplified architecture of the framework. For instance in a 5G SBA setting, a 5G core network function can act as an ACME client. On the other hand, an integrated certificate authority and registration authority entity can be an ACME server. The proposed grouping does not preclude that the functions are collocated or fully integrated in the CA/RA. This will depend on the final solutions. But in its simplest form, an NF requests for certificates enrolment and the CA server enrolls the client identifiers and securely delivers a certificate to the client/NF and this is conducted in a fully automated fashion.

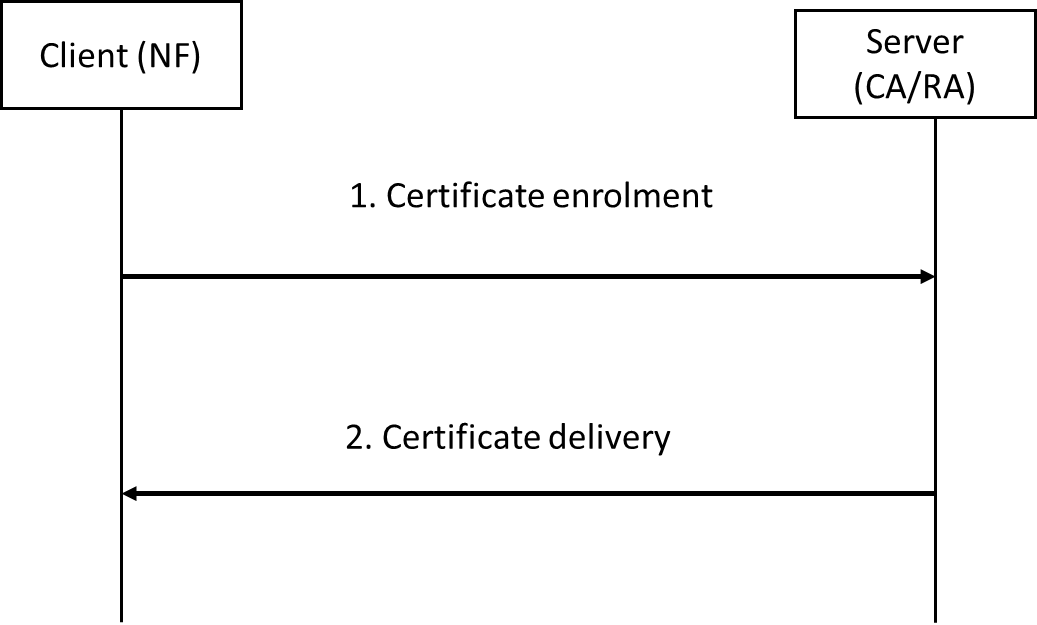


Figure 6.x.2.1.2-1 Simplified ACME Architecture

#### 

#### 6.x.2.2 ACME Profiling for SBA

The following ACME procedures are in the scope of certificate management for SBA:

- Certificate Enrolment: Register an ACME account using an ACME client, prepare and send a request for a certificate, and solve the validation challenge to confirm control of the target(s), then the certificates will be issued. Robust solutions would include automated challenge solving.

- Certificate Renewal: ACME best practice is that at 2/3rds of the certificate lifetime, the certificate will be renewed. If automation is used, no manual work is required. Where appropriate, support for RFC 8739 [5], Short-Term, Automatically Renewed (STAR) Certificates is desirable.

NOTE: The support of the SBA certificate profile as specified in TS 33.310 [2] by ACME has not been addressed in the present document.

##### 6.x.2.2.1 General Requirements

The following requirements apply to ACME usage in Service Based Architecture:

- ACME provides full certificate lifecycle functionality, but it needs appropriate challenge validation solving architecture.

- ACME allows a client to request certificate management actions using a set of JavaScript Object Notation (JSON) messages [6] carried over HTTPS [7].

- Certification authorities (CAs) in the 5G SBA PKI are trusted to verify that an applicant for a certificate legitimately represents the domain name(s) in the certificate.

- The original use case for ACME was obtaining certificates for websites (HTTPS). In this case, a web server is intended to speak for one or more domains, and the process of certificate issuance is intended to verify that this web server actually speaks for the domain(s). ACME has been extended to provide certificate automation for other use cases. E.g. the STIR/SHAKEN PKI.

- All requests and responses sent via HTTP by ACME clients, ACME servers, and validation servers as well as any inputs for digest computations MUST be encoded using the UTF-8 character set, as defined inRFC3629 [8]

- Communications between an ACME client and an ACME server are done over HTTPS, using JSON Web Signature (JWS) [9]

- Each ACME function is accomplished by the client sending a sequence of HTTPS requests to the server, carrying JSON messages. Use of HTTPS is required.

- ACME servers SHOULD follow the recommendations of RFC 7525 [10] when configuring their TLS implementations.

##### 6.x.2.2.2 Profile for PKI Fields

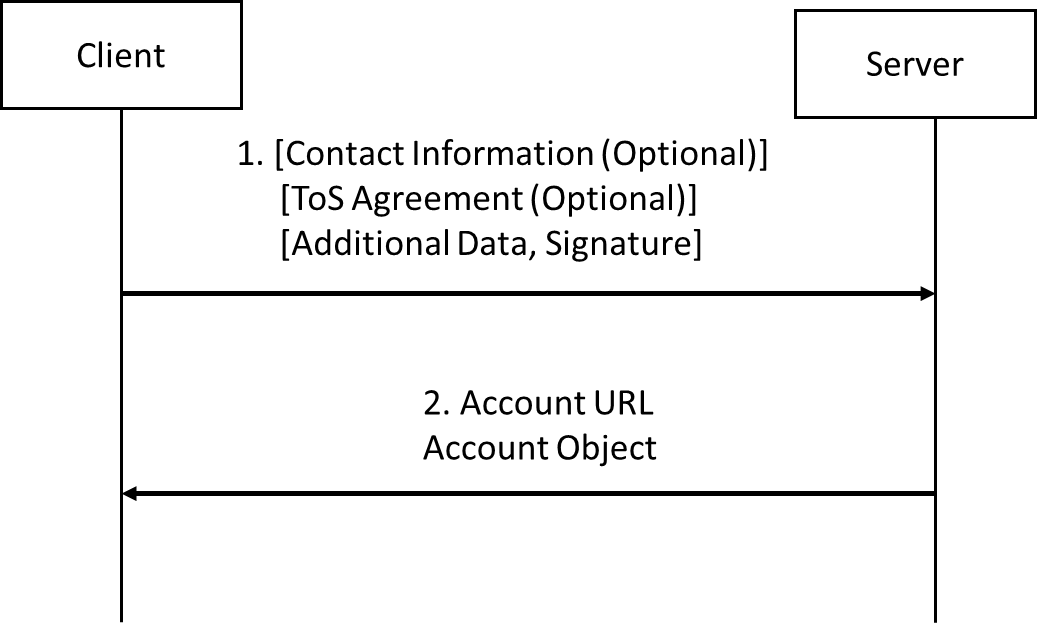
###### 6.x.2.2.2.1 General

The NF Instance certificate enrolment supports the following ACME PKI functions:

* newAccount :
  + A client creates a new account with the server by sending a POST request to the server's newAccount URL
* newOrder:
  + If the client wishes to obtain a renewed certificate, the client initiates a new order process to request one.
* newNonce
  + Generally, before sending a request to the server, an ACME client needs to have a fresh anti-replay nonce.
* newAuthz
  + This is a special field used only for pre-authorization. It may be used if a server wished to enable clients to obtain authorization for identifiers proactively.
* keyChange
  + To change the key associated with an account, the client sends a request to the server containing signatures by both the old and new keys.

###### Editor’s Note: How ACME PKI fields should be aligned to map with SBA.x.2.2.2.2 Account Initialization and Response

The normal ACME flow for ACME account setup below would be adapted. The following describes the first phase of the ACME exchange wherein a Client requests an account with the ACME server.



**Figure 6.x.2.2.2.2-1 – Account Initiation and Response**

In Step 1, the ACME client generates a key pair or alternatively an OAM can generate key pair for the client. ACME client sends to the ACME Server a newAccount request for a new account signed with the generated private key The ACME client may optionally provide contact information, Terms of Service (ToS) and other data. The creation request is signed by the client using its private key.

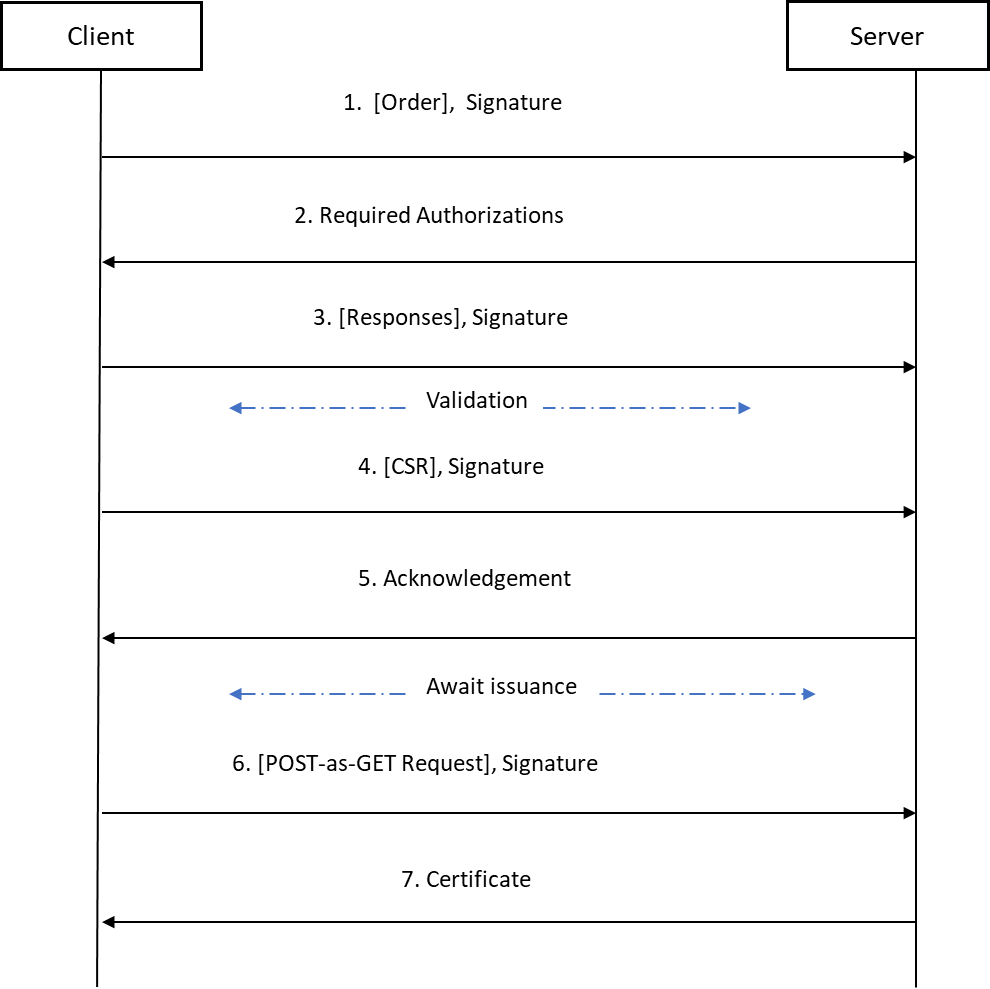
In Step 2, the ACME server validates the client request, creates the Account and sends a response. The response contains the Account URL and the Account Object. The account object contains the account resource information. This includes a valid status of the account, the account order, which basically is a URL from which a list of orders submitted by this account can be fetched and other optional information such as ToS agreement.

Once an ACME client account is registered, there are four major steps the client needs to take to get a certificate:

1. Submit an order for a certificate to be issued
2. Prove control of any identifiers requested in the certificate
3. Finalize the order by submitting a Certificate Signing Request
4. Await issuance and download the issued certificate

Editor’s Note: Account initialization and response procedure should align with SBA procedure.

6.x.2.2.2.3 Certificate enrolment request



**Figure 6.x.2.2.2.3-1 – ACME certificate enrolment**

1. The ACME client function requests a certificate by sending a new order to the newOrder PKI server function using an HTTP POST request. The request contains the algorithm type, indication for a new order, a nonce to protect against replay attacks, and the body. The body of the POST request contains an array of identifiers that the client wants to submit an order for. The JSON payload contains the client identifier type and client identifier value, where the type is ‘dns’, meaning an FQDN and value is the dns name, e.g, [www.example.com](http://www.example.com). The request message is signed using the ACME client’s private key.
2. If the request is valid and the server is willing to issue the requested certificate, it responds with a 201 (Created) response back to the client indicating that the request has been fulfilled with further actions. The body of this response contains the request information, the status, as well as an array of authorization objects which must be satisfied before issuance.
3. The client will check the authorization objects within an order and complete any that are in the “pending” state by solving the listed challenges. When the client is ready for the server to validate the challenge, they respond to the challenge with a POST-as-GET which is signed by the account key.
4. In this step the client sends a Certificate Signing Request (CSR) to the server. After all challenges are validated and the authorizations are in the “valid” state, the order is “ready” and the client will send a Certificate Signing Request (CSR) via a POST request to the finalized URL listed in the order object.

The encoded CSR shall indicate the exact same set of requested identifiers as the initial new-order request.

1. If the call to finalize the order is valid, the server will issue the certificate and publish it at a certificate URL provided in the order object.
2. To download the certificate, the client simply sends a POST-as-GET request to the certificate URL provided in the order.

Editor’s Note: ACME certificate enrolment procedure should align with SBA procedure.

###### 6.x.2.2.2.4 Challenge Validation

There are two objectives for ACME server to create challenges for the client. First, to test whether the client entity holds the private key of the account key pair. The second, to see if the client entity has control over its identifier. The server presents a set of challenges in the authorization object it sends to a client. An example of a valid authorization for [www.example.com](http://www.example.com) that was achieved by solving an ‘http-01’ type challenge is depicted below.

{

"status": "pending",

"expires": "2015-03-01T14:09:07.99Z",

"identifier": {

"type": "dns",

"value": "[www.example.org](http://www.example.org)"

},

"challenges": [

{

"url": "<https://example.com/acme/chall/prV_B7yEyA4>",

"type": "http-01",

"status": "valid",

"token": "DGyRejmCefe7v4NfDGDKfA",

"validated": "2014-12-01T12:05:58.16Z"

}

],

"wildcard": false

}

As the server sends the challenge the client needs to provision the required challenge response based on the challenge type and indicate to the server that it is ready for the challenge validation to be attempted. The client does that by sending a suitable JSON response. In particular, the client sends an empty JSON body in a POST request to the Challenge URL received in the challenge. Before sending a POST request to the server, an ACME client needs to have a fresh anti-replay nonce to put in the "nonce" header of the JWS. The details for various challenge types are described in detail as follows.

For this solution, three challenge types are proposed to validate control over a domain name if used as an identifier. Challenge types to support other NF identifiers in the 5GC SBA are out of scope of this solution.

IETF RFC 8555 [3] section 8 covers the standard challenge validation methods using DNS and HTTP. The third challenge type proposed in this solution is the TLS-ALPN extension defined in RFC 8737 [11].

* DNS Challenge Type: The ACME client responds to a DNS type challenge request by creating a DNS TXT resource record containing a nonce value provided by the ACME server with the ACME client’s account key. (Note: How the TXT resource record gets into the DNS for the domain name in the 5GC SBA is up to operator’s implementation.) After the DNS has been updated with the TXT resource record containing the nonce, the ACME server queries for the TXT records and validates that the nonce returned matches the one provided by the ACME server to validate that ACME client generated the resource record and has control over the domain name.
* HTTP Challenge Type: The ACME client provides a HTTP resource based on the challenge requested by the ACME server on a server accessible under the domain name. (Note: How the HTTP resource gets provisioned on the server is up to operator’s implementation if the ACME client does not have direct access privileges.) After the HTTP resource has been provisioned, the ACME server verifies that the ACME client generated the resource and has control over the domain by verifying that the resource has been provisioned as expected.
* TLS-ALPN Challenge Type: The ACME client responds to a TLS-ALPN type challenge request by creating a self-signed certificate that contains the domain name being validated and configuring a TLS server (i.e., accessible via the domain name) to respond to specific attempts using the TLS-ALPN extension specified in RFC 7301 [11]. The ACME server validates control of the domain name by verifying specific content embedded in the TLS server certificate (i.e., self-signed by ACME client).

It is worth noting that the scope of this solution is limited to the aforementioned challenge types. However, the choice of which challenge types to offer is a matter of ACME server policy and up to operator’s implementation. Additional challenge types may be defined outside of the WebPKI to reflect the needs of a given environment.

Editor’s Note: ACME challenge validation procedure should align with SBA procedure.

###### 6.x.2.2.2.5 Key Update Request and Key Update Response

A client may want to change the public key associated with the account to recover from a key compromise or proactively mitigate future key compromise. The process is as follows:

1. The client needs to send a keyChange request to the server containing signatures by both the old and new keys. The signature by the new key covers the account URL and the old key. This signifies the request from the new key holder to take over the account from the old key holder [3]. The signature by the old key covers this request and its signature. This indicates the old key holder’s approval to update the key.

The client must construct a keyChange object describing the account to be updated and its account key. Thus shall contain the URL for the account being modified and the old key.

The client then encapsulated the keyChange object in an "inner" JWS, signed with the requested new account key. This "inner" JWS becomes the payload for the "outer" JWS that is the body of the ACME request.

1. On receiving a keyChange request, the server shall perform the following validation of the request from the client in addition to the typical JWS validation as described in RFC 8555 [3].

1. Validate the POST request belongs to a currently active account.

2. Check that the payload of the JWS is a well-formed JWS object (the "inner JWS").

3. Check that the JWS protected header of the inner JWS has a "jwk" field.

4. Check that the inner JWS verifies using the key in its "jwk" field.

5. Check that the payload of the inner JWS is a well-formed keyChange object (as described above).

6. Check that the "url" parameters of the inner and outer JWSs are the same.

7. Check that the "account" field of the keyChange object contains the URL for the account matching the old key (i.e., the key id or "kid" field in the outer JWS).

8. Check that the "oldKey" field of the keyChange object is the same as the account key for the account in question.

9. Check that no account exists whose account key is the same as the key in the "jwk" header parameter of the inner JWS.

1. If the checks pass, then the server updates the corresponding account by replacing the old account key with the new public key and returns status code 200 (OK)

###### 6.x.2.2.2.6 Certificate renewal

If an ACME client wants to renew a certificate, it shall initiate a new order process as described in clause 6.x.2.2.2.1. Support for RFC 8739 [5] may be advantageous to support short lived certificates in some scenarios.

NOTE: Life spans for certificates are not defined by ACME and while typical use is short-lived, the life span of the certificates would be up to implementation.

NOTE: Further details on the ACME profiling will be addressed during normative work.

Editor’s Note: ACME certificate renewal process should align with SBA procedures.

### 6.x.3 Evaluation

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

\*\*\* END OF CHANGES \*\*\*