**3GPP TSG-SA3 Meeting #119Adhoc-e S3-250078-r4**

**Online 13 - 16 January 2025**

**Source: KPN**

**Title: Resolving ENs in Solution #42 of TR 33.713**

**Document for: Approval**

**Agenda Item: 5.9**

# 1 Decision/action requested

***This contribution proposes changes to resolve ENs in solution #42 of TR 33.713.***

# 2 References

[1] 3GPP TR 33.713 Study on security aspects of Ambient Internet of Things (AIoT) services in 5G

# 3 Rationale

This contribution provides an update to the solution to address the following Editor’s Notes.

Editor’s Note: Procedure to update set of nonces is FFS

To address above EN, procedure of the solution has been updated with additional steps performed to update nonce which are created at Authentication Server and sent to the AIoT device where they are stored.

Editor’s Note: Procedure to prevent replay attack is FFS

To prevent replay attacks in above EN, procedure has been modified where the AIoT device and Authentication Server mark the nonce as used upon its usage. Authentication server checks for freshness of Nonce everytime it receives a message.

Editor’s Note: Procedure to perform integrity protection of messages in this solution is FFS

To address integrity protection in above EN, AIoT device computes hash of the message which is then received by Authentication Server. The Authentication server computes hash of the data and checks if it matches the hash of the data sent in the message.

Editor’s Note: How encryption in this solution leads to authentication is FFS

Authentication is achieved in this solution in the form of (implicit) challenge response mechanism where nonces are initially pre-configured and are regularly updated which are then used by AIoT device.

# 4 Detailed proposal

**\*\*\*\*\* START OF 1st CHANGE \*\*\*\*\***

## 6.42 Solution #42: Combined authentication and data protection for Ambient IoT services

### 6.42.1 Introduction

This solution addresses key issue #5 on authentication and key issue #4 on protection of information during AIoT service communication. It combines authentication and data transmission into a single message so that low complexity/ low energy devices are not required to perform multiple exchange of messages in order to first authenticate and then to send a message securely. A successful decryption of the message is an indication of authentication of the device.

### 6.42.2 Solution details



Figure 6.42.2-1 Procedure for combined authentication and data protection

0. The device is pre-provisioned with a device ID, shared symmetric key K\_AIoT and a set of nonces. For each device, Authentication Server stores its device ID and the associated key K\_AIoT and set of nonces.

1. AF to NEF: Request data (device information, [type of data])

Device information contains the information about the devices that needs to provide data to the AF, this could be for instance a device ID or a set of device IDs or a group ID. Optionally, type of data expected from the devices can be part of this request.

The NEF selects an appropriate AIoT function.

2. NEF to AIoT function: Request data (device information, [type of data]).

The AIoT function selects a Reader capable of interacting with the required device(s).

3. AIoT function to Reader: Request data (device information, [type of data]).

4. Reader to AIoT device: Paging ([device information]).

The paging message optionally contains device information indicatng the devices that need to be paged.

5a. AIoT device selects a nonce randomly from the set of provisioned nonces (Nonce1) and derives a symmetric key K\_d using K\_AIoT and Nonce1 as inputs.

5b. AIoT device marks Nonce1 as used and removes it from the set of nonces.

5c. AIoT device encrypts data that needs to be sent to the AF along with device ID using K\_d as the encryption key or compute keyed hash of device ID using K\_d, resulting in Enc\_K\_d (data) and Enc\_K\_d (device ID), respectively.

5d. AIoT device generates a hash of data.

6. AIoT device to Reader: Send\_data (device ID, Enc\_K\_d(data), Enc\_K\_d(device ID), Nonce1)

7. Reader to AIoT function: Send\_data (device ID, Enc\_K\_d(data), Enc\_K\_d(device ID), Nonce1)

AIoT function selects AIoT specific Authentication Server holding the K\_AIoT associated with the device ID.

8. AIoT function to Authentication Server: Decrypt\_data\_request (device ID, Enc\_K\_d(data), Enc\_K\_d(device ID), Nonce1)

9a. Authentication Server obtains K\_AIoT based on received device ID.

9b. Authentication Server derives K\_d using K\_AIoT and Nonce1 as inputs.

NOTE 1: The algorithm used by the Authentication Server to derive K\_d is the same as the one used by the AIoT device to derive K\_d.

9c. Authentication Server decrypts the Enc\_K\_d(device ID), using K\_d, and checks if the decrypted device ID matches the unencrypted device ID received and checks if the Nonce is not used.

If the decrypted device ID matches the received unencrypted device ID, the AIoT device is considered to be authenticated.

If the decrypted device ID doesn’t match the received unencrypted device ID, the AIoT device is considered to be not authenticated. An appropriated error response is provided to the AIoT function.

NOTE 2: The error case where the decrypted device ID doesn’t match the received unencrypted device ID is not specified in detail in this solution.

9d. Authentication Server marks the nonce from the message as used.

9e. Authentication Server creates a new nonce or a set of nonces and stores them along with the device ID.

9f. Authentication Server decrypts Enc\_K\_d(data), using K\_d, resulting in an (unencrypted) data.

9g. Authentication Server computes hash of the data and checks if it matches the hash sent in the message.

10. Authentication Server to AIoT device: Send\_data (Enc\_K\_AIoT(New nonce(s)))

11. AIoT device stores the nonce(s) provided by the Authentication Server.

12. AIoT device to Authentication server: Send\_data (Enc\_K\_AIoT(Ack))

AIoT device acknowledges the successful receipt of new nonce(s) by sending acknowledgement message to authentication server.

13. Authentication Server to AIoT function: Decrypt\_data\_response (authentication\_result, data)

Authentication result is Successful if the match in step 9c is successful, else it is Failed.

Data contains the decrypted data obtained in step 9d.

14. AIoT function to NEF: Send data (data).

Message containing data is sent from AIoT function to NEF, if authentication result is Successful. If authentication result is Failed, an appropriate response is sent to the NEF.

15. NEF to AF: Send data (data).

Message containing data is sent from NEF to AF, if authentication result is Successful. If authentication result is Failed, an appropriate response is sent to the AF.

Editor’s Note: Applicability and addressing Nonce exhaustion attack in this solution is FFS

Editor’s Note: Applicability and addressing (D)DoS attack in this solution is FFS

### 6.42.3 Evaluation

This solution addresses key issue #5 on authentication and key issue #4 on information protection. The solution is applicable when the amount of data that needs to be sent from device to network/application function is small enough to be embedded in a single message. If the acknowledgement from the device to the network is not successfully delivered (in step 12), it can lead to synchronisation issues. Use of cleartext ID in step 8 message creates a privacy vulnerability. This vulnerability can be addressed by using some solutions that protect device ID privacy.

The solution does not involve a handshake to perform the authentication, instead it is based on implicit authentication. In this case, only the authenticated device can successfully encrypt the data and the device ID. Similarly, only the authenticated network can decrypt the message and device ID and verify it. Replay protection is achieved with the help of nonce.

The solution remains valid even if the paging message does not contain any device specific information and provides one way authentication.

Editor’s Note: Further evaluation is FFS

**\*\*\*\*\* END OF 1st CHANGE \*\*\*\*\***