|  |  |
| --- | --- |
| 3GPP TR 33.713 V0.5.0 (2024-11) | |
| Technical Report | |
| 3rd Generation Partnership Project;  Technical Specification Group Services and System Aspects;  Study on Security Aspect of Ambient IoT Services in 5G  (Release 19) | |
|  | |
|  |  |
| The present document has been developed within the 3rd Generation Partnership Project (3GPP TM) and may be further elaborated for the purposes of 3GPP. The present document has not been subject to any approval process by the 3GPPOrganizational Partners and shall not be implemented. This Specification is provided for future development work within 3GPPonly. The Organizational Partners accept no liability for any use of this Specification. Specifications and Reports for implementation of the 3GPP TM system should be obtained via the 3GPP Organizational Partners' Publications Offices. | |

|  |
| --- |
|  |
| ***3GPP***  Postal address  3GPP support office address  650 Route des Lucioles - Sophia Antipolis  Valbonne - FRANCE  Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16  Internet  http://www.3gpp.org |
| ***Copyright Notification***  No part may be reproduced except as authorized by written permission. The copyright and the foregoing restriction extend to reproduction in all media.  © 2024, 3GPP Organizational Partners (ARIB, ATIS, CCSA, ETSI, TSDSI, TTA, TTC).  All rights reserved.  UMTS™ is a Trade Mark of ETSI registered for the benefit of its members  3GPP™ is a Trade Mark of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners LTE™ is a Trade Mark of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners  GSM® and the GSM logo are registered and owned by the GSM Association |

Contents

Foreword 11

Introduction 12

1 Scope 13

2 References 13

3 Definitions of terms, symbols and abbreviations 14

3.1 Terms 14

3.2 Symbols 14

3.3 Abbreviations 14

4 Architecture and Security Assumptions 14

5 Key issues 14

5.1 Key Issue #1: Protection for disabling device operation 14

5.1.1 Key issue details 14

5.1.2 Threats 15

5.1.3 Potential security requirements 15

5.2 Key Issue #2: Authorization for 5G Ambient IoT services 15

5.2.1 Key issue details 15

5.2.2 Security threats 15

5.2.3 Potential security requirements 15

5.3 Key issue #3: Privacy by protecting AIoT device identifiers 15

5.3.1 Key issue details 15

5.3.2 Security Threats 16

5.3.3 Potential security requirements 16

5.4 Key issue #4: Protection of information during AIoT service communication 16

5.4.1 Key issue details 16

5.4.2 Security threats 16

5.4.3 Potential security requirements 16

5.5 Key Issue #5: Authentication in Ambient IoT service 17

5.5.1 Key issue details 17

5.5.2 Threats 17

5.5.3 Potential security requirements 17

5.6 Key issue #6: Exposure of Inventory Device Quantity 17

5.6.1 Key issue details 17

5.6.2 Security threats 17

5.6.3 Potential security requirements 18

5.X Key Issue #X: <Key Issue Name> 18

5.X.1 Key issue details 18

5.X.2 Security threats 18

5.X.3 Potential security requirements 18

6 Solutions 18

6.0 Mapping of solutions to key issues 19

6.1 Solution #1: Ambient IoT device disabling mechanism 19

6.1.1 Introduction 19

6.1.2 Solution details 20

6.1.3 Evaluation 22

6.2 Solution #2:PCF based Service Authorization and Provisioning to UE 22

6.2.1 Introduction 22

6.2.2 Solution details 22

6.2.3 Evaluation 22

6.3 Solution #3: Authorization of Intermediate UE for AIoT services 23

6.3.1 Introduction 23

6.3.2 Solution details 23

6.3.3 Evaluation 24

6.4 Solution #4: Protection for inventory and command procedure 24

6.4.1 Introduction 24

6.4.2 Solution details 25

6.4.2.1 Protection for inventory-only procedure 25

6.4.2.2 Protection for inventory and command procedure 26

6.4.2.3 Auth\_token and XAuth\_token derivation function 26

6.4.3 Evaluation 27

6.5 Solution #5: Disabling and Enabling AIoT Device 27

6.5.1 Introduction 27

6.5.2 Solution details 28

6.5.3 Evaluation 29

6.6 Solution #6: AIoT device authentication 29

6.6.1 Introduction 29

6.6.2 Solution details 29

6.6.3 Evaluation 31

6.7 Solution #7: Lightweight AIoT Authentication solution 31

6.7.1 Introduction 31

6.7.2 Details 31

6.7.3 Evaluation 34

6.8 Solution #8: Mutual authentication for AIoT system 34

6.8.1 Introduction 34

6.8.2 Details 34

6.8.3 Evaluation 36

6.9 Solution #9: Device authentication and data communication security 37

6.9.1 Introduction 37

6.9.2 Solution details 37

6.9.3 Evaluation 39

6.10.1 Introduction 39

6.10.2 Solution details 39

6.10.2.1 UE reader case 39

6.10.2.1.1 Alternative 1 – UE reader granularity 39

6.10.2.1.2 Alternative 2 – AIoT device granularity 41

6.10.2.2 RAN reader case 43

6.10.2.3 Example of usage of authentication method and protocol 45

6.10.3 Evaluation 46

6.11 Solution #11: Authentication and ID Privacy of AIoT devices with USIM on AIoT AS Layer 46

6.11.1 Introduction 46

6.11.2 Solution details 46

6.11.3 Evaluation 48

6.12 Solution #12: Authentication and ID Privacy of AIoT devices with USIM on AIoT Layer 48

6.12.1 Introduction 48

6.12.2 Solution details 49

6.12.3 Evaluation 51

6.13 Solution #13: Authentication and ID privacy of AIoT devices without USIM 51

6.13.1 Introduction 51

6.13.2.1 Solution details 52

6.13.2.2 Handling of Temporary ID mismatch 56

6.13.2.3 Handling of group of devices 59

6.13.2.4 Handling of encryption key mismatch 60

6.13.3 Evaluation 63

6.14 Solution #14: Information protection during AIoT service communication 63

6.14.1 Introduction 63

6.14.2 Solution details 64

6.14.2.1 Inventory Service information protection 64

6.14.2.2 Command Service information protection 64

6.14.3 Evaluation 66

6.15 Solution #15: End-to-end security protection of command procedure 66

6.15.1 Introduction 66

6.15.2 Solution details 67

6.15.3 Evaluation 68

6.16 Solution #16: Disabling operation procedure for Ambient IoT services 68

6.16.1 Introduction 68

6.16.2 Solution details 69

6.16.3 Evaluation 70

6.17 Solution #17: Disabling operation procedure for AIoT services 70

6.17.1 Introduction 70

6.17.2 Solution details 70

6.17.3 Evaluation 72

6.18 Solution #18: Authorization procedure for AF-based intermediate node selection 72

6.18.1 Introduction 72

6.18.2 Solution details 72

6.18.3 Evaluation 73

6.19 Solution #19: Authorization of AIoT capable UE in topology 2 74

6.19.1 Introduction 74

6.19.2 Solution details 74

6.19.3 Evaluation 76

6.20 Solution #20: Lightweight AIOT ID privacy based on hashes 76

6.20.1 Introduction 76

6.20.2 Details 77

6.20.3 Evaluation 79

6.21 Solution #21: Ephemeral AIOT ID security context based on puzzles for privacy 79

6.21.1 Introduction 79

6.21.2 Details 80

6.21.3 Evaluation 82

6.22 Solution #22: Solution for protecting AIoT ID by using temporary ID 83

6.22.1 Introduction 83

6.22.2 Solution details 83

6.22.3 Evaluation 84

6.23 Solution #23: AIoT device ID privacy protection using anonymity key 84

6.23.1 Introduction 84

6.23.2 Solution details 84

6.23.3 Evaluation 86

6.24 Solution #24: temporary ID based AIoT device privacy protection 86

6.24.1 Introduction 86

6.24.2 Solution details 86

6.24.3 Evaluation 87

6.25 Solution #25: Use temporary identifier to protect the privacy of AIoT device identifiers. 87

6.25.1 Introduction 87

6.25.2 Solution details 88

6.25.3 Evaluation 90

6.26 Solution #26: Local generated Temporary ID to provide device privacy 90

6.26.1 Introduction 90

6.26.2 Solution details 91

6.26.2.1 Temporary ID generation. 91

6.26.3 Evaluation 93

6.27 Solution #27: Privacy protection of AIoT device identifier based on a temporary identifier 93

6.27.1 Introduction 93

6.27.2 Solution details 94

6.27.2.1 Procedures 94

6.27.2.2 Generation of a temporary identifier 95

6.27.3 Evaluation 95

6.28 Solution #28: Privacy protection on AIoT device IDs 95

6.28.1 Introduction 95

6.28.2 Details 95

6.28.3 Evaluation 97

6.29 Solution #29: Providing a network-computed AIoT concealed device identifier (AICI) to an AIoT device 98

6.29.1 Introduction 98

6.29.2 Solution details 98

6.29.3 Evaluation 99

6.30 Solution #30: Privacy protection for inventory operation 99

6.30.1 Introduction 99

6.30.2 Solution details 99

6.30.2.1 Inventory procedure with unprotected inventory request parameters 99

6.30.2.2 Inventory procedure with protected inventory request parameters 101

6.30.3 Evaluation 103

6.31 Solution #31: Ambient IoT ID privacy 103

6.31.1 Introduction 103

6.31.2 Solution details 103

6.31.3 Evaluation 104

6.32 Solution #32: Authentication Using L1 Parameter 104

6.32.1 Introduction 104

6.32.2 Solution details 105

6.32.3 Evaluation 106

6.33 Solution #33: L1 Security Key Generation 106

6.33.1 Introduction 106

6.33.2 Solution details 106

6.33.3 Evaluation 108

6.34 Solution #34: PHY key based protecting AIoT device identifiers 108

6.34.1 Introduction 108

6.34.2 Solution details 109

6.34.3 Evaluation 110

6.35 Solution #35: Configurable device/network authentication, data confidentiality, integrity and id privacy protection 110

6.35.1 Introduction 110

6.35.2 Solution details 110

.6.35.3 Evaluation 111

6.36 Solution #36: AIoT Authentication with dedicated AIoT NF 112

6.36.1 Introduction 112

6.26.2 Solution details 112

6.36.3 Evaluation 112

6.37 Solution #37: Mutual Authentication Using AEAD for Inventory and Command case 112

6.37.1 Introduction 112

6.37.2 Solution details 113

6.37.3 Evaluation 114

6.38 Solution #38: Authentication and privacy of AIoT device 114

6.38.1 Introduction 114

6.38.2 Solution details 115

6.38.3 Evaluation 116

6.39 Solution #39: reuse of existing authentication frameworks 116

6.39.1 Introduction 116

6.39.2 Solution details 116

6.39.3 Evaluation 116

6.40 Solution #40: Communication security for reading all information from AIoT device 116

6.40.1 Introduction 116

6.40.2 Solution details 117

6.40.3 Evaluation 117

6.41 Solution #41: Disabling protection for AIoT device 118

6.41.1 Introduction 118

6.41.2 Solution details 118

6.41.2.1 Disable an AIoT device permanently or temporarily 118

6.41.2.2 Enable a temporarily disabled AIoT device 119

6.42 Solution #42: Combined authentication and data protection for Ambient IoT services 120

6.42.1 Introduction 120

6.42.2 Solution details 120

6.42.3 Evaluation 122

6.Y Solution #Y: <Solution Name> 122

6.Y.1 Introduction 122

6.Y.2 Solution details 122

6.Y.3 Evaluation 122

7 Conclusions 122

Annex <X> (informative): Change history 123

# Foreword

This Technical Report has been produced by the 3rd Generation Partnership Project (3GPP).

The contents of the present document are subject to continuing work within the TSG and may change following formal TSG approval. Should the TSG modify the contents of the present document, it will be re-released by the TSG with an identifying change of release date and an increase in version number as follows:

Version x.y.z

where:

x the first digit:

1 presented to TSG for information;

2 presented to TSG for approval;

3 or greater indicates TSG approved document under change control.

y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.

z the third digit is incremented when editorial only changes have been incorporated in the document.

In the present document, modal verbs have the following meanings:

**shall** indicates a mandatory requirement to do something

**shall not** indicates an interdiction (prohibition) to do something

The constructions "shall" and "shall not" are confined to the context of normative provisions, and do not appear in Technical Reports.

The constructions "must" and "must not" are not used as substitutes for "shall" and "shall not". Their use is avoided insofar as possible, and they are not used in a normative context except in a direct citation from an external, referenced, non-3GPP document, or so as to maintain continuity of style when extending or modifying the provisions of such a referenced document.

**should** indicates a recommendation to do something

**should not** indicates a recommendation not to do something

**may** indicates permission to do something

**need not** indicates permission not to do something

The construction "may not" is ambiguous and is not used in normative elements. The unambiguous constructions "might not" or "shall not" are used instead, depending upon the meaning intended.

**can** indicates that something is possible

**cannot** indicates that something is impossible

The constructions "can" and "cannot" are not substitutes for "may" and "need not".

**will** indicates that something is certain or expected to happen as a result of action taken by an agency the behaviour of which is outside the scope of the present document

**will not** indicates that something is certain or expected not to happen as a result of action taken by an agency the behaviour of which is outside the scope of the present document

**might** indicates a likelihood that something will happen as a result of action taken by some agency the behaviour of which is outside the scope of the present document

**might not** indicates a likelihood that something will not happen as a result of action taken by some agency the behaviour of which is outside the scope of the present document

In addition:

**is** (or any other verb in the indicative mood) indicates a statement of fact

**is not** (or any other negative verb in the indicative mood) indicates a statement of fact

The constructions "is" and "is not" do not indicate requirements.

# Introduction

This clause is optional. If it exists, it shall be the second unnumbered clause.

Editor’s Note: This clause contains some background information for the study.

# 1 Scope

The present document identifies potential threats and security requirements to enable AIoT services for various use cases. Consideration for the energy and complexity constraints of AIoT devices is taken into account in identifying and developing potential security mechanisms to support AIoT services. Specifically, the present document focuses on the following:

1. Identify security and privacy and threats introduced by AIoT services for use cases captured in TS 22.369 [2], for topologies captured in RP-234058[3], and for architecture captured in TR 23-700-13[4].

2. Identify security requirements to address the identified threats.

3. Develop potential solutions that fulfil the security requirements, taking into account AIoT device constraints agreed upon in other 3GPP working groups.

NOTE 1: Enable/disable device operation is within the scope of the present document.

# 2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication, edition number, version number, etc.) or non‑specific.

- For a specific reference, subsequent revisions do not apply.

- For a non-specific reference, the latest version applies. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document *in the same Release as the present document*.

[1] 3GPP TR 21.905: "Vocabulary for 3GPP Specifications".

[2] 3GPP TS 22.369: "Service Requirements for ambient power-enabled IoT".

[3] RP-234058, RAN New SID for Study on Solution for Ambient IoT in NR.

[4] 3GPP TR 23-700-13: "Study on Architecture Support of Ambient power-enabled Internet of Things".

[5] 3GPP TS 33.501: "Security Architecture and Procedures for 5G System".

[6] R2-2406202 RAN2#126 Meeting Report

[7] RFC 4739: "Multiple Authentication Exchanges in the Internet Key Exchange (IKEv2) Protocol".

[8] 3GPP TR 38.848: "Technical Specification Group Radio Access Network; Study on Ambient IoT (Internet of Things) in RAN".

[9] 3GPP TR 38.789: “Study on solutions for Ambient IoT (Internet of Things) in NR”

[10] Jiao L, Wang N, Wang P, et al. Physical layer key generation in 5G wireless networks[J]. IEEE wireless communications, 2019, 26(5): 48-54.

[11] Zhao H, Zhang Y, Huang X, et al. A physical-layer key generation approach based on received signal strength in smart homes[J]. IEEE Internet of Things Journal, 2021, 9(7): 4917-4927.

[12] 3GPP TS 33.501: “Security architecture and procedures for 5G System”

# 3 Definitions of terms, symbols and abbreviations

## 3.1 Terms

For the purposes of the present document, the terms given in 3GPP TR 21.905 [1] and the following apply. A term defined in the present document takes precedence over the definition of the same term, if any, in 3GPP TR 21.905 [1].

**example:** text used to clarify abstract rules by applying them literally.

## 3.2 Symbols

For the purposes of the present document, the following symbols apply:

<symbol> <Explanation>

## 3.3 Abbreviations

For the purposes of the present document, the abbreviations given in 3GPP TR 21.905 [1] and the following apply. An abbreviation defined in the present document takes precedence over the definition of the same abbreviation, if any, in 3GPP TR 21.905 [1].

<ABBREVIATION> <Expansion>

# 4 Architecture and Security Assumptions

Editor’s Note: This clause contains security architecture and assumptions to be considered for the study (e.g., per work task/KI).

The following architecture and security assumptions are applied:

* The architecture assumptions and requirements for Ambient IoT services as defined in TR 23.700-13 [4] are used as architecture assumptions in this study.
* Two functional cases are considered as baseline: (1) inventory, (2) command.

# 5 Key issues

Editor’s Note: This clause contains all the key issues identified during the study.

## 5.1 Key Issue #1: Protection for disabling device operation

### 5.1.1 Key issue details

As specified in TS 22.369 [2], the enable/disable device operation is used for the operator to manage the Ambient IoT device, which can enable/disable the Ambient IoT device's capability to transmit RF signals. Based on operator policy, there are two categories of disabling device operations, i.e. permanent disabling of the capability and temporary disabling of the capability.

### 5.1.2 Threats

As a management operation, the availability of Ambient IoT devices will be impacted if the disabling device operation is not securely performed. For example, if the Ambient IoT device follows the spoofed permanent/temporary disable device operation from an attacker, the Ambient IoT devices will not respond to the network either permanently, or for a period of time, leading to the Denial of Service (DOS).

### 5.1.3 Potential security requirements

The means to securely disable the Ambient IoT device(s)’s capability to transmit RF signals shall be supported.

Editor’s Note: Whether the solutions for this key issue are the same or different from those for communication protection issue is FFS.

Editor’s Note: Security solutions for this Key Issue should be aligned with the conclusion of Ambient IoT system architecture in SA2..

## 5.2 Key Issue #2: Authorization for 5G Ambient IoT services

### 5.2.1 Key issue details

In TR 23.700-13 [4], Key Issues #1 and #3 describe the issues on the system architecture and procedure to support 5G Ambient IoT services.

In the Topology 2 as defined in TR 38.848 [8], the UE acting as the intermediate node is responsible for transferring the information between AIoT device and 5GS. If the authorization of intermediate node is not supported, the attacker can play the role of intermediate node and arbitrarily deny 5G AIoT service.

Therefore, it is necessary to study how to authorize the UE for acting as the intermediate node.

### 5.2.2 Security threats

If the 5GC cannot verify if the UE acting as an intermediate node is authorized, the attacker UE may impersonate the intermediate node. The attacker UE may then deny the 5G Ambient IoT services.

### 5.2.3 Potential security requirements

The 5GS shall be able to support the authorization of the AIoT capable UE as an intermediate node in 5G Ambient IoT services.

## 5.3 Key issue #3: Privacy by protecting AIoT device identifiers

### 5.3.1 Key issue details

5G Ambient IoT service is a type of cellular IoT communication system where Ambient IoT devices utilize harvested energy to generate RF signals for bi-directional information transmission. Ambient IoT devices are characterized by limited functions, requiring only small and infrequent data transfers.

TS 22.369 [2] clause 5.2.6 defines the following privacy-related requirements:

“The 5G system shall be able to provide a mechanism to protect the privacy of information (e.g., location and identity) exchanged during communication between an Ambient IoT device and the 5G network or an Ambient IoT capable UE.”

In AIoT services, identifiers of AIoT device are used to identify the device. If the identifiers associated with a device are not privacy protected (e.g., exposed over the air), an attacker (e.g., an over-the-air attacker) can identify and track an AIoT device based on the identifiers associated with the AIoT device. Thus, this key issue is to investigate potential mechanisms to privacy protect the AIoT device identifiers.

### 5.3.2 Security Threats

An attacker can identify, monitor and track an AIoT device based on the identifiers associated with the AIoT device if the identifiers are not privacy protected.

Editor’s Note: It is FFS how the above threat affects various use cases.

Editor’s Note: Security threat and requirement for potential exposure of quantity of devices after adversary broadcasts an inventory message is FFS.

### 5.3.3 Potential security requirements

Mechanisms for mitigating privacy threats (described above) by identifying, linking, and tracking the identifiers of AIoT Device(s) shall be supported.

Editor’s Note: AIoT use cases that do not need the above privacy protection mechanisms are FFS.

## 5.4 Key issue #4: Protection of information during AIoT service communication

### 5.4.1 Key issue details

As per TS 22.369 [2], Ambient power-enabled IoT (AIoT) services aim to support various use cases, including inventory taking, sensor data collection, asset tracking, and actuator control. These services intended to operate with lower power consumption and complexity than the existing IoT technologies such as eMTC, NB-IoT, and RedCap. To fulfil these requirements, AIoT devices require a communication capability.

Considering the ambient IoT device will be deployed in the indoor system, if the restricted access means (e.g., in factory) are provided, the possibility of attack is minimal. However, if the environment (e.g., in shopping mall) is open to the attacker, from a security perspective, security mechanisms to protect the information transmitted during AIoT service communication need to be supported. Failure to provide such security mechanisms will lead to various attacks such as eavesdropping, manipulation and/or unauthorized transmission of the information during AIoT service communication.

### 5.4.2 Security threats

For command operation (e.g., write, read), the following threats are applicable:

An attacker may acquire data transmitted to/from AIoT devices by eavesdropping messages if the communication of AIoT service is not confidentiality protected.

An attacker may manipulate information during communication of AIoT service if the communication of AIoT service is not integrity protected.

An attacker may replay a message if replay protection is not activated.

### 5.4.3 Potential security requirements

The Ambient IoT system shall support a means to ensure confidentiality, integrity and/or anti-replay of information for AIoT services.

Editor’s Note: Whether information protection between AIoT device and 5G core is mandatory is ffs.

Editor’s Note: Whether information protection can be done on application layer is ffs.

## 5.5 Key Issue #5: Authentication in Ambient IoT service

### 5.5.1 Key issue details

The TR 23.700-13 [4] studies the architecture support of Ambient Internet of Things (AIoT) device, considering the service requirements for ambient power-enabled IoT device. In TR 23.700-13 [4], the validation of the AIoT device identity and authentication are explicitly mentioned.

### 5.5.2 Threats

In the air interface, an attacker may impersonate the victim device and report fake identification to the network side. If the billing is based on per AIoT device’s identity, the fake identity may lead to charging problem. This can be used by an adversary to steal an AIoT device by replacing the AIoT device with a fake device, which might cause a loss to the owner of the device. Whether fake identities pose a threat in case of inventory use case, is up to the individual deployments risk profile.

Over the air interface, without authentication, an adversary can impersonate a legitimate network and send a command — e.g., disablement command will permanently deactivate a device.

### 5.5.3 Potential security requirements

The 5G system shall provide a means to perform mutual authentication between the AIoT device and the network.

NOTE: If solution reuses the existing authentication framework (e.g., 5G-AKA, EAP-AKA’, other EAP methods for SNPN), the existing UE security requirements in TS 33.501[5] apply. Otherwise, the core network entities, if any, that are used for authentication are dedicated for Ambient IoT service.

Editor’s Note: Secure storage and processing of credentials (on the AIoT device and network side) for solution not reusing existing authentication framework as per the note above is ffs.

Editor's Note: Whether performing authentication is to be mandated or optional, one way or mutual, or left to the application layer is to be discussed and decided in the conclusion.

## 5.6 Key issue #6: Exposure of Inventory Device Quantity

### 5.6.1 Key issue details

The inventory service is a fundamental process for AIoT devices, which includes both "inventory only" and "inventory and command" cases. In both scenarios, the mandatory steps involve AIoT paging and Device ID transmission. The AIoT paging message may contain an ID of a single A-IoT device, a group ID that maps to multiple A-IoT devices, or multiple IDs of A-IoT devices. If AIoT paging message does not contain an ID, it will map to all the A-IoT devices. After these steps, the network can calculate the quantity of device IDs for this inventory.

The inventory device quantity may contain business information, such as the quantity of stock in a shopping mall. If this information falls into the hands of competitors, they may adjust their sales strategy to attract more customers from that shopping mall.

### 5.6.2 Security threats

By broadcasting a fake inventory message with a group ID, an attacker could potentially calculate the quantity of devices in a group by observing the differences in reported device IDs, even if the IDs are encrypted. This could lead to the exposure of the inventory device quantity associated with the group ID. For example, in a shopping mall, assuming the attacker has knowledge of the link between the group ID and goods (such as knowledge of the link between SUPI and the real subscriber), the attacker could use a fake reader to broadcast this group ID. Subsequently, the attacker would receive multiple device IDs and calculate the device quantity for this group ID, allowing them to determine the number of specific goods.

An attacker could calculate the quantity of all devices by observing differences in reported device IDs, even if the IDs are encrypted, after sending a fake inventory message without any IDs. This could result in the exposure of the inventory device quantity within an area. For instance, in a shopping mall, if an attacker can control the broadcast scope into the shopping mall, they could utilize a fake reader to broadcast an inventory message without any ID. Subsequently, the attacker would receive multiple device IDs, enabling them to calculate the device quantity in this area and determine the stock levels of all the goods of this shopping mall.

### 5.6.3 Potential security requirements

TBA.

## 5.X Key Issue #X: <Key Issue Name>

### 5.X.1 Key issue details

### 5.X.2 Security threats

### 5.X.3 Potential security requirements

# 6 Solutions

Editor’s Note: This clause contains the proposed solutions addressing the identified key issues.

## 6.0 Mapping of solutions to key issues

Table 6.1-1: Mapping of solutions to key issues

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Solutions | KI#1 | KI#2 | KI#3 | KI#4 | KI#5 |
| **1** | X |  |  |  |  |
| **2** |  | X |  |  |  |
| **3** |  | X |  |  |  |
| **4** |  |  |  |  | X |
| **5** | X |  |  |  |  |
| **6** |  |  |  |  | X |
| **7** |  |  |  |  | X |
| **8** |  |  |  |  | X |
| **9** |  |  |  |  | X |
| **10** |  |  |  |  | X |
| **11** |  |  | X | X | X |
| **12** |  |  | X | X | X |
| **13** |  |  | X | X | X |
| **14** |  |  |  | X |  |
| **15** |  |  |  | X |  |
| **16** | X |  |  |  |  |
| **17** | X |  |  |  |  |
| **18** |  | X |  |  |  |
| **19** |  | X |  |  |  |
| **20** |  |  | X |  |  |
| **21** |  |  | X |  |  |
| **22** |  |  | X |  |  |
| **23** |  |  | X |  |  |
| **24** |  |  | X |  |  |
| **25** |  |  | X |  |  |
| **26** |  |  | X |  |  |
| **27** |  |  | X |  |  |
| **28** |  |  | X |  |  |
| **29** |  |  | X |  |  |
| **30** |  |  | X |  |  |
| **31** |  |  | X |  |  |
| **32** |  |  |  | X | X |
| **33** |  |  |  | X |  |
| **34** |  |  | X |  |  |
| **35** |  |  | X | X | X |
| **36** |  |  |  |  | X |
| **37** |  |  |  |  | X |
| **38** |  |  | X |  | X |
| **39** |  |  |  |  | X |
| **40** |  |  |  | X |  |
| **41** | X |  |  |  |  |
| **42** |  |  |  | X | X |
|  |  |  |  |  |  |

Editor’s Note: Each solution should be mapped here.

## 6.1 Solution #1: Ambient IoT device disabling mechanism

### 6.1.1 Introduction

This solution addresses KI#1.

According to TS 22.369, the enable/disable device operations are used by the network operator to manage the Ambient IoT device’s capability to transmit RF signals. As the disabling of RF transmission capability could, according to the operator’s policy, be temporary or permanent, it is paramount to ensure that the disabling, specifically of a permanent nature, is performed securely and in a manner that allows device recovery in case the system was compromised, and an attacker has managed to issue “disable” commands to one or multiple Ambient IoT devices.

### 6.1.2 Solution details

A screenshot of a computer

Description automatically generated

Figure 6.1.1 – Ambient IoT device disabling mechanism

The permanent disabling of an AIoT device is performed in a two-stage operation, where initially, the AIoT device is temporarily disabled, and then, following a cool-down period (i.e., recovery time window), the AIoT device could be disabled permanently. The two-stage permanent disabling operation is performed as follows:

In Step 0, the AIoT device is provisioned with a configuration determining how the device processes disabling operations. The configuration includes the required cool down period that needs to be met before a permanent “disable” command is allowed.

In Step 1, the AIoT managing function issues a temporary disable command to the Ambient IoT device. The command includes a counter T1.

Note 1: The AIoT Management Function (AIoT MF) depends on the entity that owns or manages the resource (i.e., AIoT device). If a device is owned or managed by the network, the disabling is triggered by a CN function, otherwise, it is triggered by a 3rd party Management Function managing the device.

Editor’s Note: Whether the solution aligns with SA2 system architecture and procedures is FFS.

In Step 2, The AIoT device, upon receiving the temporary disable command, retrieves and stores the counter T1, which will be used in subsequent processing.

In Step 3, the AIoT device sends an ACK to the AIoT MF, which may contain the counter received in Step 0. Then, the AIoT device temporarily disables its RF transmission capability..

In Step 4, depending on whether the AIoT MF intends to re-enable the AIoT device (i.e., recovery scenario), or permanently disable the AIoT device. The AIoT MF sends in:

- Step 4a: a message containing an enable command to recover the AIoT device, or

- Step 4b: a message containing a permanent disable command, in which a second counter T2 is included.

Note 2: The messages in step1 and step3 carrying the temporary and permanent disable commands are protected using the same means of protection applicable to other commands (e.g., write), and so is the verification that the disable commands are coming from a legitimate party.

Editor’s Note: How inventory/authentication is performed before the permanent disable command is FFS.

In Step 5, The AIoT device processes the received command. If the AIoT is requested to enable its RF transmission capability, the AIoT device recovers from the temporary disabled state and discards the stored counter T1. Otherwise, if the AIoT device is requested to perform a permanent disable command, it retrieves the second counter from the message, then checks whether the following conditions are met:

- The RF transmission capability is temporarily disabled.

- Whether the value T2 – T1 is greater or equal to the cool-down period configured in the AIoT device.

If the checks succeed, the AIoT device temporarily enables its RF transmission capability to send an ACK message to the AIoT MF in step 6 confirming that the device is permanently disabling its RF transmission capability.

In Step 7, the AIoT device disables its RF transmission capability permanently.

### 6.1.3 Evaluation

To mitigate the threat of unauthorized permanent disabling (e.g., through spoofed commands), the solution involves a controlled, two-stage operation for permanent disabling. By enforcing the cool-down period, the system adds an extra layer of control, allowing time for validation and potential recovery if an unauthorized command is detected.

In the solution proposed, the protection of disable commands (temporary and permanent) re-uses the same means of protection applicable to other commands (e.g., write command).

Ambient IoT device(s) need to maintain an internal state to support the security mechanism proposed by the solution.

Editor’s Note: Further evaluation is FFS.

## 6.2 Solution #2:PCF based Service Authorization and Provisioning to UE

### 6.2.1 Introduction

This solution addresses the KI#2 Authorization for 5G Ambient IoT services.

### 6.2.2 Solution details

This solution propose to reuse the existing mechanism for 5G Prose U2N relay as specified in TS 23.304[x] with following changes:

A UE acting as an intermediate node is registered with 5GC using the existing mechanism, with some enhancements to indicate its capability of acting as an intermediate node regardless of for which AIoT device, and is authorized as an intermediate node (UE) during the registration procedure.

For PCF based Service Authorization and Provisioning to UE, the Registration procedures as defined in clause 4.2.2.2 of TS 23.502 [x], UE Policy Association Establishment procedure as defined in clause 4.16.11 of TS 23.502 [x] and UE Policy Association Modification procedure as defined in clause 4.16.12 of TS 23.502 [x] apply with the following additions:

- If the UE indicates AIoT Capability in the Registration Request message and if the UE is authorized to use 5G AIoT service based on subscription data, the AMF selects the PCF which supports AIoT Policy/Parameter provisioning and establishes a UE policy association with the PCF for AIoT Policy/Parameter delivery.The AMF reports the authorized AIoT Capability to the selected PCF, which may determine the AIoT Policy/Parameter based on the UE's authorized AIoT Capability.

This solution based on the assumption that PCF hold the Authorization information and the AIoT capabiltiy can be include in the Registration Request message.

### 6.2.3 Evaluation

This solution addresses the KI#2, based on the assumption that PCF hold the Authorization information and the AIoT capability can be include in the Registration Request message. The existing Registration procedure is reused.

## 6.3 Solution #3: Authorization of Intermediate UE for AIoT services

### 6.3.1 Introduction

The solution addresses the security requirement of KI#2: Authorization for 5G Ambient IoT services. Specifically, this solution proposes a method to authorize the UE as Intermediate UE in AF-initiated AIoT service procedure. The AMF/AIoT NF select the UE based on the information provided by AF, e.g., location information or external UE ID, and then interact with the UDM to obtain the selected UE’s subscription data and check whether it is allowed to act as Intermediate UE for AIoT secvice. Only after the UE is successfully authorized as intermediate UE, the network will then perform subsequent AIoT service procedure.

### 6.3.2 Solution details

Depicted in Figure 6.3.2-1 is the authorization procedure of Intermediate UE for AIoT Services.



**Figure** **6.3.2-1: Authorization of Intermediate UE for AIoT service**

1. The UE performs the registration procedure as specified in TS 23.502 [x] with the enhancement to indicate its AIoT Intermediate node capability, and is authorized as an intermediate UE during the registration procedure.
2. The AF sends the AIoT Service Request to the AMF/AIoT NF via the NEF, including the AIoT device ID, seivice type (e.g., Inventory, Command), location information, external UE ID (GPSI).
3. The AMF/AIoT NF selects the Intermediate UE based on the information provided by AF, e.g., location information and/or GPSI, etc.

NOTE1: The selection of Intermediate UE is up to SA2 WG decision.

1. The AMF/AIoT NF sends the UE Authorization Request to the UDM with the info of the selected UE.
2. The UDM checks whether the selected UE is allowed to act as Intermediate UE against the UE's subscription data for AIoT service.

NOTE2: The relevant subscription data could be configured offline in the UDM, or provided and updated in the UDM based on the AF-initiated AIoT service requests.

1. The UDM returns the UE Authorization Response to the AMF/AIoT NF.
2. The AMF/AIoT NF sends the AIoT Service Request to the Intermediate UE, including the AIoT device ID, service type, authorized result.
3. The inventory/Command procedure is carried out.

Editor’s Note: Whether the Intermediate UE is authorized during the registration or after the Intermediate UE selection is FFS.

Editor’s Note: Which entity performs the Intermediate UE authorization should be aligned with the AIoT system designed by SA2, which is FFS.

### 6.3.3 Evaluation

TBD

## 6.4 Solution #4: Protection for inventory and command procedure

### 6.4.1 Introduction

This solution addresses key issue on authentication, key issue on information protection and key issue on protection for disabling device operation.

The solution is assumed to be used in dedicated network for ambient IoT service. It is assumed that every AIoT Device is preconfigured with a pre-shared key, while the details of secure storage and processing of credentials e.g. pre-shared key is out of scope of this solution. Accordingly, it is assumed that such a key is also preconfigured/stored in the network side along-side the device ID.

The solution provides a new AIoT device authentication procedure that could be used for both the inventory as well as the command procedures.

### 6.4.2 Solution details

### 6.4.2.1 Protection for inventory-only procedure

Figure 6.4.2-1: Information Flow for protection of inventory-only service



1. AF sends AIoT service operation request to NEF, including the AF ID, device information, and inventory operation.

2. NEF sends the AIoT service operation request to the selected AMF/AIoTMF.

3. The AMF/AIoTMF selects the Ambient IoT capable RANs. In addition, the Nonce1 is generated, which is used as fresh parameter for authentication. For each of the selected reader, the AMF/AIoTMF sends the request message with the generated Nonce1.

4. Upon reception of the request message, Reader executes inventory by triggering the Paging-like procedures with Nonce1 towards the AIoT devices.

5. The AIoT device performs random access like procedures to establish the connection with reader.

6. The AIoT device generates Nonce2 as the fresh parameter from device side. When deriving Auth\_token, Nonce1 and Nonce2 are used to form the input S to the KDF and the pre-shared key is used as input key. Auth\_token is used for network to authenticate device.

7-8. The AIoT Device sends the AIoT Device ID, Nonce2 and Auth\_token to Reader. In the following step, Reader reports the AIoT Device ID, Nonce2 and Auth\_token to AMF/AIoTMF.

AIoT Device uses registers for temporarily keeping the nonces required for command protection.

9. AMF/AIoTMF sends request message to authentication server for device verification, which includes AIoT Device ID, Nonce1 and Nonce2.

10. XAuth\_token is calculated with same input as in device side.

11. Authentication server sends response message to AMF/AIoTMF, which includes XAuth\_token.

12. AMF/AIoTMF compares Auth\_token and XAuth\_token.

NOTE: As an alternative, authentication server can verify the Auth\_token and send the authentication result to AMF/AIoTMF.

13-14. If the verification succeeds, the AMF/AIoTMF reports the AIoT Device ID to NEF by sending the AIoT\_ Notify message. The NEF forwards the received information to AF by sending the AIoT\_ Notify message.

### 6.4.2.2 Protection for inventory and command procedure

Figure 6.4.2-2: Information Flow for protection of information transfer for AIoT services



Compared with the inventory-only procedure in clause 6.4.2.1, the protection for command is enhanced with the following modification.

10. In addition to the XAuth\_token calculation, the session key Ks is derived if receiving additional indication from AIoT Function. The Nonce1 and Nonce2 is used to form the input S to the KDF and the pre-shared key is used as input key to derive the session key Ks. The Ks is different with XAuth\_token as different input (e.g., FC value).

11. Session key is also included. If the device security capability (e.g., AES-128 algorithm) is stored in AIoT\_ UDM, it will also be sent to AMF/AIoTMF.

12. The command message is protected with session key. For example, the MAC1 will be calculated to protect the integrity of the message. The algorithm will be selected based on device security capability.13-14. The AMF/AIoTMF sends the protected AIoT Command message to the AIoT Device, which includes encrypted data and MAC1. No AIoT device ID is required to be included in step 14, as other ID will be used between device and reader, which is discussed and decided in RAN groups. It is assumed that the device receives the command message while energy is available in energy storage.

15-16. The device will receive the message in step 14. Integrity verification and decryption will be performed. After successful verification, the device will follow the command operation. The uplink message will also be sent from device to network if needed. For example, if the command is “read” operation, data2 to be reported will be included after encryption. The message will also be integrity protected.

17-18. The AMF/AIoTMF reports the result of the AIoT service operation request to the NEF by sending the AIoT\_ notification message.

Editor’s Note: The information flow will be updated based on the conclusion regarding command operation procedure in SA2.

### 6.4.2.3 Auth\_token and XAuth\_token derivation function

The following parameters are used to form the input S to the KDF.

- FC = 0xaa,

- P0 = Nonce1,

- L0 = length of the Nonce1(i.e. 0x00 0x10),

- P1 = Nonce2,

- L1 = length of Nonce2 (i.e. 0x00 0x10),

The input key KEY shall be equal to the pre-shared key.

The (X)Auth\_token is identified with the 128 least significant bits of the output of the KDF.

### 6.4.3 Evaluation

Challenge-based symmetric algorithm is used for security protection. The privacy of device identifier in uplink and downlink is not addressed in this solution.

This solution assumes there is preconfigured pre-shared key as the authentication credential stored in AIoT device and network side securely.

For inventory-only case, one-way device authentication is provided. For inventory and command case, mutual authentication is supported, in which the network authentication in device is implicitly provided by MAC verification.

Editor’s Note: the impact to include nonce in paging message is FFS.

Editor’s Note: whether replay attacks are possible against the device or the network is FFS.

Editor’s Note: whether additional authentication token is required for device to authenticate network is FFS.

## 6.5 Solution #5: Disabling and Enabling AIoT Device

### 6.5.1 Introduction

This solution addresses KI#1 to disable or enable the RF transmission capabilities of an AIoT device according to operator’s security policy. Solution makes the following assumptions:

1. Disabling/enabling instruction is to be sent as part of inventory and command procedure if command only procedure is not supported. If command only procedure is supported, the solution can be part of the command only procedure.

2. Since part of the command case is to carry instructions for the AIoT device to perform some specific functions, it is also assumed that the AIoT device and the network has established security to protect the inventory and command procedure and that the command procedure is confidentiality-, integrity-, or replay-protected. The protection of either the inventory or command procedure is not in scope of this solution.

3. For a group of AIoT device, solution assume that the group can be paged (i.e., via inventory procedure) if the paing ID corresponds to a group ID and that disable/enable code is the same for the group.

NOTE 1: While AIoT device’s RF transmission capabilities are disabled, the device is still able to receive RF signals.

NOTE 2: Protection of command procedure is recommended.

### 6.5.2 Solution details



1. Inventory procedure is performed between the AIoT device and the network (CN or AF).

NOTE: Whether Command operation procedure is stand-alone or is coupled with inventory and command is to be aligned with RAN2.

2. Depending on the business model and security protection, the Command Operation Trigger containing Code1 and Code2 can come from third-party AF or within the 3GPP network. If third-party AF initiates the Command Operation Trigger, the request is routed through to the 3GPP core network, in which case, the AF is authorized by the 3GPP network.

3. The AIoT Function/AMF (either initiated by 3GPP network or as a result of request from third-party AF) sends a Command Operation Trigger with Code1 and Code2 to the AIoT Reader (RAN node or an AIoT-capable UE).

4. The AIoT Reader (RAN node or an AIoT-capable UE) sends the Command Operation Request to the AIoT device.

5. The AIoT Device checks Code1 and Code2 against the code stored in the AIoT device, using the following logic:

If Code1 and Code2 are the same and that they match the code stored in the AIoT device, the AIoT device permanently disables its RF transmission capabilities.

If Code1 and Code2 are different, and Code1 matches the code stored in the AIoT device, the AIoT device temporarily disables its RF transmission capabilities if its RF transmission capabilities are enabled, or if the AIoT device’s RF transmission capabilities are temporarily disabled, then the AIoT device re-enables its RF transmission capabilities. Furthermore, the AIoT device replaces its stored code with the value of Code2 for future instructions.

If Code1 does not match the code stroed in the AIoT device, the Command Operation Request is discarded.

6, 7. AIoT device responds to the Command Operation Request to the Reader. The AIoT device may include its received/updated code (i.e., Code 2) in the response. The Reader in turn responds to the Command Operation Trigger to the AIoT Function.

8. If the Command Operation Trigger comes from the AF, the response is forwarded to the AF.

### 6.5.3 Evaluation

Solution fully addresses security requirement in KI#1. The solution does not address the security of the Command Operation procedure as it is part of communication security between AIoT device and the network.

The solution requires the AIoT device to be provisioned with a code to disable or re-enable RF transmission capabilities. The codes from the network (either CN or AF) are sent as part of secured Command Operation procedure and are secure from attackers. Replacing the code in the AIoT device after each temporary disabling/enabling RF transmission capabilities using the Command Operation procedure also prevents the code being replayed.

## 6.6 Solution #6: AIoT device authentication

### 6.6.1 Introduction

This solution addresses KI#5.

This solution propose an AIoT device Security Management and an AIoT controller. The AIoT device Security Management is a core network function, has the following security capabilities:  
 1.Stores the initial Kaiot together with the AIoT device. This key is used for MAC calculation.

1. Sends the Kaiot to AIoT controller to calculate the network MAC.

AIoT controller has capability about calculating the network MAC and verifying the device MAC.

AIoT device has capability about calculating the device MAC and verifying the network MAC.

The counter is maintained by the AIoT controller and the AIoT device.

### 6.6.2 Solution details

The following figure shows the call flow for AIoT device authentication.



Figure 6.6.2-1: AIoT device authentication

0. AIoT device and AIoT device Security management both store the Kaiot. AIoT device stores Device ID and initial counter. The AIoT device Security management stores the mapping relationship between the Device ID and reader ID.

1. AF sends Inventory(or command) request to AIoT controller, which includes Reader ID and Device ID. The Reader ID indicates the reader requested for Inventory(or command). Multiple Reader IDs can be included. The Device ID indicates the AIoT device for Inventory(or command). Multiple Device IDs can be included.

2. AIoT controller sends Device information request to AIoT device Security management, which includes Device ID received in step 1. If multiple Device IDs are received in step 1, AIoT controller can repeat this step per Device ID for multiple times, or AIoT controller can send the Device ID list to AIoT device Security management in the same message.

3. AIoT device Security management sends Device information response to AIoT controller, which includes the reader ID, and Kaiot. If Multiple Device IDs are received in step 2, this message may include all device information corresponding to the Device IDs.

4. The AIoT controller calculates the network MACn based on the Kaiot and counter, for example, MACn= HASH (device ID, reader ID, counter, Kaiot).

5. The AIoT controller sends the Inventory(or command) Request to the reader(s) base on reader ID in step 3, which includes Device ID and MACn. If the counter is initial value, which includes start indication.

6. The Reader(s) send Inventory(or command) Request to AIoT device, which includes Device ID and MACn, and which may includes start indication.

7. The AIoT device verifies the MACn, that is, the AIoT device calculates the local MACn', for example, MACn’=HMAC(device ID, reade ID, counter), the input key is Kaiot. The counter is the counter stored in the AIoT device. If start indication is valid, the the counter is initial counter value. If MACn and MACn' are the same, the verification succeeds. The AIoT device increases the counter by one, and store the new counter. The AIoT device can optionally calculate the MACu, for example, MACu= HMAC (device ID, reade ID, new counter), the input key is Kaiot, where Kaiot is the Kaiot stored on the AIoT device.

8. The AIoT device sends Inventory(or command) Response to the Reader with its Device ID, which may includes MACu.

9. The Reader sends Inventory(or command) Response to the AIoT controller with the Device ID and Reader ID, which may includes MACu.

10. The AIoT controller calculates the MACu' like the AIoT device, and verifies the MACu'. If the verification is successful, the Inventory(or command) is successful. The AIoT controller increases the counter by one. The AIoT controller stores the new counter.

11. AIoT controller stores the Device ID and the Reader ID, if multiple Inventory(or command) Responses are received from different Readers, the AIoT controller stores the Device ID and multiple Reader IDs. AIoT controller sends the Inventory(or command) Response to AF with the Device ID and Reader ID(s).

Editor’s Note: How the AIoT device get reader ID in step 7 is FFS.

Editor’s Note:if maintaining device ID and reader ID mapping is required or not, and if required how it is done is FFS.

Editor’s Note: It is FFS how synchronization of counter between devices and Controller is ensured and how to recover from the even of de-synchronization.

Editor’s Note: The impact of sharing key Kaiot with AIoT controller is FFS.

Editor’s Note: How the AIoT controller verifes the MAC if there is multiple response in step 10.

Editor’s Note: It is FFS how to protect privacy of device identifier in the step 6 is FFS.

Editor’s Note: It is FFS if replay attacks are possible against the device or the network.

Editor’s Note:It is FFS if K needs confidentialkity/integrity protection against physical attack in the device.

### 6.6.3 Evaluation

TBD.

## 6.7 Solution #7: Lightweight AIoT Authentication solution

### 6.7.1 Introduction

The assumption of this solution is AIoT device and network sharing a long-term root key K[x].

MAC/XMAC is used for device authenticating network. MAC is calculated by network side with the K, XMAC is calculated by AIoT device side with the K.

Optionally, RES/XRES is used for network authenticating device. XRES is calculated by network side with the K, RES is calculated by device side with the K.

To provide the freshness of the authentication vectors, RAND is also used to calculate MAC/XMAC and optionally RES/XRES.

### 6.7.2 Details

MAC/XMAC is used for device authenticating network. MAC is calculated by network side with the K, XMAC is calculated by AIoT device side with the K.

Optionally, RES/XRES is used for network authenticating device. XRES is calculated by network side with the K, RES is calculated by device side with the K.

To provide the freshness of the authentication vectors, RAND is also used to calculate MAC/XMAC and optionally RES/XRES.

Similar with 5G-AKA, and based on RAN2 agreement for “inventory and command” case[6], AIoT authentication solution has the following steps:



Figure 6.7.2-1 AIoT Authentication based on 5G-AKA

1. AIoT NF sends authentication vector request to UDM/ARPF for AIoT device.
2. UDM/ARPF calculates MAC with K (e.g. the root key of AIoT device) and RAND.
3. Optionally, UDM/ARPF calculates XRES with K and RAND, if network wants to authenticate AIoT device.
4. UDM/ARPF sends RAND, MAC, device ID and optionally XRES to AIoT Authentication Function.
5. AIoT Authentication Function sends Authentication Request including RAND, MAC, device ID to AIoT NF.
6. AIoT NF sends Authentication Request including RAND, MAC, device ID to Reader.
7. Reader sends Authentication Request including RAND and MAC to AIoT device. Authentication Request is carried by Step C in the “inventory and command” case.
8. AIoT device calculates XMAC with RAND and K.
9. Optionally, AIoT device calculates RES with K and RAND.
10. AIoT device verifies XMAC=MAC, then the network authentication is successful. If there is command, only after successful verification, the command can be proceeded.
11. AIoT device sends Authentication Reponses to Reader, this message optionally including RES if network wants to authenticate AIoT device. Authentication Reponses is carried by Step D in the “inventory and command” case.
12. Reader sends Authentication Reponses to AIoT NF.
13. Optionally, AIoT NF sends Uplink Authentication Request including RES to AIoT Authentication Function.
14. Optionally, AIoT Authentication Function verifies XRES=RES, then the AIoT device Authentication is successful.

As an example, the generation of MAC/XMAC and RES/XRES is described as below.



Figure 6.7.2-2 The generation of MAC and XRES at network side.



Figure 6.7.2-3 The generation of MAC and XRES at AIoT device side.

Editor's Note: Further evaluation of the usage of the crypto primitives is FFS.

Editor's Note: The impact of removing the use of SEQNO (e.g., replay against device or network) from AKA is FFS

Editor's Note: Anonymity key (AK) is used to protect SEQNO in AKA. If SEQNO is not used, the necessity of using AK to compute MAC and XRES is FFS.

Editor's Note: It is FFS if the shared key needs confidentiality/integrity protection against physical attack in the device

Editor's Note: When the AIoT NF triggers the authentication is FFS.

Editor's Note: The security level of MAC/RES calculation compare to 5G-AKA is FFS.

Editor's Note: The intermediate authentication parameter storage in AIoT device is FFS.

### 6.7.3 Evaluation

TBA.

## 6.8 Solution #8: Mutual authentication for AIoT system

### 6.8.1 Introduction

This solution addresses key issue#5: “Authentication in Ambient IoT service.”

### 6.8.2 Details

Preassumption:

1. AIoT AUSF and AIoT UDM are independent network entities for AIoT system, they can also be collocated with legacy AUSF and UDM, depending on operators’ deployment.
2. The Applacation Function is out of operator domain, which can be the AIoT device manufacture.
3. It is assumed AIoT UDM owns the AIoT device credentials, while Application Function has no access to those credentials.
4. Step 1-3 are following RAN2 procedure using 2-step RACH.

A screenshot of a computer

Description automatically generated

Step 1. AF sends the authentication request to the AIoT UDM,

NOTE: The authentication request message can be carried in paging message, depending on RAN2/SA2 decision.

Step 2. AIoT UDM generates the RAND, sends the device ID and RAND to AIOT AUSF.

NOTE: AIoT UDM and AIoT AUSF can be collocated.

Step 3. AIoT AUSF sends the device ID and RAND to the AIoTF.

Step 4. AIoTF sends the device ID and RAND to the Reader, Reader includes the device IDs in the paging message.

NOTE: what IDs to be included in paging message will be decided by RAN2.

Step 5. AIoT device calculates the RES using K and device ID and RAND, using HASH function.

Step 6. AIoT device sends the device ID, RES, and random number Counter to Reader in uplink AS message.

NOTE: The message details to be dependent on SA2 decision.

Step 7-8. Reader sends the Authentication Request to AIoT AUSF for authentication.

Step 9. AIoT AUSF sends authentication request to AIoT UDM including the device ID and Counter.

Step 10, AIoT UDM calculates the XRES and network authentication Token. AIoT UDM then sends both values back to AIoT AUSF.

Step 11. AIoT AUSF compares the RES and XRES, if they are equal, AIoT AUSF sends network authentication Token back to Reader in the subsequent messages. Otherwise, the authentication fails.

Step 12-13. the AIoT AUSF passes the authentication result and network authentication Token back to Reader.

Step 14. If the authentication success, the Reader continue with step 15, otherwise, the Reader ceases the authentication procedure.

Step 15. Reader sends the network authentication Token to AIoT devices in DL command message.

Step16. AIoT device verifies if the network authentication Token is correct. If yes, then the authentication of the network is successful, and AIoT device continue with the subsequent UL message. Otherwise, the AIoT device will cease the procedure.

Depending on the device capability, network should decide how often authentication to be performed. If the device capability is sufficient and the authentication is not run every time before the inventory request, the network and the device shall maintain the authentication status of the devices. If the AIoT device capability is too limited, i.e. no storage for the authentication status, the network may need to run the authentication everytime when the inventory is triggered.

Editor’s Note: The synchronization issue is FFS.

Editor’s Note: The architecture of AIoT is FFS, based on SA2 progress.

### 6.8.3 Evaluation

Editor’s Note: The impact on device and network is FFS.

Editor’s Note: Further evaluation is FFS.

## 6.9 Solution #9: Device authentication and data communication security

### 6.9.1 Introduction

This solution tries to address the Key issue #3, Key issue #4 and Key issue #5.

This solution protects the privacy of AIoT Device ID by encrypting AIoT Device ID or calculating temporary ID using AIoT device authentication keys.

For AIoT Device authentication, only entities (AIoT Device or network) that possess the AIoT Device authentication key can successfully decrypt or verify the protected AIoT Device ID.

For communication security, only entities (AIoT Device or network) that possess the AIoT Device authentication key can generate session key. For each data transmission, both network nonce and AIoT Device nonce are used to protect the communication content.

### 6.9.2 Solution details

The security procedure for command operations is shown in the following figure.

 Figure 6.9. 2-1: Security procedure for command operations

0. During the AIoT Device initialization phase, the AIoT Device Authentication Key is pre-configured in the AIoT Device. The AIoT Device Authentication Key is used to authenticate the device and secure the communication with the AIoT Device. Security policies for AIoT Device ID protection and communication may also be pre-configured.

1. The AF sends a Command Operation Request to the Ambient IoT Function (AIoTF). The request includes the AIoT Device ID and Payload.

2. To authenticate an AIoT Device, the AIoTF performs the following operations:

Generate a Network Nonce;

Use the Device Authentication Key and Network Nonce to derivate keys for confidentiality and/or integrity protection;

Protect the AIoT Device ID with the new derived keys to obtain the Protected AIoT Device ID; The Protected AIoT Device ID can also be a temporary ID generated using the authentication key and security parameters;

Send a Paging message to the AIoT Devices through the AIoT Reader. The message includes the Protected AIoT Device ID and Network Nonce.

3. The AIoT Device performs the following operations:

Use the same method as the AIoTF to derive the keys, and then decrypts and/or verifies the Protected AIoT Device ID or calculate the temporary ID;

Check if the Device ID carried in the paging message matches the locally stored Device ID or locally calculated temporary ID;

Generate a Device Nonce;

Use the Device Authentication Key, Network Nonce and Device Nonce to derivate keys for Device ID confidentiality and/or integrity protection;

Protect the AIoT Device ID with the new derived keys to obtain the Protected AIoT Device ID\*; The Protected AIoT Device ID\* can also be a temporary ID generated using the authentication key and security parameters;

The AIoT Device returns the Protected Device ID\* and Device Nonce to the AIoTF through the AIoT Reader. It may also return its Device capability.

4. The AIoTF performs the following operations:

Use the same method as the AIoT Device to derive the keys, and then decrypts and/or verifies the Protected AIoT Device ID\* or calculate the temporary ID;

Check if the Device ID carried in the authentication request matches the locally stored Device ID or locally calculated temporary ID; If they match, the authentication is successful.

5. To protect a command message, the AIoTF performs the following operations:

Generate a Network Nonce;

Use the Device Authentication Key and Network Nonce to derivate keys for confidentiality and/or integrity protection;

Protect the AIoT Device ID with the new derived keys to obtain the Protected AIoT Device ID; The Protected AIoT Device ID can also be a temporary ID generated using the authentication key and security parameters;

Protect the Payload and/or the entire message using the new derived keys according to the Security Policy that specifies how the communication message is protected;

Send Command message to the AIoT Device through the AIoT Reader. The request includes Protected AIoT Device ID, Security Policy, Network Nonce, Secured Payload and MAC.

6. The AIoT Device performs the following operations:

Use the same method as the AIoTF to derive the keys;

Use the same method as step 3 to check the protected AIoT device ID;

Decrypts and/or verifies the Secured Payload and/or the entire message according to the Security Policy in the message; If the verification is successful, the AIoT Device continues to perform the following operations;

Generate a Device Nonce;

Use the Device Authentication Key, Network Nonce and Device Nonce to derivate keys for confidentiality and/or integrity protection;

Protect the Payload and/or the entire message using the new derived keys according to the Security Policy (either from the command request message or from the pre-configured security policy);

Send Command Response Message to the AIoTF through the AIoT Reader. The response includes Security policy, Device Nonce, Secured payload and MAC.

7. The AIoTF performs the following operations:

Use the same method as the AIoT Device to derive the keys, and then decrypts and/or verifies the secured Payload and/or the entire message; If the verification is successful, the AIoTF continue to perform the following operations;

Return the Payload to the AF.

Editor's Note: It is FFS if generating device Nonce twice makes the protocol computationally correct.

Editor's Note: It is FFS if the shared key needs confidentiality/integrity protection against physical attack in the device.

Editor's Note: It is FFS how to prevent power and resource exhaustion in the AIoT devices when all of devices in the paging area always have to decrypt the Device ID.

### 6.9.3 Evaluation

TBD

6.10 Solution #10: Authentication for AIoT device

### 6.10.1 Introduction

This solution addresses KI#5 to propose 3 authentication procedures for AIoT devices including UE reader and RAN reader cases.

### 6.10.2 Solution details

#### 6.10.2.1 UE reader case

##### 6.10.2.1.1 Alternative 1 – UE reader granularity

The following figure shows the call flow for AIoT device authentication via UE reader with UE reader granularity. This procedure has the following main points:

1. The operator is responsible for authorizing UE readers to provide AIoT services. This means that the operator does not manage individual AIoT subscriptions, but rather focuses on UE reader subscriptions. This allows the AIoT service to operate within licensed spectrum, with billing based on UE reader usage (e.g. per AIoT service) rather than the AIoT devices themselves.
2. As there is only SLA between the UE readers and operators, the operator does not require knowledge of AIoT authentication. Therefore, there is no specification of authentication between AAA and AIoT, and no need for SLA and trust establishment between AAA and 5GC. Instead, authentication is encapsulated in an Authentication Container.



Figure 6.102.1.1-1: Authentication for AIoT device via UE reader with UE reader granularity

0. The UE reader has established a PDU Session for authentication. The 5GC authorizes the PDU Session establishment procedure, e.g., based on subscription data.

1. The AIoT AF sends Inventory command to the AIoT NF/AMF via NEF.

2. The AIoT NF/AMF selects the UE reader for the inventory.

3. The AIoT NF/AMF interacts with UDM/UDR to authorize the UE reader for the inventory, e.g., whether the UE reader is allowed to serve the AF for the inventory for the AIoT devices. If authorization succeeds, the UDM/UDR returns authorization information (e.g. AAA address) to the UE reader.

4. In case the authorization succeeds, the AIoT NF/AMF responds with authorization success indication to the AIoT AF via NEF, the AIoT AF is ready for inventory notification according to the success indication.

5. In case the authorization succeeds, the AIoT NF/AMF sends AIoT paging command encapsulated in a DL NAS Transport message to the UE reader. The authorization information may be included in the DL NAS Transport message.

6. The UE reader may interact with AAA-S based on the authorization information over the PDU Session before paging AIoT devices, e.g., using mechanism described in RFC 4739 [7]. The AAA-S may return Auth1 (e.g., NONCENW) information to the UE reader.

An example of using the IKEv2 extension described in RFC 4739 [7] is that UE reader is performed as initiator, and the AAA-S is performed as responder, at the first authentication, the UE reader provides its own UE ID to be authenticated with AAA-S, while at the second authentication, the UE reader provides AIoT ID to the AIoTF to be authenticated with AAA-S, the UE reader will terminate EAP protocol, and translates it to Auth container to simplify AIoT’s authentication protocol.

7. The UE reader broadcasts AIoT paging message, if the authentication information is received, it may include an Auth Container I that constructs based on the aAuth Information1 received from the AAA-S.

NOTE 1: The AIoT paging message is not mandate to support including Auth Container, in case that Auth Container is not included in the AIoT paging, it will be included in step 8 and the following steps, and there is no impact on AIoT paging. In case that AIoT container is included in the AIoT paging, whether and how to include Auth Container in the AIoT paging message, and the size restriction of the Auth Container needs RAN coordination.

8. The AIoT device determines to responds to the AIoT paging message, it sends an AIoT message to the UE reader with Device ID and optional Auth Container 2, which contains information for authentication.

9. The UE reader acts as proxy of the AIoT device to interact with AAA-S over the PDU Session for authentication between the AIoT device and the AAA-S, e.g., using EAP framework or mechanism described in RFC 4739 [7]. The UE reader may further interact with the AIoT device for authentication if required by AAA-S.

10. In case the authentication succeeds, the UE reader will receive authentication success indication from AAA-S. The UE reader reports the inventory result to the AIoT AF.

\

The example of usage of authentication method and protocol can be referred in 6.10.2.3.

##### 6.10.2.1.2 Alternative 2 – AIoT device granularity

The following figure shows the call flow for AIoT device authentication via UE reader with AIoT device granularity. This procedure has the following main points:

1. The operator is responsible for managing AIoT subscriptions, but the AIoT credentials are stored in AAA. That means the operator needs to trust authentication result from AAA (i.e. there is SLA between operators and external AAA owner, or operator manage the AAA server). This allows operator to bill based on AIoT devices themselves.
2. It is also assumed that AIoT credentials are stored in AAA with specific authentication method, or AIoT device and AAA are within the same network domain or the same vendor. Therefore, there is no specification of authentication between AAA and AIoT. Instead, authentication is encapsulated in an Authentication Container.



Figure 6.10.2.1.2-1: Authentication for AIoT device via UE reader with AIoT device granularity

0. The UE reader has registered into 5G network. The UE reader may establish a PDU Session for AIoT service.

1-4. The same as described in steps 1-4 of clause 6.10.2.1.1.

5. In case the authorization succeeds, the AIoT NF/AMF sends instructs the UE reader to page AIoT device(s). In case of CP method, the instruction is encapsulated in a DL NAS Transport message to the UE reader. In case of UP method, the instruction is delivered via the PDU Session to the UE reader. The authorization information may be sent to the UE reader too.

6. The UE Reader may interact with AAA-S via the AIOT NF/AMF before paging AIoT devices, e.g., using mechanism described in RFC 4739[7]. The AAA-S may return Auth Information 1 to the UE Reader via the AIoT NF/AMF.

An example of using the IKEv2 extension described in RFC 4739 [7] is that the UE reader acts as Initiator and EAP Peer, the AIoT NF acts as EAP Authenticator, the AAA-S acts as responder and EAP Authentication Server. At the first authentication, the UE reader may provide its own ID to the AIoTF to be authenticated with AAA-S (external authentication), while at the successive authentication, the UE reader provides AIoT ID to the AIoT NF to be authenticated with AAA-S, the UE reader will terminate EAP protocol, and translates it into Auth container to simplify authentication protocol for AIoT device.

To be more specific, the UE reader initiates IKE\_SA\_INIT request and receives MULTIPLE\_AUTH\_SUPPORTED notification from the AAA-S via the AIoT NF. The UE reader then may perform step 7 or send its ID in the first IKE\_AUTH request to trigger authentication between the UE reader and the AAA-S using EAP over IKEv2 to establish trust between the AAA-S and the UE reader for proxying the authentication between the AIoT devices and the AAA-S. The AAA-S may include the Auth Information1 in the AUTH field of the first IKE\_AUTH response, which also contains an EAP request.

The successive example IKEv2 message exchange based on RFC 4739 [7] is described in step 9.

7. The UE reader broadcasts AIoT paging message, which may include the Auth Container 1.

NOTE 1: The AIoT paging message is not mandate to support including Auth Container, in case that Auth Container is not included in the AIoT paging, it will be included in step 8 and the following steps, and there is no impact on AIoT paging. In case that AIoT container is included in the AIoT paging, whether and how to include Auth Container in the AIoT paging message, and the size restriction of the Auth Container needs RAN coordination.

8. The AIoT device determines to responds to the AIoT paging message, it sends an AIoT message to the UE reader with Device ID and optional Auth Container 2, which contains information for authentication. If the Auth Container1 is included in the AIoT paging message, the Auth Container2 is generated based on the Auth Container1. The UE reader forwards the AIoT message encapsulated in a UL NAS Transport message or via a PDU session to the AIoT NF/AMF.

9. The AIoT NF/AMF acts as proxy of the AIoT device to interact with AAA-S for authentication between the AIoT device and the AAA-S, e.g., using EAP framework or mechanism described in RFC 4739 [7]. The AIoT NF/AMF may further interact with the AIoT device via the UE reader for authentication if required by AAA-S.

NOTE 2: Step 9a may consist multiple message exchanges between the UE reader and the AAA-S for sending the Device ID and the Auth Information2 constructed based on Auth Container2 separately to the AAA-S.

In case step 6 is performed, an example of using the IKEv2 extension described in RFC 4739 [7] in this step is that after first authentication is done, the UE reader initiates further IKE\_AUTH request with ANOTHER\_AUTH\_FOLLOWS notification to the AAA-S to request further authentication with AIoT ID. After receiving IKE\_AUTH response from the AAA-S, the UE reader sends another IKE\_AUTH request with AIoT ID for authentication between the AIoT device and the AAA-S using EAP over IKEv2.

10. In case the authentication succeeds, the AIoT NF/AMF will receive authentication success indication from AAA-S. The AIoT NF/AMF reports the inventory result to the AIoT AF.

The example of usage of authentication method and protocol can be referred in 6.10.2.3.

#### 6.10.2.2 RAN reader case

The following figure shows the call flow for AIoT device authentication via RAN reader. The main points are the same with alternative 2 of UE reader case as depicted in 6.10.2.1.2.



Figure 6.10.2.2-1: Authentication for AIoT device via RAN reader

1. The AIoT AF sends Inventory command to the AIoT NF/AMF via NEF.

2. The AIoT NF/AMF selects the RAN reader for the inventory.

3. The AIoT NF/AMF interacts with UDR to authorize the inventory, e.g., whether the AF is allowed to perform the inventory. An example of the authorization can be based on the association between the AF ID and the inventory targets (e.g., device owner information, allowed area information, etc.) that stored in the UDR.

4. In case the authorization succeeds, the AIoT NF/AMF may interact with AAA-S before paging AIoT devices, e.g., using mechanism described in RFC 4739 [7]. The AAA-S may return authentication information to the AIoT NF/AMF.

An example of using the IKEv2 extension described in RFC 4739 [7] is that the AIoT NF/AMF acts as Initiator and EAP Peer, the AAA-S acts as responder and EAP Authentication Server. At the first authentication, the AIoT NF/AMF may be authenticated with AAA-S using certificate (external authentication), while at the successive authentication, the AIoT NF/AMF provides AIoT ID to be authenticated with AAA-S, the AIoT NF will terminate EAP protocol, and translates it into Auth container to simplify authentication protocol for AIoT device.

5. In case the authorization succeeds, the AIoT NF/AMF responds with authorization success indication to the AIoT AF via NEF, the AIoT AF ready for inventory reports according to the success indication.

6. In case the authorization succeeds, the AIoT NF/AMF sends AIoT paging command to the RAN reader, which may include an Auth Container 1 that constructs based on Auth Information 1 received from the AAA-S.

7. The RAN reader broadcasts AIoT paging message, which may include the Auth Container.

NOTE 1: The AIoT paging message is not mandate to support including Auth Container, in case that Auth Container is not included in the AIoT paging, it will be included in step 8 and the following steps, and there is no impact on AIoT paging. In case that AIoT container is included in the AIoT paging, whether and how to include Auth Container in the AIoT paging message, and the size restriction of the Auth Container needs RAN coordination.

8. The AIoT device determines to responds to the AIoT paging message, it sends an AIoT message to the AIoT NF/AMF via the RAN reader with Device ID and optional another Auth Container, which contains information for authentication.

9. The AIoT NF/AMF acts as proxy of the AIoT device to interact with AAA-S for authentication between the AIoT device and the AAA-S, e.g., using EAP framework or mechanism described in RFC 4739[7]. The AIoT NF/AMF may further interact with the AIoT device for authentication if required by AAA-S.

NOTE 2: Step 9 may consist multiple message exchanges between the AIoT NF/AMF and the AAA-S for sending the Device ID and the Auth Information2 constructed based on Auth Container2 separately to the AAA-S.

10. In case the authentication succeeds, the AIoT NF/AMF will receive authentication success indication from AAA-S. The AIoT NF/AMF reports the inventory result to the AIoT AF.

The authentication method and protocol can be referred in 6.10.2.3.

#### 6.10.2.3 Example of usage of authentication method and protocol

The authentication method and protocol used between the UE reader and AAA-S determines the process of transporting Auth information. If the UE reader and AAA-S are from different vendors, standardised procedures such as example#1 or example2 can be used. However, if they are from the same vendor, procedures such as example#3 can be used. The EAP framework can be used between the UE reader and the AAA-S no matter which authentication method used.

Example#1 - CHAP: the Auth Information1 will include NONCENW, and the Auth Container2 is generated based on the NONCENW, e.g., including RES, no more steps on Auth Container3 and Auth Container4 is needed.

Example#2 - AKA based: there is no Auth Information1and Auth Container2. The Auth Information3 includes RAND and AUTN. The Auth Container4 will include RES\*.

Example#3 –lightweight authentication method: the Auth Information1 will include NONCENW, the Auth Container2 includes a NONCEDV and RES generated based on the NONCEDV and the NONCENW, no more steps on Auth Container3 and Auth Container4 is needed.

### 6.10.3 Evaluation

TBA

## 6.11 Solution #11: Authentication and ID Privacy of AIoT devices with USIM on AIoT AS Layer

### 6.11.1 Introduction

This solution is addressing the Key Issue #5: Authentication and Key issue #3: Privacy by protecting AIoT device identifiers and Key issue #4: Protection of information during AIoT service communication.

Some solutions in 3GPP TR 23.700-13 [4] suggests the following simplified protocol stack for Ambient IoT as shown in Figure 6.11.1-1:



Figure 6.11.1-1: Potential AIoT protocol stack

This new architecture provides a control plane delivery of commands and instructions towards the AIoT device from the corresponding AF in charge. Since NAS protocol is not supported and the AIoT devices have a low complexity, it is assumed that only EAP-AKA’ is supported and the non-3GPP access procedures can be reused, comparable to N5CW or AUN3 devices. With that concept it is possible to either to protect the messages between AIoT device and AIoT function, similar to untrusted access, or between AIoT device and AIoT Reader similar to trusted access.

It is assumed that the AIoT device can use SUCI based on the presence of the USIM.

The security relationship is established between AIoT Device and the AIoT Reader, it is assumed that the communication between AIoT Reader and the AIoT Function is protected with NDS/IP.

### 6.11.2 Solution details

In this solution for adopting the trusted access principles, the AIoT Reader is taking the role as the TNAP and the AIoT Function the role as TWIF/TNGF and AMF.



Figure 6.11.2.2-1: AIoT Device authentication and ID Privacy as trusted non-3GPP access

1. The AF is authenticated based on the mechanisms described in TS 33.501, e.g. TLS, a token based authorization mechanism or local configuration at the NEF. The AF subscribes for device information of authenticated AIoT devices.

2. The AIoT device connects to an AIoT Reader as an access network. The AIoT device may be triggered by the AIoT Reader to send this message, e.g. based on the presence of a specific broadcast message from AIoT Reader.

3. The AIoT reader sends an L2 message with an EAP-Identity Request to the AIoT Device. This step may be part of step 5.

4. The AIoT device provide its unique AIoT Identity, e.g. SUCI or 5G-GUTI in an EAP-Identity Response to the AIoT Reader. The message may contain other information e.g. Device EPC.

5. The AIoT Reader selects a AIoT Function, e.g. based on the received realm, and sends an AAA request to the selected AIoT Function. The AIot Reader may include the received information from the AIoT Device, e.g. Device EPC, and the device location in the request.

6. The AIoT Function shall select an AUSF and sends Nausf\_UEAuthentication\_Authenticate Request message to the AUSF. The Nausf\_UEAuthentication\_Authenticate Request message contains SUCI or SUPI (in case of a valid 5G-GUTI is received by the AIoT Function acting as an AMF). The request message contains also an indication that the request is from an AIoT device.

7. The AUSF shall send Nudm\_UEAuthentication\_Get Request to the UDM including SUCI or SUPI and the AIoT indication. Upon reception of the Nudm\_UEAuthentication\_Get Request, the UDM shall invoke SIDF if a SUCI is received. SIDF shall de-conceal SUCI to gain SUPI before UDM can process the request. The UDM may select an authentication method based on the "realm" part of the SUPI, the AIoT device indicator, a combination of the "realm" part and the AIoT device indicator, or the UDM local policy. The UDM/ARPF shall first generate an authentication vector. The UDM shall subsequently send this transformed authentication vector AV' (RAND, AUTN, XRES, CK', IK') to the AUSF. The UDM shall also send the MSK indicator to the AUSF to indicate that the AIoT device does not support the 5G key hierarchy.

8. The AUSF shall send the EAP-Request/AKA'-Challenge message to the AIoT Function in a Nausf\_UEAuthentication\_Authenticate Response message.

9. The AIoT Function shall transparently forward the EAP-Request/AKA'-Challenge message to the AIoT Reader in an AAA response message.

10. The AIoT function forwards the EAP-Request/AKA’-Challenge message to the AIoT Device in a L2 message.

11. The AIoT device computes the authentication response message.

12. The AIoT device shall send the EAP-Response/AKA'-Challenge message to the AIoT Reader in a Auth-Resp message on L2.

13. The AIoT Reader shall send the EAP-Response/AKA'-Challenge message to the AIoT Function in a AAA request message.

14. The AIoT Function shall transparently forward the EAP-Response/AKA'-Challenge message to the AUSF in Nausf\_UEAuthentication\_Authenticate Request message.

15. The AUSF shall verify the message by comparing the XRES and RES. If successful, based on the MSK indicator received in step 11, the AUSF shall generate the MSK, the AUSF shall not generate the KAUSF.

16. The AUSF shall send to the AIoT Function an Nausf\_UEAuthentication\_Authenticate Response message including the EAP-Success, the MSK, and the SUPI and if available the GPSI.

17. AIoT Function sends an EAP-Success to the AIoT Reader upon reception of the MSK, and the SUPI and if available the GPSI.

18. The AIoT Reader stores the MSK and forwards the EAP-Success to the AIoT Device in a L2 message.

Editor’s Note: it is FFS whether the nr of interactions with the device are feasible for AIoT

### 6.11.3 Evaluation

Editor’s Note: The solution needs to be aligned with the final SA2 conclusions on the architecture.

## 6.12 Solution #12: Authentication and ID Privacy of AIoT devices with USIM on AIoT Layer

### 6.12.1 Introduction

This solution is addressing the Key Issue #5: Authentication and Key issue #3: Privacy by protecting AIoT device identifiers and Key issue #4: Protection of information during AIoT service communication.

Some solutions in 3GPP TR 23.700-13 [4] suggests the following simplified protocol stack for Ambient IoT as shown in Figure 6.12.1-1:



Figure 6.12.1-1: Potential AIoT protocol stack

This new architecture provides a control plane delivery of commands and instructions towards the AIoT device from the corresponding AF in charge. Since NAS protocol is not supported and the AIoT devices have a low complexity, it is assumed that only EAP-AKA’ is supported and the non-3GPP access procedures can be reused, comparable to N5CW or AUN3 devices. With that concept it is possible to either to protect the messages between AIoT device and AIoT function, similar to untrusted access, or between AIoT device and AIoT Reader similar to trusted access.

It is assumed that the AIoT device can use SUCI based on the presence of the USIM.

The security relationship is established between AIoT Device and the AIoT Function and adopts the untrusted access concepts.

### 6.12.2 Solution details

In this solution for adopting untrusted access principles, the AIoT Reader is taking the role as the Access Point and the AIoT Function the role as N3IWF and AMF.



Figure 6.12.2.1-1: AIoT Device authentication and ID Privacy as untrusted non-3GPP access

1. The AF is authenticated based on the mechanisms described in TS 33.501, e.g. TLS, a token based authorization mechanism or local configuration at the NEF. The AF subscribes for device information of authenticated AIoT devices.

2. The AIoT device connects to an AIoT Reader as an access network. The AIoT device may be triggered by the AIoT Reader to send this message, e.g. based on the presence of a specific broadcast message from AIoT Reader. When the AIoT device decides to attach to the AIoT Reader, the AIoT device selects an AIoT Function in a 5G PLMN. The AIoT Device may retrieve the address of the AIoT Function in a broadcast message of the AIoT Reader to which the AIoT Reader is connected to, or, the AIoT Function address may be preconfigured in the AIoT Device. The AioT Device retrieves limited IP connectivity from the AIoT Reader to perform the procedure.

3. The AIoT device proceeds with the establishment of an IPsec Security Association (SA) with the selected AIoT Function by initiating an IKE initial exchange.

4. The AIoT device shall initiate an IKE\_AUTH exchange by sending an IKE\_AUTH request message. The AUTH payload is not included in the IKE\_AUTH request message, which indicates that the IKE\_AUTH exchange shall use EAP signalling (in this case EAP-5G signalling).

5. The AIoT Function responds with an IKE\_AUTH response message which includes the AIoT Function identity, the AUTH payload to protect the previous message it sent to the AIoT device (in the IKE\_SA\_INIT exchange) and an EAP-Request/5G-Start packet.

6. The AIoT device shall validate the AIoT Function certificate and shall confirm that the N3IWF identity matches the AIoT Function selected by the AIoT device. The AIoT device shall send an IKE\_AUTH request which includes an EAP-Response/5G-NAS packet that contains a unique AIoT identifier, e.g. such as SUCI or 5G-GUTI and may contain the Electronic Product Code (EPC) of the AIoT device.

7. The AIoT Function shall select an AUSF and sends Nausf\_UEAuthentication\_Authenticate Request message to the AUSF. The Nausf\_UEAuthentication\_Authenticate Request message contains SUCI or SUPI (in case of a valid 5G-GUTI is received by the AIoT Function acting as an AMF). The request message contains also an indication that the request is from an AIoT device.

8. The AUSF shall send Nudm\_UEAuthentication\_Get Request to the UDM including SUCI or SUPI and the AIoT indication. Upon reception of the Nudm\_UEAuthentication\_Get Request, the UDM shall invoke SIDF if a SUCI is received. SIDF shall de-conceal SUCI to gain SUPI before UDM can process the request. The UDM may select an authentication method based on the "realm" part of the SUPI, the AIoT device indicator, a combination of the "realm" part and the AIoT device indicator, or the UDM local policy. The UDM/ARPF shall first generate an authentication vector. The UDM shall subsequently send this transformed authentication vector AV' (RAND, AUTN, XRES, CK', IK') to the AUSF. The UDM shall also send the MSK indicator to the AUSF to indicate that the AIoT device does not support the 5G key hierarchy.

9. The AUSF shall send the EAP-Request/AKA'-Challenge message to the AIoT Function in a Nausf\_UEAuthentication\_Authenticate Response message.

10. The AIoT Function shall transparently forward the EAP-Request/AKA'-Challenge message to the AIoT device in an IKE\_AUTH response message.

11. The AIoT device computes the authentication response message.

12. The AIoT device shall send the EAP-Response/AKA'-Challenge message to the AIoT Function in a Auth-Resp message.

13. The AIoT Function shall transparently forward the EAP-Response/AKA'-Challenge message to the AUSF in Nausf\_UEAuthentication\_Authenticate Request message.

14. The AUSF shall verify the message by comparing the XRES and RES. If successful, based on the MSK indicator received in step 11, the AUSF shall generates the MSK, the AUSF shall not generate the KAUSF.

15. The AUSF shall send to the AIoT Function an Nausf\_UEAuthentication\_Authenticate Response message including the EAP-Success, the MSK, and the SUPI and if available the GPSI.

16. AIoT Function sends an EAP-Success/EAP-5G to the AIoT device upon reception of the MSK, and the SUPI and if available the GPSI.

Editor’s Note: it is FFS whether the number of device interactions are feasible for AIoT

Editor’s Note: how USIM is supported is FFS

### 6.12.3 Evaluation

Editor’s Note: The solution needs to be aligned with the final SA2 conclusions on the architecture.

## 6.13 Solution #13: Authentication and ID privacy of AIoT devices without USIM

### 6.13.1 Introduction

This solution is addressing the the Key Issue #5: Authentication and Key issue #3: Privacy by protecting AIoT device identifiers and Key issue #4: Protection of information during AIoT service communication.

Some solutions in 3GPP TR 23.700-13 [4] suggests the following simplified protocol stack for Ambient IoT as shown in Figure 6.13.1-1:



Figure 6.13.1-1: Potential AIoT protocol stack

This new architecture provides a control plane delivery of commands and instructions towards the AIoT device from the corresponding AF in charge. Since NAS protocol is not supported and the AIoT devices have a low complexity, it is assumed that the devices in addition do not have a USIM for authentication and the security procedures. It is further assumed that the devices do have a simple security configuration which is shared with the AF. For ID privacy, the devices use a default ID for the onboarding to the AIoT network, after that only a temporary ID is used, derived from the security configuration. Confidentiality keys are derived from the security configuration, integrity protection may not be required based on the level of importance of the downlink messages.

### 6.13.2.1 Solution details



Figure 6.13.2-1: AIoT Device authentication and ID Privacy

1. The AF has a preshared configuration of the AIoT devices, which includes a unique Default ID of the device and respective security parameters for deriving a security key and temporary IDs for ID privacy. The security parameters comprise a shared secret key, only known to the AIoT Device and the AF.

2. The AF sends an AIoT Request to the NEF with the Default Id and the security parameters of the AIoT device.

3. The Nef forwards the AIoT Request to the selected AioT Function.

4. The AIoT Function generates a Nonce and uses it to derive an Encryption Key and a Temporary ID from the received security context from the NEF. The AIoT Function uses the Encryption Key to calculate an Expected Result.

The AIoT Function sets the sequence number SQN to “1”.

The encryption key K and the Temporary ID are computed as follows:



Figure 6.13.2-2: Temporary ID and Encryption Key generation

The Temporary ID and the Encryption Key are in concatenated form the output of the HMAC function. The split of the Temporary ID and the Encryption Key may be equal, i.e. the key length and the Temporary ID length are the same. The Temporary ID may be formed from the most or least significant bits of the output hash, the encryption key bits are the remaining ones.

The expected result is a proof of the Default Identity, computed in the following way:



Figure 6.13.2-3: Proof/Expected Result generation

The AIoT Function computes a MAC over the full message, i.e. using the inputs Default ID, Nonce and SQN:



Figure 6.13.2-4: Paging MAC generation

5. The AIoT Function sends an AIoT Request to the AioT Reader, including the Default ID and the Nonce, the SQN and the MAC.

6. The AIoT Reader sends the AIoT Request to the AIoT Device, which is listening to requests with the Default ID for initial onboarding to the AIoT network.

7. The AIoT Device verifies the MAC and the expected SQN. The AIoT Device calculates the Encryption Key and the Temporary ID for the next usage in a similar way as the AIoT Function. The AIoT Device calculates the Result as a proof that it holds the security context as shown in Figure 6.13.2-3. The AIoT Device increases the sequence number SQN and protects the message with a MAC, with the proof as input instead of the Nonce to the HMAC function. The AIoT Device will listen now to the paging with the Temporary ID#1 and expects the payload encrypted with the encryption key K#1.

8. The AIoT Device sends a AIoT response to the AIoT Reader, including the computed Result.

9. The AIoT Reader forwards the AIoT Response to the AiOT Function.

10. The AIoT Function verifies the MAC and the expected SQN and compares the received result with the expected result and authenticates the AIoT Device if both are identical.

11. The AIoT Function sends a AIoT Response to the NEF, indicating the success of the authentication.

12. The Nef forwards the AioT Response to the AF.

13. The initial registration with the setup of the security association and authentication is completed.

All further requests from the AF are then encrypted by the AIoT Function and the AIoT Device, the AIoT device is addressed by the Temporary ID only. The AIoT Function may change the Encryption Key and the Temporary ID by providing a new Nonce in a protected downlink request.

14. The AF sends an AIoT Request to the NEF with the Default Id of the AIoT device and the Command request.

15. The NEF forwards the AIoT Request to the selected AIoT Function.

16. The AIoT Function increases the sequence number SQN before using it for the Temporary ID and Encrpytion Key generation. The AIoT Function selects the device context based on the Default ID and generates a new Nonce#2 and uses it to derive a new Encryption Key K#2 and a new Temporary ID#2, which are activated when the AIoT Function receives a reply to the request with the Temporary ID#1. The AIoT Function uses the previously generated Encryption Key K#1 to encrypt the new Nonce#2 and the Command message. The AIoT Function uses the previously generated Temporary ID#1 to address the device. The AIoT Function protects the message with a MAC as follows:



**Figure 6.13.2-5: MAC generation of Command Messages**

The message is then protected in the following way:



**Figure 6.13.2-6: AIoT Function Message protection**

17. The AIoT Function sends an AIoT Request to the AIoT Reader, including the Temporary ID#1, the Nonce, the SQN, the Command and the MAC.

18. The AIoT Reader sends the AIoT Request to the AIoT Device, which is listening to requests with the Temporary ID#1.

19. The AIoT Device verifies the MAC and the expected SQN. The AIoT Device calculates the new Encryption Key K#2 and the Temporary ID#2 for the next usage in a similar way as the AIoT Function. The AIoT Device executes the Command from the AF. The AIoT Device increases the sequence number SQN and uses the previously generated Encryption Key K#1 to encrypt the Command Result and protects the message with a MAC.

The AIoT Device will listen now to the paging with the Temporary ID#2 and expects the payload encrypted with the encryption key K#2.

20. The AIoT Device sends a AIoT response to the AIoT Reader, including the Command Result.

21. The AIoT Reader forwards the AIoT Response to the AIoT Function.

22. The AIoT Function verifies the MAC and the expected SQN. The AIoT function decrypts the Result with the Encryption Key K#1 before forwarding towards the AF. For the next message, the AIoT Function activates the encryption key K#2 and Temporary ID#2 and generates a new Nonce, Temporary ID and Encryption Key.

23. The AIoT Function sends a AIoT Response to the NEF, including the result of the Command Request.

24. The Nef forwards the AIoT Response to the AF.

Editor’s Note: Whether Nonce in Step 6 can be sent to the device depends on RAN paging message

#### 6.13.2.2 Handling of Temporary ID mismatch

In case of Temporary ID mismatch, the paging to the AIoT device will fail. It is assumed that the AIoT Device listens also to the paging with the Default ID to reset the security context.



**Figure 6.13.2.2-1: Re-synchronization in case of Temporary ID mismatch**

1. The AIoT Device is registered according to Figure 6.13.2-1, steps 1 – 13.

2. The AF sends an AIoT Request to the NEF with the Default Id of the AIoT device and the Command request.

3. The NEF forwards the AIoT Request to the selected AIoT Function.

4. The AIoT Function selects the device context based on the Default ID and generates a new Nonce#2 and uses it to derive a new Encryption Key K#2 and a new Temporary ID#2, which are activated when the AIoT Function receives a reply to the request with the Temporary ID#1. The AIoT Function increases the sequence number SQN. The AIoT Function uses the previously generated Encryption Key K#1 to encrypt the new Nonce#2 and the Command message. The AIoT Function uses the previously generated Temporary ID#1 to address the device. The AIoT Function protects the message with a MAC.

5. The AIoT Function sends an AIoT Request to the AIoT Reader, including the Temporary ID#1, the Nonce, the SQN, the Command and the MAC.

6. The AIoT Reader tries to deliver the AIoT Request to the AIoT Device, which is not responding due to Temporary ID mismatch.

7. AIoT Reader sends a response message to the AIoT Function that the AIoT device is not reachable.

8. The AIoT Function resets the profile of the device and generates a new Nonce#3 to derive an Encryption Key K#3 and a Temporary ID#3. The AIoT Function uses the Encryption Key to calculate an Expected Result as a proof of the Default ID. The AIoT Function sets the sequence number SQN to “1”.

The AIoT Function computes a MAC over the full message, i.e. using the inputs Default ID, Nonce and SQN:

9. The AIoT Function sends an AIoT Request to the AIoT Reader, including the Default ID, the Nonce#3, the SQN and the MAC.

10. The AIoT Reader sends the AIoT Request to the AIoT Device, which is listening to requests with the Default ID.

11. The AIoT Device detects that the paging occurs with the Default ID and not with the expected Temporary ID and that the AIoT Function in that way resets the secrutiy association. The AIoT Device verifies the MAC and detects that the SQN is reset to “1”. The AIoT Device calculates the Encryption Key#3 and the Temporary ID#3 for the next usage in a similar way as the AIoT Function. The AIoT Device calculates the Result as a proof that it holds the security context. The AIoT Device protects the message with a MAC.

The AIoT Device will listen now to the paging with the Temporary ID#3 and expects the payload encrypted with the encryption key K#3.

12. The AIoT Device sends a AIoT response to the AIoT Reader, including the computed Result.

13. The AIoT Reader forwards the AIoT Response to the AIoT Function.

14. The AIoT Function verifies the MAC and the expected SQN and compares the received result with the expected result and authenticates the AIoT Device if both are identical. The AIoT Function increases the sequence number SQN before using it for the Temporary ID and Encrpytion Key generation. The AIoT Function selects the device context based on the Default ID and generates a new Nonce#4 and uses it to derive a new Encryption Key K#4 and a new Temporary ID#4. The AIoT Function uses the previously generated Encryption Key K#3 to encrypt the new Nonce#4 and the Command message. The AIoT Function uses the previously generated Temporary ID#3 to address the device. The AIoT Function protects the message with a MAC.

15. The AIoT Function sends an AIoT Request to the AIoT Reader, including the Temporary ID#3, the Nonce, the SQN, the Command and the MAC.

16. The AIoT Reader sends the AIoT Request to the AIoT Device, which is listening to requests with the Temporary ID#3.

17. The AIoT Device verifies the MAC and the expected SQN. The AIoT Device calculates the new Encryption Key K#4 and the Temporary ID#4 for the next usage in a similar way as the AIoT Function. The AIoT Device executes the Command from the AF. The AIoT Device increases the sequence number SQN and uses the previously generated Encryption Key K#4 to encrypt the Command Result and protects the message with a MAC.

The AIoT Device will listen now to the paging with the Temporary ID#4 and expects the payload encrypted with the encryption key K#4.

18. The AIoT Device sends a AIoT response to the AIoT Reader, including the Command Result.

19. The AIoT Reader forwards the AIoT Response to the AIoT Function.

20. The AIoT Function generates a new Nonce, Temporary ID and Encryption Key and sends a AIoT Response to the NEF, including the de-crypted result of the Command Request.

21. The NEF forwards the AIoT Response to the AF.

#### 6.13.2.3 Handling of group of devices

The paging of a group of devices assumes a preconfigured group ID and a shared group security key in the AIoT devices and the AF. The paging of a group of devices is performed in the same way as a single device, just with the difference that it is assumed that a Temporary Group ID and Encryption Key is not necessary for this operation.



**Figure 6.13.2.3-1: Paging of group of devices**

1. The AF has a preshared configuration of the AIoT devices, which includes a unique Group ID of the devices and respective shared security group key for integrity protection.

2. The AF sends an AIoT Request to the NEF with the Default Id and the security parameters of the AIoT device.

3. The NEF forwards the AIoT Request to the selected AIoT Function.

4. The AIoT Function generates a Nonce and protects the whole message with a MAC in the following way:



**Figure 6.13.2.3-2: Paging MAC computation**

The payload may be the indication that the paging if for inventory etc.

5. The AIoT Function sends an AIoT Request to the AIoT Reader, including the Group ID, Nonce and the MAC.

6. The AIoT Reader sends the AIoT Request to the AIoT Device, which is listening to requests with the Group ID.

7. The AIoT Device verifies the MAC and generates a response message accordingly.

8. The AIoT Device sends a AIoT response to the AIoT Reader, including the response Result.

9. The AIoT Reader forwards the AIoT Response to the AIoT Function.

10. The AIoT Function verifies the MAC and collects the responses from other group devices.

11. The AIoT Function sends a AIoT Response to the NEF.

12. The NEF forwards the AIoT Response to the AF.

#### 6.13.2.4 Handling of encryption key mismatch

In case the AIoT device detects a key mismatch, it resets the security context and triggers a reset of the security context with the AIoT function.



**Figure 6.13.2.4-1: Re-synchronization in case of key or sequence number mismatch**

1. The AIoT Device is registered according to Figure 6.13.2-1, steps 1 – 13.

2. The AF sends an AIoT Request to the NEF with the Default Id of the AIoT device and the Command request.

3. The Nef forwards the AIoT Request to the selected AIoT Function.

4. The AIoT Function increases the sequence number SQN before using it for the Temporary ID and Encryption Key generation. The AIoT Function selects the device context based on the Default ID and generates a new Nonce#2 and uses it to derive a new Encryption Key K#2 and a new Temporary ID#2, which are activated when the AIoT Function receives a reply to the request with the Temporary ID#1. The AIoT Function increases the sequence number SQN. The AIoT Function uses the previously generated Encryption Key K#1 to encrypt the new Nonce#2 and the Command message. The AIoT Function uses the previously generated Temporary ID#1 to address the device. The AIoT Function protects the message with a MAC.

5. The AIoT Function sends an AIoT Request to the AIoT Reader, including the Temporary ID#1, the Nonce, the SQN, the Command and the MAC.

6. The AIoT Reader sends the AIoT Request to the AIoT Device, which is listening to requests with the Temporary ID#1.

7. The AIoT Device verifies the MAC and the expected SQN and detects a SQN and/or a key mismatch, i.e. the AIoT Device cannot decrypt the Nonce#2 and the Command Request. The AIoT device generates a new Nonce and computes a proof of the Default ID and a Temporary Re-Synchronization ID and Ecnryption Key and protects the message with a MAC.

The Temporary Re-Synchronization ID is computed similar to the Temporary ID, using the current Device SQN as input. The Encryption Key K#RS is used to protect the Nonce generated by the AIoT Function in the next message from the AIoT Function. The AIoT Device includes a Result with Key or SQN mismatch indication in the message to the AIoT Function.

The AIoT Device will listen now to the paging with the Temporary ID#RS and expects the payload encrypted with the encryption key K#RS.

8. The AIoT Device sends a AIoT response to the AIoT Reader, including the Device SQN, Nonce, Result, Proof of Default ID.

9. The AIoT Reader forwards the AIoT Response to the AIoT Function.

10. The AIoT Function detects the Result with the Key or SQN mismatch. The AIoT function verifies the proof of the Default ID and re-synchronizes the SQN to the value of the Device SQN. The AIoT Function computes the Temporary Re-Synchronization ID#RS and then increases the SQN and generates a new Nonce#3 to derive a new Encryption Key K#3 and a new Temporary ID#3. The previously generated Encryption Key K#2 and a new Temporary ID#2 from step 4 are deleted. The AIoT Function uses the previously generated Encryption Key K#RS to encrypt the new Nonce#3 and the Command message. The AIoT Function uses the previously generated Temporary ID#RS to address the device. The AIoT Function protects the message with a MAC.

11. The AIoT Function sends an AIoT Request to the AIoT Reader, including the Temporary ID#RS, the Nonce#3, the SQN, the Command and the MAC.

12. The AIoT Reader sends the AIoT Request to the AIoT Device, which is listening to requests with the Temporary ID#RS.

13 The AIoT Device verifies the MAC and the expected SQN. The AIoT Device calculates the new Encryption Key K#3 and the Temporary ID#3 for the next usage in a similar way as the AIoT Function. The AIoT Device executes the Command from the AF. The AIoT Device increases the sequence number SQN and uses the previously generated Encryption Key K#RS to encrypt the Command Result and protects the message with a MAC.

The AIoT Device will listen now to the paging with the Temporary ID#3 and expects the payload encrypted with the encryption key K#3.

14. The AIoT Device sends a AIoT response to the AIoT Reader, including the Command Result.

15. The AIoT Reader forwards the AIoT Response to the AIoT Function.

16. The AIoT Function verifies the MAC and the expected SQN. The AIoT function decrypts the Result with the Encryption Key K#RS before forwarding towards the AF. For the next message, the AIoT Function activates the encryption key K#3 and Temporary ID#3 and increases the sequence number SQN and generates a new Nonce, Temporary ID and Encryption Key.

17. The AIoT Function sends a AIoT Response to the NEF, including the result of the Command Request.

18. The NEF forwards the AIoT Response to the AF.

### 6.13.3 Evaluation

Editor’s Note: The solution needs to be aligned with the final SA2 conclusions on the architecture.

The solution requires a preconfiguration of a security key and/or group security key in the AIoT Device and in the AF.

For addressing, the solution requires a preconfiguration of a Default ID and/or Group ID in the AIoT Device and in the AF.

Authentication is performed with HMAC of the Default ID as a proof of the security key.

Temporary ID and Encryption Key generation is performed in one operation with an HMAC of SQN and Nonce.

Editor’s Note: further evaluation is FFS

## 6.14 Solution #14: Information protection during AIoT service communication

### 6.14.1 Introduction

The solution addresses the security requirement of KI#4: Protection of information during AIoT service communication.

The basic principle of this solution is to reuse principles of Integrity protection in 5GS [5] but simplify the tasks in a AIoT device. Furthermore, in this solution description also reuses the concept of locally generated Temporary ID (TempID) for privacy protection. However, this solution does not strictly depend on that solution, other means of handling temporary IDs are possible.

The solution assumes the following AIoT device capabilities:

- The AIoT device has higher complexity than a RFID tag that only reflects the same preconfigured device ID when excited by RF power, but significantly lower complexity than a 3GPP CIoT device.

- The AIoT device has a factory-encoded key and device ID, and the network also has these stored or can retrieve them.

- The AIoT device has a non-volatile storage capability.

As the available power in an AIoT device is very limited, the message exchange between the device and the network must be minimized and the computational requirement should be minimized. The solution is based on the following principle:

- The AIoT device can locally derive new temporary identifiers (TempID) to be used in future communication, a list of unused TempIDs. The solution in clause 6.26 for temporary ID derivation can be used. Solution 26 describes how to re-synchronize the TempID algorithm in case of TempID get out-of-synch between the CN NF and AIoT device, which can happen when temporary IDs are derived locally the in both the CN NF and AIoT device.

NOTE: The solution requires the AIoT device to have a few TempIDs available that has not been used i.e., never sent in clear text over the radio interface.

- The AIoT device uses these identifiers together with the factory-encoded key to protect the information transmitted.

- The derivation of an MAC is used. The intention is to make the derivation with low complexity.

- The Inventory service and Command service would require different type of protection.

### 6.14.2 Solution details

#### 6.14.2.1 Inventory Service information protection

The information transmitted from the AIoT device is an ID triggered by an Inventory request targeting e.g. all AIoT devices, group, or type of AIoT devices. Inventory is not typically targeted for one individual device, but that case should still be considered.

Eavesdropping: To protect against eavesdropping the AIoT device shall transmit an ID that cannot be linked to specific AIoT device.

Manipulation/unauthorized transmission: To Protect against this, the system shall never send the same ID over the radio interface and have a method to check that the ID is authentic if needed.

The solution to protection against the listed attacks is the same. The UE shall always send an ID that has never been sent over the radio interface. For the listed cases the following procedure principle applies:

**Group/all device Inventory request:**

* The Reader transmit an Inventory request targeting a group of AIoT devices or all AIoT device.
* The AIoT device responds to the request using a TempID that has never been used before (first in the list)
* In case the Reader UE wants to verify that the ID received is authentic the Reader can perform an Individual device Inventory Request.

**Individual AIoT device Inventory request:**

* The Reader transmit an Inventory request targeting a specific of AIoT device by including the device TempID that has never been used before (first in the list).
* The AIoT device responds to the request using a TempID that has never been used before (second in the list, as the first TempID was used in the Inventory request).
* In case the Reader UE wants to verify that the ID received is authentic the Reader repeats this request.

#### 6.14.2.2 Command Service information protection

The information transmitted from the Reader is an ID and command. The information transmitted by the UE is an ID and a response. The Command is triggered by a Reader and targeting e.g. a group of AIoT devices or an individual AIoT device.

Eavesdropping: To protect against eavesdropping an ID that cannot be linked to specific device and optionally encrypt the command.

Manipulation: To protect against this, a MAC can be derived/verified using at least the following input parameters the command/response and a shared secrete.

unauthorized transmission: To protect against this, the system shall never send the same ID over the radio interface and have a method to check that the ID is authentic if needed. Additionally, deriving a MAC as discussed above can be added.

**MAC derivation**

The solution proposes to use available parameters and use these in a simplified "Integrity Algorithm" e.g. scramble the input with the key.



Figure 6.14.2.2-1: Derivation of MAC or XMAC.

Figure 6.14.2.2-2: Alternative derivation of MAC or XMAC.

**Downlink (Reader -> AIoT device)**

The DL command message is sent to the AIoT device using the TempID (first in the list)

In addition to the Command, the Reader includes also a MAC derived as follows:

The KEY input is the factory-encoded key in the device.

The TempID input is the next TempID in the list (second ID).

The DIRECTION bit shall be set to 1 for downlink.

MESSAGE input is the Command.

NOTE: In case DL Group-Command message is supported in Release 19, the following is proposed. Devices are provisioned with a Group ID and a group KEY. Both the Group Key and Group ID are used as inputs to derive the MAC and XMAC.

**Uplink (AIoT device -> Reader)**

The DL command message was sent to the AIoT device using a TempID (first in the list)

The AIoT device responds to the Reader by using the next TempID (second in the list) and includes the response (e.g., an ACK or Data) plus a MAC derived as follows:

The KEY input is the factory-encoded key in the device.

The TempID input is the next TempID in the list (third TempID).

The DIRECTION bit shall be set to 0 for uplink.

MESSAGE input is the UL response (e.g., ACK or data).

### 6.14.3 Evaluation

This solution fulfils KI#4 requirements with the following properties.

* For Inventory service, as proposed in this solution the use of TempID that is send only once over the radio interface provides information protection. If needed, further protection can be achieved by the network performing an individual inventory request.
* For the command service, as proposed in this solution the Temp IDs serves dual purpose i.e., addressing the device and deriving MAC (used as freshness parameter). That is beneficial as AIoT devices are power constraint.

Editor’s Note: Further evaluation is FFS.

## 6.15 Solution #15: End-to-end security protection of command procedure

### 6.15.1 Introduction

This solution addresses key issues #1 and #4. This solution provides a security mechanism for protecting the commands transmitted between an AIoT device and an Application Function (AF) for AIoT services. This solution assumes the AF manages the AIoT device identifier and the corresponding security protection profile. The security protection profile includes a device credential and an algorithm to use to protect the command.

NOTE 1: Device credential types are determined based on each AIoT service and device capability.

The proposed mechanism is also applied to protect the messages exchanged for enable/disable device operation. The security mechanism for protecting the commands is applied when the AIoT device and AF are provisioned with the security protection profile.

### 6.15.2 Solution details



Figure 6.15.2.1-1: End-to-end protection of messages during Command procedure

NOTE 2: The reference architecture and Command procedure described in clause 6.3 of TR 23.700-13 (i.e., solution #3) [4] are used to describe the end-to-end protection mechanism in this solution. The proposed mechanism can be applied to any reference architectures and procedures for AIoT services that require transmissions of messages between AIoT devices and an Application Function.

0. Each AIoT device is provisioned with its AIoT device identifier and security protection profile such as a device credential and an algorithm to use to protect the command. An Application Function (AF) manages the AIoT device identifier and the associated security protection profile.

1. When the AF triggers a Command procedure towards individual AIoT device(s), the AF protects a Command based on the Command Protection Key (CPK) and a freshness parameter. The AF, then, sends a message containing the protected Command to the AIoT Controller.

When the AF triggers a Command procedure towards a group of AIoT devices, it protects a Command based on the Group Command Protection Key (GCPK) and a freshness parameter. Then, the AF sends the protected Command to the AIoT Controller.

NOTE 3: CPK is either derived from the device credential or provisioned on the AIoT device by the AF.

NOTE 4: GCPK, if used, is provisioned on the AIoT devices by the AF.

NOTE 5: Freshness parameter can be a counter, time-based counter or a random number depending on the AIoT service and device capability.

2. The AIoT Controller provides the Command to the selected Reader(s).

3. The Reader sends the Command to the AIoT device(s).

NOTE 6: Whether device ID is included in the Command depends on the AS procedure specified in TR 38.769 [x]. If the Inventory (i.e., A-IoT Paging, A-IoT random access procedure, and D2R data transmission) is followed by the Command, the device ID is not included in the Command in step 3 as Reader already has the association with the target AIoT device after Inventory.

4. Upon receiving the Command, the AIoT device decrypts/verifies the received Command. If the verification is successful, the AIoT device processes the Command.

5. If the AIoT device needs to send a response, it generates a Command Response and protects it based on the CPK and a freshness parameter. The AIoT device, then, sends the protected Command Response.

In case of Group command, the AIoT device protects the Command response based on the GCPK and a freshness parameter.

6. Upon receiving the Command Response, the Reader sends it to the AIoT Controller with optional Enrichment data such as the location of the Reader if configured by AIoT Controller.

7. The AIoT Controller provides the protected Command Response to the AF.

8. The AF decrypts/verifies the received Command Response based on the CPK and the freshness parameter.

Editor’s Note: The procedure needs to align with SA2.

### 6.15.3 Evaluation

This solution addresses key issues #1 and #4.

In this solution, the Command procedure is protected using the symmetric key between AF and AIoT device.

Editor’s Note: Further evaluation is FFS.

## 6.16 Solution #16: Disabling operation procedure for Ambient IoT services

### 6.16.1 Introduction

This solution is proposed to address Key Issue #1, which provides a method for permanently disabling the Ambient IoT device(s). This solution applies to Topology 1 and Topology 2.

Considering different roles are involved in the Ambient IoT services, it is assumed that only the owner of Ambient IoT device can authenticate the Ambient IoT device and initiate the disabling operation procedure to disable the Ambient IoT device’s capability to transmit RF signals.

In this solution, it is assumed that the Ambient IoT device is stateful. After the authentication is performed successfully, the Ambient IoT device enters the next state (e.g. secure state), in which it can execute the disabling operation.

For disabling the Ambient IoT device, the owner sends the Disabling request to the Ambient IoT device via the gNB/Intermediate node. If the owner is the application provider, the Disabling request is E2E protected between the AF and Ambient IoT device. If the owner is the operator, the Disabling request is protected by the AIoTF/AMF.

Once receiving the Disabling command, the Ambient IoT device determines whether to execute this operation based on authentication status and verification result. By maintaining the authentication status, the Ambient IoT device can ensure that the Disabling command is sent from the legitimate requester instead of the attacker. By maintaining the security material, the Ambient IoT device can ensure that the Disabling command is not tampered by the attacker.

NOTE 1: The states of Ambient IoT device are defined in described in solutions (e.g. #15, #16, #33, etc.) of TR 23.700-13 [4], which are maintained in the NVM of Ambient IoT device.

### 6.16.2 Solution details

This solution assumes that the authentication between the Ambient IoT device and its owner has been performed before disabling operation and the authentication status is stored in the Ambient IoT device. It is also assumed that the Ambient IoT device already share the security materials for communication protection with the AF or AIoTF/AMF before disabling operation, which can be dedicated for disabling operation, or be generalized for all command procedure (i.e. read, write, disable).

NOTE 2: The authentication and security context establishment between the Ambient IoT device and its owner is out of scope of this solution.



Figure 6.16.2-1: Disabling operation procedure for Ambient IoT services

1. The device owner sends the Disabling request to AIoTF/AMF. The Disabling request may include Ambient IoT device ID(s), disabling command, etc. If the owner of Ambient IoT device is application provider, the disabling command is E2E protected between the AF and Ambient IoT device.

2. The AIoTF/AMF determines the gNB/Intermediate node to transmit this Disabling request.

3. If the owner of Ambient IoT device is operator, the AIoTF/AMF protects the disabling command using the shared security materials. The AIoTF/AMF sends the Disabling request to the selected gNB/Intermediate node.

4. Upon reception of the request message, the selected gNB/Intermediate node executes inventory procedure with the Ambient IoT device.

5. The selected gNB/Intermediate node sends the protected Disabling Command message to the Ambient IoT device.

6. Once receiving the Disabling command, the Ambient IoT device firstly determines whether the authentication has been performed based on the authentication status. If the requester has been authenticated, the Ambient IoT device further verifies the received Disabling request by using the shared security materials. If the verification is successful, the Ambient IoT device can execute the disabling operation.

Note: Once the disabling operation is finished, the owner removes the security context of Ambient IoT device.

Editor’s Note: Whether the Disabling acknowledgement message is needed is FFS.

Editor’s Note: Alignment with conclusion from TR 23.700-13 [4] is FFS.

### 6.16.3 Evaluation

This solution addresses the requirement of Key Issue #1 and applies to both Topology 1 and Topology 2.

This solution ensures that the disabling operation can only be performed after checking the authentication status and verification result. If the Ambient IoT device has not been authenticated by the network or the verification of disabling request is failed, the Ambient IoT device discards the disabling request.

In this solution, the states and security materials shared with AF or AMF/AIoTF are required to be maintained by the Ambient IoT device.

This solution assumes that the Ambient IoT device is stateful. The states are maintained in NVM of Ambient IoT device.

Editor’s Note: Further evaluation is FFS.

## 6.17 Solution #17: Disabling operation procedure for AIoT services

### 6.17.1 Introduction

The solution addresses the security requirement of KI#1: Protection for disabling device operation. According to TS 22.369 [2], the network operator shall provide a suitable mechanism to temporarily/permanently disable the capability of an Ambient IoT device or a group of Ambient IoT devices to transmit RF signals. The solution assumes the network operator is responsible for managing the AIoT device using the temporarily/permanently disable operation.

Specifically, the AIoT NF sends the AIoT Disable Request to the AIoT device. The message includes the disable security parameter, which is calculated by the shared key between AIoT devices and AIoT NF. The AIoT device calculates and verifies the disable security parameter using the same shared key to decide whether to perform the disable operation.

### 6.17.2 Solution details

Depicted in Figure 6.17.2-1 is the disable operation procedure for AIoT Services.

**

**Figure 6.17.2-1: Disable operation procedure for AIoT service**

1. The AF sends the AIoT Disable Request to the AIoT NF via the NEF, including the AIoT device filter information, and disable type (i.e., temporarily/permanently disable).
2. The AIoT NF sends the AIoT Disable Trigger to the RAN/UE Reader, including the AIoT device filter information.

The AIoT device filter information is used to page an AIoT device or a group of AIoT device, which could be the AIoT device ID, the group ID, or the partial ID of AIoT device.

1. The RAN/UE Reader sends the AIoT Paging Request to the AIoT device, including the AIoT device filter information.
2. The AIoT device checks whether the ID matches according to the AIoT device filter information.
3. The AIoT device sends the AIoT Paging Response to the RAN/UE Reader, including the AIoT device ID.
4. The RAN/UE Reader sends the AIoT Disable Trigger Response to the AIoT NF, including the AIoT device ID.
5. The AIoT NF uses f function to calculate the disable security parameter with the input parameters key K, AIoT ID, disable type and counter, i.e., f (K, AIoT ID, disable type, counter). The initial value of the counter is set to zero and AIoT NF increases the counter by one for each subsequent disable operation. When the counter overflows, the AIoT NF resets the counter to zero.

NOTE: The f functions can be HMACfunctions, f1-f5 functions, AES functions, etc.

NOTE: The key K can be the root key or its derived key of the AIoT device.

1. The AIoT NF sends the AIoT Disable Request to the RAN/UE Reader. The message contains the AIoT ID, disable security parameter, disable type and counter.
2. The RAN/UE Reader forwards the AIoT Disable Request to the AIoT device.
3. The AIoT device is required to store the counter. When the received counter in the AIoT Disable Request is less than the local counter, the device refuses the Request, otherwise the AIoT device uses the same shared key to verify the disable security parameters.
4. (a) The AIoT device performs the disable operation according to the disable type, if the calculated disable security parameter matches the received disable security parameter.

(b) The AIoT device sends the AIoT Disable Response to the RAN/UE Reader before disabling the RF capability, if the AF requires a feedback.

1. The RAN/UE Reader forwards the AIoT Disable Response to the AIoT NF.
2. The AIoT NF sends the AIoT Disable Response to the AF via the NEF.

### 6.17.3 Evaluation

This solution fulfils the security requirements in key issue #1.

This solution addresses the security of the Command Operation (i.e., manipulation) procedure by requiring the AIoT device to maintain the counter.

This solution requires the AIoT device and AIoT NF to be provisioned with a shared key to calculate the disable security parameter.

Editor’s Note: Further evaluation is FFS.

## 6.18 Solution #18: Authorization procedure for AF-based intermediate node selection

### 6.18.1 Introduction

This solution is proposed to address Key Issue #2, supporting the authorization for AF-based intermediate node selection. This solution applies to Topology 2.

The Inventory/Command procedure described in solutions (e.g. #11, #33, etc.) of TR 23.700-13 [2] is used to describe the UE authorization mechanism.

The authorization of intermediate node is based on the UE subscription data stored in the UDM. Once receiving the potential intermediate node information provided by the AF, the AIoTF interacts with its UDM to obtain the UE subscription data. The AIoTF determines whether the UE is authorized to provide the requested Ambient IoT service by using the Ambient IoT service information, Ambient IoT service area information, and/or Ambient IoT device information included in the UE subscription data.

### 6.18.2 Solution details



Figure 6.18.2-1: Authorization procedure for AF-based intermediate node selection

1. The AF sends the Inventory/Command request to AIoTF via NEF. The Inventory/Command request may include UE ID(s), Ambient IoT device ID, area information, command, etc.

2. For each UE as the potential intermediate node provided by the AF, the AIoTF interacts with its UDM to obtain the UE subscription data for UE authorization.

3. For each UE, the AIoTF checks the UE authorization as follows:

- If the Ambient IoT service information is included in UE subscription data, the AIoTF determines whether the UE is authorized to provide the requested Ambient IoT service.

- If the Ambient IoT service area information is included in UE subscription data, the AIoTF determines whether the UE is authorized to provide the requested Ambient IoT service.

- If the Ambient IoT device information is included in UE subscription data, the AIoTF determines whether the UE is authorized to provide the requested Ambient IoT service.

If multiple UEs are authorized, the AIoTF further decides to select one or more UE(s) as intermediate node(s).

4. For the selected UE, the AIoTF sends the inventory/command request to the UE.

Editor’s Note: The details of subscription information is to be aligned with SA2.

### 6.18.3 Evaluation

Editor’s Note: Further evaluation is FFS.

This solution addresses the requirement of Key issue #2.

This solution is applicable for both Inventory and Command procedures.

This solution allows the AIoTF to check the UE authorization by using the UE subscription data. The AIoTF obtains the identifier of candidate UE Reader from AF and interacts with the UDM to obtain its subscription data.

In this solution, the granularity of UE authorization can be Ambient IoT service level, Ambient IoT device level, or area level.

## 6.19 Solution #19: Authorization of AIoT capable UE in topology 2

### 6.19.1 Introduction

This solution addresses key issue #2: Authorization for 5G Ambient IoT services.

In the Topology 2, as defined in TR 38.848, the AIoT capable UE acting as the intermediate node is responsible for transferring the Ambient IoT data and/or signalling between AIoT devices and 5GS and under network control.

A black background with a black square

Description automatically generated with medium confidence

Figure 6.19.1-1: Topology 2

In this solution, it is proposed that the AIoT device and the AIoT capable UE acting as the intermediate UE are authorized by the network and provisioned by the network with a common security material to protect the interface between the AIoT devices and the AIoT capable UE acting as the intermediate UE.

In this solution, the interface between the AIoT capable UE acting as the intermediate node and AIoT devices is called Ntopo2. Security material to protect Ntopo2 procedure (e.g. Inventory, Command) between the AIoT devices and the AIoT capable UE acting as the intermediate UE is called Ntopo2 security material, e.g. for integrity protection, confidentiality protection.

When the protected messages received from the peer side over Ntopo2 is successfully verified by the receiving side (the AIoT devices or the intermediate UE), it indicates that the sending side (the intermediate UE or the AIoT devices) has been authorized by the network.

### 6.19.2 Solution details

The procedure for authorization for AIoT capable UE acting as the intermediate node for 5G Ambient IoT is described as follows.



Figure 6.19.2-1: Authorization of Intermediate UE for topology 2

0. The AIoT device is provisioned with the Ntopo2 security material and its associated information from the network. During the provisioning procedure, the network checks whether the AIoT device is authorized to use Ntopo2 for Ambient IoT service. The Ntopo2 security material and its associated information is described in step 3.

Editor’s Note: It’s FFS how the Ntopo2 security material is provisioned to the AIoT device, and by which 5GC NF.

NOTE1: One or more AIoT devices (e.g. AIoT devices in the same group) can be associated with a common Ntopo2 security material.

1. The AIoT capable UE performs Registration procedure as defined in TS 23.502. The UE includes the AIoT Intermediate node capability indication in Registration Request message. The AMF determines whether the UE is authorized to work as Intermediate UE for AIoT based on the UE’s Subscription data. If the UE is authorised to work as Intermediate UE, then the AMF also includes the authorization information in NGAP message sent to NG-RAN.

2. AF sends AIoT service request (e.g. Inventory Request, Command Request) to the AIoT capable UE. The AIoT service request may be sent to the AIoT capable UE via 5GC or from AF directly over user plane with input parameters containing the area information, device information, optional inventory strategy information, and optional report aggregation info etc.

NOTE2: The detail of AIoT service request procedure is to be defined by SA2.

3. The AIoT capable UE sends a Key Request message to the network to get the Ntopo2 security material if the Ntopo2 security material is not available locally. In addition, the AIoT capable UE includes its Ntopo2 security capability to be used over the interface with AIoT devices, e.g. supported ciphering algorithms. The AIoT capable UE may also include the information that is associated with Ntopo2 security material e.g. the spatial information (e.g. geographical area), the temporal information (e.g. time period, expiry), and/or the AIoT device information (e.g. device ID, device group ID, device type, device mask info etc).

4. The 5G NF checks whether the AIoT capable UE is authorized to work as Intermediate UE for Ambient IoT service and provisions Ntopo2 security material. The 5G NF may check the Ambient IoT service authorization information for the AIoT capable UE with the AF.

Editor’s Note: It’s FFS which 5G NF performs the authorization and provisions the security material.

5. The 5G NF returns the Ntopo2 security material in Key Response Message. In addition, the 5G NF may include the chosen Ntopo2 ciphering algorithm, the Ntopo2 security policies, and the information that is associated with Ntopo2 security material as described in step 3.

6. The AIoT capable UE interacts with NG-RAN for radio resource allocation if the UE is authorized as in step 1.

NOTE3: The detail for radio resource allocation is assumed to be defined by RAN.

7. If the AIoT service request is Inventory Request, the AIoT capable UE initiates Inventory request over Ntopo2, selects Ntopo2 security material as received in step 5 and based on the associated security material information and input parameters in AIoT service request, and protects the Inventory request message with Ntopo2 security material.

8. If verification of the protected Inventory request message is successful based on the Ntopo2 security material received in step 0, the AIoT device reports the device ID and/or other info to the AIoT capable UE in the response message. The response message is protected by Ntopo2 security material. Successful verification of the protected Inventory request message assures the AIoT device that the AIoT capable UE is authorized to provide the AIoT service.

9. If the AIoT service request is Command Request, the AIoT capable UE initiates Command request over Ntopo2 and protects the Command request message with Ntopo2 security material simliar as step 7.

10. If verification of the protected Command request message is successful based on the Ntopo2 security material received in step 0, the AIoT device performs the Command and sends the response message. The response message is protected by Ntopo2 security material. Successful verification of the protected Command request message assures the AIoT device that the AIoT capable UE is authorized to provide the AIoT service.

11. The AIoT capable UE verifies the AIoT service response messange (Inventory response or Command response). Successful verification of the AIoT service response message assures the AIoT capable UE that the AIoT device is authorized for the AIoT service. The AIoT capable UE sends the AIoT service response to the AF. The AIoT service response may be sent to the AF via 5GC or directly over user plane.

Editor’s Note: What is the security material and the protocol betwen AIoT device and reader is FFS.

Editor’s Note: Clarify if ntopo2 key is long-term key or not. If long-term, the security impact is FFS.

### 6.19.3 Evaluation

Editor’s Note: Each solution should motivate how the potential security requirements of the key issues being addressed are fulfilled.

## 6.20 Solution #20: Lightweight AIOT ID privacy based on hashes

### 6.20.1 Introduction

The assumption of this solution is AIoT device can not support 5G-AKA due to power or computational resource limitation.

The existing Key issue #3, Privacy by protecting AIoT device identifiers, specifies the following requirement:

*Mechanisms for mitigating privacy threats (described above) by identifying, linking, and tracking the identifiers of AIoT Device(s) shall be supported*.

In addition, RAN2#125bis chairman notes [4] detail the following baseline procedure:

*…*

*3. RAN2 will support two use cases, “inventory” and “command”. The definition, detailed wording is FFS*

*4. Baseline:*

*Step A: Based on the service request, the reader sends the Initial Trigger Message indicating device(s) that need to respond; Details FFS*

*Step B: Triggered device(s) performs the random access-like procedure, if needed; Details FFS*

*Step C: The device may perform the data communication with the reader as needed,: Details FFS*

*…*

This solution aims to address both, the KI#3 of the present document and the baseline procedure described in the RAN2#125bis agreement.

Moreover, the proposed procedure for obfuscating the AIoT Device AIOT\_ID describes the “inventory” use case but is equally applicable to the “command” use case.

### 6.20.2 Details

The simplified call flow associated with the proposed solution is presented below.



Figure 6.20.2-1: Call flow for Lightweight AIOT ID privacy based on hashes

Steps associated with the call flow in Figure 6.20.2-1:

0. Configuration and provisioning step

0a. AIOT Device is configured/provisioned with AIOT\_ID

0b. RAN Reader/Intermediate Node is configured/provisioned with a list of AIOT\_IDs

1. AF sends an Inventory Operation Request with the following information: target area for the operation, client which requests the operation, and match information used to filter and discover the target AIoT devices for the operation.

2. The NEF authorizes the AF request. If the AF request is authorized, The NEF discovers the AIoT function using the information in the Inventory Operation Request, e.g. using the target area for the operation to discover the AIoT function from NRF. If the target area for the operation matches the AIoT service area of the AIoT Function, the NRF returns the information for the AIoT function to the NEF.

3. The NEF forwards the Inventory Operation information to each of the selected AIoT functions.

4. The AIoT function discovers and selects a reader or an Intermediate Node to perform Inventory Operation according to the Inventory Operation information. e.g. using the target area for the operation to discover the readers. If the target area for the operation matches the AIoT service area supported by the readers, those readers can be selected to execute the inventory operation.

5. The AIoT Function sends an Inventory Request with the Inventory Operation information for each selected reader.

6. The selected reader executes the inventory operation towards the target AIoT Devices.

Note that step 6 may be eavesdropped, manufactured, and replayed by an attacker. An appropriate authorisation of the Reader and freshness control to prevent message 6 replay will be needed to remedy such attacks.

7. The AIoT Device performs AIOT\_ID selection based on matching of hashed AIOT\_ID

a. The AIoT Device computes a hash using RAND\_READ as salt for each AIOT\_ID received in step 6 as HASH(RAND\_READ, AIoT\_ID)

b. The AIoT Device compares the hash values of its provisioned AIOT\_ID and AIOT\_ID hashes from the broadcast message to find at least one matching hashed AIoT\_ID, i.e., HASH(RAND\_READ, AIoT\_ID) = HASH(RAND\_READ, provisioned AIoT\_ID)

c. The AIoT Device selects an AIOT\_ID that has a matching hash.

Note that step 7 may create an overhead for the AIOT that may lead to a resources depletion attack on AIoT Device. This overhead is either lighter or comparable with other methods for confidentiality protection of the AIoT Device identity.

8. The AIoT Device sends a Registration Request containing hashed obfuscated AIOT-ID

9. The RAN Reader or Intermediate Node performs AIOT\_ID selection based on matching of hashed AIOT\_ID

a. compute a hash using RAND\_READ for each AIOT\_ID

b. check that the hashed AIOT\_ID received from the AIOT Device matches one of the stored AIOT-ID from step 0b

10. The RAN Reader or Intermediate Node sends the Inventory Report with AIOT\_ID to AMF/AIOT AF

11-12.. AIoT Function reports the operation result to the AF vie NEF/AF.

### 6.20.3 Evaluation

This solution addresses the requirement of Key issue #3.

This solution proposes a lightweight privacy protection method for the AIoT Device identity. The proposed method provides lightweight confidentiality protection of AIoT Device identifiers.

Impact on the AIoT device and RAN Reader:

The AIoT device is assumed to have sufficient power budget to compute multiple HASH(RAND\_READ, AIoT\_ID). This computation (i.e., step 7) is required for every AIoT device in range for every received paging message if protection of paging messages is required.

The RAN Reader needs to be configured with AIoT\_ID list and be able to compute HASH(RAND\_READ, AIoT\_ID). Editor’s Note: Further evaluation is FFS.

## 6.21 Solution #21: Ephemeral AIOT ID security context based on puzzles for privacy

### 6.21.1 Introduction

The assumption of this solution is AIoT device can not support 5G-AKA due to power or computation resource limitation.

The existing Key issue #3, Privacy by protecting AIoT device identifiers, specifies the following requirement:

*Mechanisms for mitigating privacy threats (described above) by identifying, linking, and tracking the identifiers of AIoT Device(s) shall be supported*.

This solution aims to address KI#3 of the present document.

The proposed procedure for obtaining the ephemeral security context is equally applicable to both, “inventory” and “command” use cases.

The agreed AIOT Random Access framework is presented below.



**Figure 6.21.1-1: AIOT Random Access framework**

The steps in Figue6.X.1-1 are described below.

1. The reader sends a paging message and a set of occasion synchronization messages which respectively provides the device IDs of the devices to respond and configures/delimits the random access occasions for transmissions by the AIOT devices

2. An AIOT device selects an occasion (using at least slotted ALOHA as the baseline), and transmits a random device ID in MSG1

3. The reader, upon successful reception of MSG1, transmits MSG2 by including the received random device ID in MSG2.

4. If the device receives the echoed random device ID in MSG2, it transmits MSG3 which contains upper layer data (e.g., an application layer device ID)

5. MSG4 may be transmitted by the reader (e.g., for subsequent command transmission), but the understanding is that contention is already resolved at MSG2 transmission.

### 6.21.2 Details

The simplified call flow based on the AIOT Random Access framework is presented below. This procedure is based on the security properties of cryptographic puzzles.



**Figure 6.21.2-1: Modified AIoT Random Access Procedure for establishing ephemeral security between AIOT Device and Reader**

The steps in Figure 6.21.2-1 are described below.

1. The Reader determines to prepare a set of cryptographic puzzles and the Reader will prepare a set of N tuples. Each tuple is comprised of a K-MACi (key) and corresponding K-MACi-IND (key index corresponding to key).

2. The Reader composes a set of N cryptographic puzzles each hiding a tuple comprising of the Ephemeral Key K-MACi, corresponding Ephemeral Key Index K-MACi-IND , and either partial key or partial hash function argument.

3. The reader transmits a paging message and a set of occasion synchronization messages. The combination of the paging message and synchronization message identifies which AIoT devices should respond to the paging message. In other words, the combination of the paging message and synchronization messages identifies which AIoT devices should perform a random access procedure. The paging message includes one or more of the puzzles that were constructed in step 2.

4. The AIoT Device uses the information in the paging message and synchronization messages to determine that the AIoT Device needs to respond to the paging message. In other words, the AIoT Device determines that it needs to perform a RACH procedure. If the AIoT Device determines that it needs to perform a RACH procedure, then, a the AIOT Device randomly selects one puzzle from the set of N puzzles that were received in the paging message. If the AIoT Device does not determine that it needs to perform a RACH procedure, or the AIoT device determines that the strength of the puzzle, as determined by the puzzle number, does not satisfy certain requirements, then the procedure will stop in this step and the AIoT will not perform a RACH procedure. Determining that the strength of the puzzle does not satisfy the AIoT security requirements for the application means that the selected puzzle strength may compromise information used by the application after the security context is established with a puzzle of a certain strength.

5. The AIoT Device solves the puzzle that was selected in step 4 and recovers the security parameters.

6. An AIOT device selects an occasion (using at least slotted ALOHA as the baseline), and transmits a random device ID in a message, MSG1. The message also includes the K-MACi-IND that was recovered in step 5.

7. The Reader performs a lookup for the K-MACi from the corresponding K-MACi-IND received in step 6.

8. The AIOT Device and the Reader enter a state where they have established an ephemeral security context using K-MACi. The subsequent messages of this procedure may now be confidentiality and integrity protected using the ephemeral security context based on K-MACi. In other words, the AIOT Device and the Reader have established an ephemeral security context. The ephemeral security context is based on K-MACi. After this step, the AIoT Device may use the ephemeral security context to encrypt data that it sends to the reader and the reader may use the ephemeral security context to encrypt data that it sends to the AIoT Device. Thus, information can be sent more securely between the AIoT Device and the Reader. In addition, the ephemeral security context obtained in this step can be used to bootstrap more persistent security associations between AIOT functional entities.

9. Upon successful reception of MSG1, the reader transmits MSG2 by including the received random device ID in MSG2. The Reader may use K-MACi to encrypt some or all of the information in MSG2.

10. The AIoT Device uses K-MACi to decrypt some or all of the information in MSG2. A random device ID is an example of information that is carried in MSG2. If the AIoT Device determines that the random device ID that it transmitted in step 6 is included in MSG2, then the AIoT Device transmits MSG3 which contains upper-layer data (e.g., an application layer device ID). The AIoT Device may use K-MACi to encrypt some or all of the information in MSG3.

11. The Reader uses K-MACi to decrypt some or all of the information in MSG3. The reader may then Transmit MSG4 (e.g., for subsequent command transmission), but the understanding is that contention is already resolved at MSG2 transmission. The reader may use K-MACi to encrypt some or all of the information in MSG4.

### 6.21.3 Evaluation

This solution addresses the requirement of Key issue #3.

This solution proposes a method for the establishment of the ephemeral security context that can be used for privacy protection for the AIoT Device identity.

A limited paging message link budget may preclude using multiple puzzles in paging messages.

The use of ephemeral security association before the AIOT command triggering may cause an additional resource consumption for all AIOT devises in range.

Power and processing-constrained AIOT devices might not be capable of using puzzle-based methods. In addition, puzzle-based methods rely on the comparative abilities of AIOT devices vs. adversaries that may be not power and processing-constrained. A security analysis is needed to determine the applicability of puzzle-based methods in AIOT security.

Editor’s Note: Further evaluation is FFS.

## 6.22 Solution #22: Solution for protecting AIoT ID by using temporary ID

### 6.22.1 Introduction

This solution addresses key issue #3: Privacy by protecting AIoT device identifiers.

It is assumed that an AIoT device has power or computational resource limitation.

To avoid fake ID reporting, AIoTF checks the authenticity of the message by verifying the MAC which is generated using KAIoT and RANDAIoTF.

It is assumed that AIoT device and AF are provisioned with AIoT device identifier and a key (K).

Editor’s Note: Alignment with conclusion from TR 23.700-13 [4] is FFS.

### 6.22.2 Solution details



Figure 6.22.2-1 AIoT ID protection call flow

0. AIoT ID and a key (K) are provisioned to the AIoT device and AF. AF generates KAIoT from K and RANDAF.

1. AF sends AIoT service request message to AIoTF (AIoT Function). This message may be sent via NEF. AIoT service request message includes Temp AIoT ID #1, RANDAF, and KAIoT.

NOTE: If this message is sent for the first time, the Temp AIoT ID #1 is the AIoT ID.

Editor’s Note: Sending the AIoT device ID in plaintext in the first message is FFS.

2. AIoTF transfers AIoT service request message. This message includes RANDAIoTF, RANDAF, and Temp AIoT ID #1.

3. AIoT device derives KAIoT from K and RANDAF. After that, AIoT device generates MAC using Temp AIoT ID #1, RANDAIoTF, and KAIoT.

4. AIoT device responds with AIoT service response. The message includes RANDAIoT, Temp AIoT ID #1, and MAC.

5. After AIoTF finds KAIoT from Temp AIoT ID #1 received in step 4, AIoTF checks the authenticity of the message by verifying the MAC.

6. If the verification in step 5 is successful, AIoTF sends AIoT service response to AF. RANDAIoT and Temp AIoT ID #1 are included in this message.

7. AIoT device and AF generate Temp AIoT ID #2 from KAIoT, RANDAIoT, and Temp AIoT ID #1. The Temp AIoT ID #2 is used next time the AF requests a service to AIoT.

Editor’s Note: How to resolove syncronization issue on the temporary ID between AIoT device and network is FFS.

NOTE: Whether the key recovery attack can be performed depends on the algorithm to derive the temporary ID.

### 6.22.3 Evaluation

This solution addresses the requirement of Key Issue #3 by using temporary AIoT ID.

This solution assumes that AIoT device and AF are provisioned with AIoT device ID and a key.

Editor’s Note: Further evaluation is FFS.

## 6.23 Solution #23: AIoT device ID privacy protection using anonymity key

### 6.23.1 Introduction

This solution addresses the security requirement of KI#3: Privacy by protecting AIoT device identifiers.

To prevent the AIoT deivce ID from being exposed in the air interface, the 5GC can trigger the reader(s) to pag an AIoT device or a group of AIoT Devices by broadcasting a partial AIoT Device ID. The AIoT device ID is consist of the common part (e.g., the Home Network identifier, or the 3rd party identifier) and the unique part (e.g., an identifier used to identify a specific Ambient IoT device). The partial AIoT Device ID is the common part of a group of AIoT device ID.

The AIoT deivce matches the partial AIoT deivce ID will perform random access responding to the pagging message and report its permanent ID to the network. The AIoT device encrypts its permanent ID with an anonymity key AK, which is a shared key between AIoT device and AIoTF. The AK and corresponding Key ID is provided to the Network by the AF. The AIoTF may receive the AK and Key ID from AF’s service request or retrieve such information from UDM (provided in advance by the AF).

### 6.23.2 Solution details

This solution describes the AIoT device ID protection as shown in the following figure 6.Y.2-1.



Figure 6.23.2-1: AIoT device ID privacy protection using anonymity key

1. The AF sends the AIoT Service Request to AIoTF, including the AIoT device ID, may include the anonymity key AK and the key ID. The anonymity key AK and the key ID can be provided by the AF to the Network, and pre-stored in UDM. The AK include two kinds of keys: (1) a specific key for each AIoT device; or (2)a group key shared by a group of AIoT devices.
2. The AIoTF requests the Reader to start Paging the AIoT device or a group of AIoT devices, using the partial AIoT Device ID.

The AIoT device ID is consist of the common part (e.g., the Home Network identifier, or the 3rd party identifier) and the unique part (e.g., an identifier used to identify a specific Ambient IoT device). The partial AIoT Device ID is the common part of a group of AIoT device ID.

1. The Reader pages the AIoT device or a group of AIoT devices using the partial AIoT Device ID.

For one or mautiple AIoT device(s):

1. The AIoT deivce matchs the partial AIoT deivce ID will perform random access responding to the message and report its permanent ID to the network. To prevent the AIoT device permanent ID being exposed, the AIoT device encrypts the AIoT device permanent ID with the anonymity key AK. The input parameters to the ciphering should include at least a fresh value, this fresh value changes to prevent the AIoT device from being linked or traced.
2. The AIoT device sends the Paging Response to the Reader, including the encrypted AIoT device permanent ID, the Key ID, and the fresh value.
3. The Reader sends the encrypted AIoT device permanent ID, the Key ID, and the fresh value to the AIoTF.
4. The AIoTF may retrieve the AK using the received Key ID from the UDM or the Credential Holder’s AAA server if needed. The AIoTF uses the anonymity key AK and the fresh value to decrypt the AIoT device permanent ID. Then the AIoTF matches the decrypted AIoT devices ID(s) with the received AIoT device ID in step 1.

The AK and Key ID for concealing AIoT device ID or GID, could be updated periodically to prevent linkability attacks and possible known ciphertext attack.

### 

### 6.23.3 Evaluation

This solution fully addresses security requirement in KI#3.

This solution requires the AIoT device to be provisioned with the anonymity key to encrypt the permanent device ID. And this solution is suitable for both Inventory-only case and Inventory and Command case.

## 6.24 Solution #24: temporary ID based AIoT device privacy protection

### 6.24.1 Introduction

This solution addresses the security requirement of KI#3: Privacy by protecting AIoT device identifiers. The purpose of the AIoT device temporary ID is to provide an unambiguous identification of the AIoT device that does not reveal the AIoT device permanent identity. The AIoT device temporary ID is allocated by the AIoTF after AIoT device initial registration.

### 6.24.2 Solution details

This solution describes the allocation and usage of the AIoT device temporary identifier, as shown in the following figure 6.24.2-1.



Figure 6.24.2-1: signalling flow of the AIoT device temporary ID allocation and usage

1. The AIoT device temporary ID is allocated by AIoTF after AIoT device initial registration. It is assumed that there are available security contexts between AIoT device and the Network after the AIoT device initial registration. Hence, the AIoT device temporary ID is protected by security contexts of the AIoT device, and transferred to the AIoT device.
2. The AF sends the AIoT Service Request to the AIoTF via the NEF, including the AIoT device ID. The NEF maps the external AIoT device ID to internal AIoT device ID.

Editor’s Note: whether there is AIoT device external ID is based on SA2’s decision, which is FFS.

1. The AIoTF maps the received AIoT device ID to the AIoT device temporary ID, and use the AIoT device temporary ID to indicate the specific AIoT device. The AIoTF sends the AIoT Service Request (Inventory or Command) via the reader.

After the AIoT device temporary ID allocation, the AIoT device can be paged by its temporary ID, or partial of the temporary ID to enable more efficient radio signalling procedures.

1. The AIoT device sends the AIoT Service Response to the AIoTF via the reader, including the AIoT device temporary ID.
2. The AIoTF maps the AIoT device temporary ID to AIoT device ID to response the AF via the NEF. The NEF maps the internal AIoT device ID to external AIoT device ID.

Editor's Note: How to synchronize and reallocate the temporary ID is FFS.

### 6.24.3 Evaluation

This solution fully addresses security requirement in KI#3.

This solution requires the AIoT device to store the temporary ID allocated by the Network

Editor's Note: More Evaluation is FFS.

## 6.25 Solution #25: Use temporary identifier to protect the privacy of AIoT device identifiers.

### 6.25.1 Introduction

This solution addresses key issue #3: Privacy by protecting AIoT device identifiers.

### 6.25.2 Solution details

****

**Figure 1. AIoT Temporary Identifier configuration procedure.**

AIoT is an Ambient IoT Device which has ultra-low complexity power, cost and resource-constrained. The reader refers to the base station or UE based on the connectivity topologies.

1. Initial temporary identifier (TempID) and crenditials such as TempID generation key are known by both the AIoTF and the AIoT device through onboarding or registration procedure of AIoT device or other NAS/AS procedure.

1. The network sends AIoT request such as inventory request or command request to AIoT device. The indication is optional for TempID synchronisation.

2. Once triggered by AIoT request, the AIoT device sends information including TempID to the reader. If the indication is included in the request, initial TempID is used by AIoT device in this step.

3. The reader transfer the AIoT information to the core network, i.e., the AIoTF.

4. The AIoTF uses tempID to identify devices and perform operations. For example, the core network verifies the validity of the AIoT device based on the TempID and other subscription data of the device.

If the received tempID does not match any one of the TempIDs stored in the network side, the AIOTF stops the subsequent procedure and re-sends the request in step 1 including an indication. By the indication, the AIoT device replies with initial TempID. Correspondingly, the intial TempID is also used in the network side.

5. The network returns an acknowledgement to the reader. And a freshness parameter is included in the response.

6. The reader transfers the acknowledgement to the AIoT device.

7. Both the AIoT device and the core network function generate a new TempID according to the TempID derivation function for later use. The freshness parameter will be used in both sides.

Editor’s Note: whether the acknowledgement message needs to be protected is FFS.

TempID derivation function

When deriving the TempID from TempID generation key, the following parameters shall be used to form the input S to the KDF:

- FC = 0xxx;

- P0 = "TempID";

- L0 = length of "TempID"; (i.e. 0x00 0x06)

- P1 = device ID;

- L1 = length of device ID.

- P2 = freshness parameter;

- L2 = length of freshness parameter.

The input key KEY (i.e. TempID generation key) is long term key pre-configured both in AIoT device and core network function.

Device ID is the fixed identifier of AIoT device.

### 6.25.3 Evaluation

In this solution, a temporary identifier (temp ID)is used to addresse key issue #3: Privacy by protecting AIoT device identifiers.

The initial temp ID is pre-confiured in both AIoT device and the network, and the temp ID is updated after each time it is used between the device and network.

The AIoT device needs to store the temp ID

Edtor’s Note: Further evaluation is FFS.

## 6.26 Solution #26: Local generated Temporary ID to provide device privacy

### 6.26.1 Introduction

The solution addresses the security requirement of KI#3: Privacy by protecting AIoT device identifiers. Specifically, this solution proposes a method for the Ambient IoT system to generate and use Temporary IDs.

The basic principle of this solution is that a Temporary ID (TempID) is locally generated both by the CN NF and the AIoT device after every time the TempID has been sent over the radio interface as a response to an Inventory request or a Command request.

The solution assumes the following AIoT device capabilities:

- The AIoT device has higher complexity than a RFID tag that only reflects the same preconfigured device ID when excited by RF power, but significantly lower complexity than a 3GPP CIoT device.

- The AIoT device has a factory-encoded key and device ID.

- The AIoT device has a non-volatile storage capability.

- The TempID generation algorithm is light weight and enough complex to avoid unauthorized AIoT device tracking.

As the available power in an AIoT device is very limited, the message exchange between the device and the network must be minimized. The solution is based on the following principle:

- The initial temporary identifier (TempID) is known by both the CN NF and the AIoT device. After the AIoT device has been onboarded to the network the CN NF provision the AIoT device with the initial TempID and/or parameters to derive the initial TempID i.e., parameters for the TempID generation algorithm.

NOTE: It is assumed that during the onboarding procedure the CN NF can retrieve information from another NF or Application Function (AF) to onboard the AIoT device. The initial message from the UE during onboarding could e.g. include the device owner ID, URL, FQDN or other info that enables the CN NF to establish an IP connection with the AF that holds additional onboarding information needed.

- Every time the TempID has been sent over the radio interface as a response to an Inventory request or a Command request, both CN NF and AIoT device locally generate a new TempID. The exact algorithm used can be decided during the normative phase, but the assumption is that it at least uses the factory-encoded key and a seed.

- If the CN NF detects that the TempID is out of sync, i.e., the CN NF expected a different TempID than received during an Inventory or Command. The CN NF has two options to re-synchronize 1) send a new seed to restart the algorithm or 2) find the received TempID in the sequence of TempIDs and continue from there.

- On the AIoT device side, the device may keep old TempID(s) and also check DL trigger against those in case the CN NF has not updated the TempID.

NOTE 2: In case the AIoT device excepts an old TempID and responds to that message, it opens up for roll back attacks.

- There is no guarantee that the TempID is unique. In case the CN NF receives a TempID and the CN NF has duplicates

### 6.26.2 Solution details

#### 6.26.2.1 Temporary ID generation.

In the procedure below CN NF is used as a generic name for the 5GC Network Function that supports the Ambient IoT functionality. It is expected that SA2 will define and specify CN NF as part of the 5GC architecture.



Figure 6.26.2-1 Local temporary ID control

0. It is assumed that the AIoT device is pre-provisioned by the device owner with information to be used when onboarding to a network. The information includes a device unique identifier (unique at owner level, not necessary globally unique), device owner ID and security Key.

It is assumed that the device owner and the MNO has a Service Level Agreement (SLA) and the network is provisioned with information needed to onboard an AIoT device(s) or a URL or FQDN to establishing IP connection to an AF that holds the device unique onboarding information.

1. The AIoT device is triggered to onboard and sends an onboarding request that includes (device owner ID, device ID). The network either holds the necessary information to continue the onboarding or connects to the device owners AF and retrieves the necessary information from the AF to continue the onboarding.

NOTE: How the AIoT device is triggered to request onboarding is up to SA2 WG to specify.

The CN NF sends a response message (Accept/Reject, selected TempID algorithm, seed) to the AIoT device.

Editor’s Note: It is FFS whether the device ID must be concealed.

1. The CN NF creates a AIoT device context and generates locally the TempID to be used when triggering the AIoT device next time e.g. when sending a Command to the AIoT device. To generate the TempID the CN NF uses the device security key received by the device owner AF, selected algorithm, and seed.
2. The AIoT device generates locally the TempID to be used next time. To generate the TempID the AIoT device uses its pre-provisioned security key and the selected algorithm and seed received in the response message.
3. The CN NF sends a command message or Inventory request, that may be triggered by 3rd-part AF. The Command message includes the AIoT device TempID, PDU.
4. The AIoT device only considers Command message that includes its expected TempID and check the command PDU. In case of Inventory request the AIoT device check whether the Inventory is for the device.
5. The AIoT device responds to the Command or Inventory request.
6. Both the AIoT device and CN NF generates the next temporary ID.

### 6.26.3 Evaluation

This solution fulfils KI#3 requirements with the following properties.

* The AIoT device can pre-calculate TempIDs e.g., before going back to sleep after an event. That allows the AIoT device to immediately correlate any DL trigger using the pre-calculated Temp ID.
* Locally generated TempIDs reduces power consumption as there is no need to send encrypted Temporary IDs over the radio interface.
* Locally derived Temp IDs in both in the CN NF and AIoT device have a risk to get out-of-synch with each other. To mitigate this the following is proposed:
* The CN NF can when detected out-of-synch send a new seed to the AIoT device to restart the algorithm. Restarting the algorithm has some overhead. To recover from the out-of-sync issue without restarting the algorithm, the CN NF and device can perform the following:
* On the CN NF side, the NF can check the received TempID against the sequence of TempIDs and continue from there.
* On the AIoT device side, the device may save old TempID and check received TempID in the DL trigger message against those.
* There is no guarantee that the TempID is unique. In case the CN NF receives a TempID and the CN NF has duplicates of this TempID, the CN NF can trigger the AIoT device with next TempID to determine the AIoT device.

Editor’s Note: Further evaluation is FFS.

## 6.27 Solution #27: Privacy protection of AIoT device identifier based on a temporary identifier

### 6.27.1 Introduction

This solution addresses key issue #3.

This solution provides a privacy protection mechanism by introducing a temporary identifier associated with an AIoT device. In the proposed mechanism, an AIoT device generates a temporary device identifier (T-ID) when it needs to transmit its device identifier towards the network (e.g., for the purpose of inventory management). This solution assumes that an Application Function (AF) manages the AIoT device identifier and the corresponding security protection profile. The security protection profile includes device credential and an algorithm to use to generate a T-ID.

NOTE 1: The privacy protection mechanism based on a temporary identifier is applied when the AIoT device and AF are provisioned with the security protection profile.

NOTE 2: Device credential types are determined based on each AIoT service and device capability.

### 6.27.2 Solution details

#### 6.27.2.1 Procedures



Figure 6.27.2.1-1: Privacy protection based on a temporary device identifier (T-ID) during Inventory procedure

NOTE 3: The reference architecture and Inventory procedure described in clause 6.3 of TR 23.700-13 (i.e., solution #3) [4] are used to describe the privacy protection mechanism in this solution. The proposed mechanism can be applied to any reference architectures and procedures for AIoT services that require transmission of AIoT device identifier by AIoT device.

0. Each AIoT device is provisioned with its AIoT device identifier and security protection profile such as a device credential and an algorithm to use to generate a T-ID. An Application Function (AF) manages the AIoT device identifier and the associated security protection profile.

1-3. The AF triggers an Inventory procedure towards AIoT devices.

4. Upon receiving the Inventory Request from the Reader, the AIoT device replies to the Reader with an Inventory Response containing a temporary device identifier (T-ID). The AIoT device generates a new T-ID as described in clause 6.27.2.2.

5. Upon receiving a T-ID, the Reader sends the received T-ID to the AIoT Controller. Additionally, the Reader sends Enrichment data (e.g., the Reader location) along with the T-ID if configured by the AIoT Controller.

6. The AIoT Controller stores the received T-ID and Enrichment data.

7. There are two methods to provide Inventory Response to AF as follows:

- Pull-based procedure: When the AIoT Controller receives a Data Request from the AF (step 7a in Figure 6.27.2.1-1), it sends an Inventory Response containing the stored T-IDs along with the Enrichment data if exist (step 7b in Figure 6.27.2.1-1). If a list of T-ID(s) was included in the Data Request, the AIoT Controller only contains the information associated with the requested T-IDs in the Inventory Response.

- Push-based procedure: the AIoT Controller sends an Inventory Response (step 7b in Figure 6.27.2.1-1) containing the information received from previous steps to the corresponding AF if T-ID includes the service identifier (e.g., AF Identity).NOTE 4: In case of pull-based procedure in step 7, a freshness parameter needs to be synchronized between the AIoT device and AF.

Once the AF receives a list of T-ID(s) from the AIoT Controller, it identifies the corresponding AIoT device(s) by checking if the received T-ID matches with the one of expected T-ID(s).

NOTE 4: It is assumed that AF has generated the list of expected T-ID(s) of AIoT device(s) of interest before the matching is performed.

NOTE 5: In case of pull-based procedure in step 7, a freshness parameter needs to be synchronized between the AIoT device and AF.

#### 6.27.2.2 Generation of a temporary identifier

A new temporary device identifier (T-ID) is generated as follows:

T-ID = F(K, freshness parameter, AIoT device identifier), where F is a service specific function that generates a temporary ID. The K is the key provisioned at the AIoT device and AF for temporary ID generation. The K is either a device credential or derived from the device credential. The freshness parameter is determined based on the device capability. For example, an index can be used as a freshness parameter along with a refresh timer. In this case, the AIoT device increments the index if a refresh timer has expired.

### 6.27.3 Evaluation

This solution addresses the security requirements in key issue #3 by introducing a temporary identity based on symmetric key provisioned at an AIoT device and AF.

## 6.28 Solution #28: Privacy protection on AIoT device IDs

### 6.28.1 Introduction

This solution addresses key issue#3: “Mechanisms for mitigating privacy threats (described above) by identifying, linking, and tracking the identifiers of AIoT Device(s) shall be supported.”

### 6.28.2 Details

Preassumption: the AIoT device has one ID preconfigured by the manufacture (Application Function). Each device is configured with a root key, named K. K is unique for every AIoT device.

The device shall also maintain an Index to indicate to the network that how to map the HASH value to the HASH table. For example, when the Index is n, it indicates this is the n-th calculation of the HASH value of the device ID. The Index shall be sent uplink together with the HASH (device ID||Index).

A screenshot of a computer screen

Description automatically generated

Step 0a. the device is configured with one Device ID in the manufacturing time. Every ID is configured corresponding

to an Index. The initial Index value is 0.

Editor’s Notes: How to address the synchronizaiton issue, e.g. when one AIoT device is connected to multi Readers, how to sync among those Readers on the HASH values of device IDs, is FFS.

.

Editor’s Notes: It is FFS how the procedure is integrated into inventory procedure, e.g. how the Reader report the device ID to the AF, when the inventory procedure is triggered.

Step 0b. the Application Function sends the Device IDs and Indexes to the AIoTF through NEF.

Step 0c. AIoTF sends the corresponding Device IDs to each Reader based on distribution policy from Application Function or local policy from MNOs.

Step 0d. Reader stores the Devices IDs and the corresponding Indexes under this Reader.

Step 1. Reader sends the paging message to the AIoT devices. Reader sends E(ID1), E(ID2), … in the paging message. The E() Function is HMAC, the key used is the root key K.

NOTE 1: According to RAN2 agreement in #116, the paging message may contain one ID (one Device ID or one group ID) or more (multiple IDs FFS in RAN2) IDs. The procedure may be updated based on RAN2 progress.

NOTE 2: IDm-n indicates the n-th ID calculation using HASH for Device m. ID1-0 indicates the original device ID. ID1-1 = HASH (ID1-0  ||(Index = 1))

Step 2: AIoT Device 1 verifies the IDs being paged, if verification is successful and one of the device IDs maps its ID, AIoT Device 1 confirms it was paged. Then Device 1 replies with ID1-1||Index to the Reader. Index should be 1 in this message, meaning this is the first time the device ID1 was HASHed.

NOTE 3: The IDm-n  sent uplink doesn’t have to be sequencial,. For example, the device could send the ID1-1||(Index=1), it can then send ID1-3||(Index=3) after ID1-1||(Index=1).

Step 3: Reader compares the ID1-1 with its HASH table, then confirm this ID1-1 is in its data base.

Step 4: Reader echoes back the ID1-1 to device, following RAN2 procedure.

Step 5: Device checks ID1-1 is correct, use ID1-2 in next message, in which ID1-2 = HASH (ID1-1||(Index =2))

NOTE 4: Reader shall use unused ID for each device for the next paging if there is any.

### 6.28.3 Evaluation

The impact on the AIoT device and the network are as following:

The AIoT device needs to support to:

1. maintain the root key,
2. decrypt the encrypted device ID using the root key,
3. calculate the HASH (device ID||Index)

The network needs to support to:

1. store the keys for each device ID,
2. paging using encrypted device ID
3. search the HASH table to verify the device ID sent uplink

Editor’s Note: Further evaluation is FFS.

## 6.29 Solution #29: Providing a network-computed AIoT concealed device identifier (AICI) to an AIoT device

### 6.29.1 Introduction

This solution addresses KI#3: Privacy by protecting AIoT device identifiers.

### 6.29.2 Solution details

The solution proposes a method for the 5G network to compute a AIoT Concealed Device Identifier (AICI) and provide the SUCI to the AIoT device in a command message. Once an AIoT device is identified, e.g., after a successful completion of inventory procedure, the network can send a command message on the downlink channel to the AIoT device. In the downlink command message, the network includes an AICI, which is computed based on the long-term identifier of the AIoT device using the public key of the network. The downlink command message is both confidentiality and integrity protected using keys derived from a shared key between the network and the AIoT device. Figure 6.29.2-1 presents a high-level message flow of the solution.



Figure 6.29.2-1: Procedure for delivering a AICI to an AIoT Device

The figure is self-explanatory, therefore, the steps are not explained step-by-step. Exact content of the messages exchanged, and details about authentication challenge, computing response to the challenge, and deriving the keys Kenc and Kint are not described because these details have to be adjusted with the authentication protocol that is finally agreed.

NOTE 1: If the AIoT device does not have a network-computed AICI, for example, in the very first time of the device’s life cycle, then the AIoT device computes AICI using null scheme. This happens only in the beginning. To avoid using null scheme in the first time, a network can choose to provision every AIoT device with a network-computed AICI before they are handed out to their users.

Editor’s Note: The content of the paging message and how the device decides to respond to the paging message based on the content of the paging message is FFS.

NOTE 2: AICI is not stored in the network. Instead, the network decrypts AICI. On the other hand, a device accepts an AICI only if it is computed by the legitimate home network — hence a device cannot obtain an AICI that the network won’t recognize. Therefore, the question about AICI synchronization is not relevant.

NOTE 3: The solution requires AIoT devices to have the capability to update and store AICI.

Editor’s Note: Whether AICI is to be computed and decrypted in SIDF/UDM is FFS

### 6.29.3 Evaluation

The solution fulfills the potential security requirement in KI#3 in the context of the uplink messages.

The solution requires an NF in the 5GC to compute AICI on behalf of the AIoT device.

The solution requires AIoT devices to have the capability to update and store AICI.

Editor’s Note: Further evaluation is FFS.

## 6.30 Solution #30: Privacy protection for inventory operation

### 6.30.1 Introduction

This solution addresses Key issue #3: Privacy by protecting AIoT device identifiers.

The principle of this solution is that the key used to protect inventory operations is pre-configured in AIoT devices. These keys are shared among multiple AIoT devices.

This solution assumes that the capabilities and complexity of AIoT devices may be different. Therefore, security policies are supported in the AIoT Device operation messages so that flexible message security capabilities can be supported.

This solution provides two procedures:

- Inventory procedure with unprotected inventory request parameters, and

- Inventory procedure with protected inventory request parameters, which can address the privacy issues related to groups of devices.

### 6.30.2 Solution details

#### 6.30.2.1 Inventory procedure with unprotected inventory request parameters

The inventory procedure with unprotected inventory request parameters is shown in the following figure.

Figure 6.30.2-1: Inventory procedure with unprotected inventory request parameters

0. During the device initialization phase, the Device ID protection keys are pre-configured in the AIoT device. The Device ID protection keys are used to protect Device ID during inventory process. A key ID for these keys may also be pre-configured in order to facilitate key management. A security policy for AIoT Device ID protection may also be pre-configured.

1. The AF sends an inventory operation request to the Ambient IoT Function (AIoTF). The AF may also provide Device ID Matching Information that is a filter criteria or an AIoT Device ID MASK information used to address only specified AIoT Devices.

2. The AIoTF generates a Network Nonce and sends Inventory request to the AIoT Devices through the AIoT Reader. The Inventory request includes the Device ID Matching Information and Network Nonce.

3. The AIoT Device that matches the Device ID Matching Information performs the following operations:

- Generate a Device Nonce;

- Use the pre-configured Device ID protection keys, Network Nonce and Device Nonce to derivate keys for Device ID confidentiality and/or integrity protection;

- Use new derived keys to protect the AIoT Device ID according to the AIoT Device ID protection policy (security policy);

- The AIoT Device sends Inventory response to the AIoTF through the AIoT Reader. The response includes Security policy, Device Nonce, protected AIoT Device ID and the key ID.

4. The AIoTF uses the same method as the AIoT Device to derive the keys, and then decrypts and/or verifies the protected AIoT Device ID.

5. The AIoTF, AIoT Devices and UDM continue to perform steps 4-7 on the remaining AIoT Devices.

6. The AIoTF returns the Inventory result to the AF.

#### 6.30.2.2 Inventory procedure with protected inventory request parameters

The inventory procedure with protected inventory request parameters is shown in the following figure.

Figure 6.30.2-2: Inventory procedure with protected inventory request parameters

0. Same as step 0 in clause 6.30.2.1.

1. Same as step 1 in clause 6.30.2.1.

2. The AIoTF performs the following operations:

- Generate a Network Nonce;

- Use the Device ID protection keys and Network Nonce to derivate keys for confidentiality and/or integrity protection;

- Use new derived keys to protect the AIoTF Device ID Matching Information according to the AIoT Device ID protection policy (security policy);

The protected AIoT device ID matching information may also be a temporary ID generated using security key and security parameters through HMAC algorithm.

- Send Inventory request to the AIoTF Devices through AIoT Reader. The request includes protected AIoT Device ID Matching Information, Security policy, Network Nonce, and the key ID .

3. The AIoT Device performs the following operations:

- Check if the key ID in the request matches one of the locally stored keys.

- Use the same method as the UDM to derive the keys, and then decrypts and/or verifies the protected AIoT Device ID Matching Information;

If temporary ID is used, the locally stored key and parameters received from the request are used to calculate the local temporary ID.

- Check if its Device ID/Group ID matches the Device ID Matching Information;

If temporary ID is used, check if the local temporary ID is equal to the temporary ID in the request.

The other operations are the same as step 3 in clause 6.30.2.1.

4. The remaining steps and operations are the same as steps 6-9 in clause 6.30.2.1.

### 6.30.3 Evaluation

This solution addresses the Key Issue #3.

This solution uses pre-configured keys to protect AIoT Device IDs during inventory operations. The pre-configured key can be shared among all AIoT Devices or a group of AIoT devices. The pre-configured keys can also be used on the network side to protect the privacy of Device ID matching information.

The key ID may leak privacy information related to a group of AIoT Devices.

In order to protect data privacy, for the inventory procedure with protected inventory request parameters requires all the AIoT Devices to perform cryptograph computing to resolve AIoT Device ID identification. It may result the power consumption to the AIoT Devices that are not the target AIoT Devices.

Editor’s Note: Further evaluation is FFS.

## 6.31 Solution #31: Ambient IoT ID privacy

### 6.31.1 Introduction

This solution addresses KI#3 and proposes a solution to protect the privacy of the identifier of an Ambient IoT (AIoT) device. The solution focuses on the functionality in the AIoT device:

- verifying a request to share the identity of the AIoT device,

- protecting the identity of the AIoT device, when sharing it.

The procedure fits the baseline RAN procedure agreed in RAN2#125bis:

*Step A: Based on the service request, the reader sends the Initial Trigger Message indicating device(s) that need to respond;*

*Step B: Triggered device(s) performs the random access-like procedure, if needed;*

*Step C: The device may perform the data communication with the reader as needed.*

### 6.31.2 Solution details

A diagram of a company

Description automatically generated

Figure 6.31.2-1: Ambient IoT ID privacy

In step 0, AIoT device(s) are configured with identity(ies) and device specific parameters, such as security credentials including device- and group-specific credentials.

In step 1, the AIoT Management Function (AIoT MF) sends a message to the reader to trigger an inventory procedure including an inventory request message and a configuration associated with the inventory procedure. The configuration includes timers (e.g., to broadcast the request and to collect responses from AIoT device(s)), response aggregation requirements and forwarding criteria, etc.

Note 1: The AIoT Management Function (AIoT MF) depends on the entity that owns or manages the resource (i.e., AIoT device). If a device is owned or managed by the network, the disabling is triggered by a CN function, otherwise, it is triggered by a 3rd party Management Function managing the device.

In step 2, the reader UE/gNB sends the inventory request message, as received in step 1, to AIoT device(s). The inventory request includes a device (or group) identifier and a device-specific (or group-specific) challenge. Based on the configuration received, the reader UE/gNB may start a timer T1 to collect responses.

In step 3, The AIoT device processes the received request and checks whether the message:

3.1 is addressed to the AIoT device by matching its identifier (i.e., device or group identifier) against the received identifier, and

3.2 is sent by a trusted party by verifying the challenge value using its security credentials and identifiers.

If the checks 3.1 and 3.2 succeed, the AIoT computes a pseudonym based on its identifier and the received challenge value. This can be implemented by means of a lightweight function, e.g., a keyed hash function.

In step 4, the AIoT device sends its pseudonym in a response message (e.g., an inventory response message).

In step5, based on the configuration received in step 1, the reader UE/gNB aggregates the response message(s) received, until the time T1 runs out and/or other forwarding criteria (e.g., all device responses are received) are met.

In step 6, the reader UE/gNB forwards the response(s) to the AIoT MF.

### 6.31.3 Evaluation

This solution addresses KI#3: Privacy by protecting AIoT device identifiers, in scenarios where a specific device or devices belonging to a group is/are inventoried.

This solution describes a lightweight solution for privacy protection of AIoT device(s) identity(ies).

The solution incorporates a challenge generated by the AIoT management function (e.g., NF/AF) using device or group specific credentials and sent together with the device or group identifier in the inventory request.

An AIoT device needs to verify the identifier(s) and challenge value received in the paging message to check whether it is concerned with the inventory request, and only if the verification is successful does it generate a pseudonym and sends a response.

## 6.32 Solution #32: Authentication Using L1 Parameter

### 6.32.1 Introduction

This solution addresses KI #5 and KI #4. It proposes an authentication solution using L1 parameters for “inventory and command case”.

The random number generation utilizing the reciprocity and randomness of wireless channels to generate shared keys includes three main stages: channel feature extraction, channel feature quantization and key negotiation[10].

Channel feature parameters are L1 measurements e.g., the reference signal received power (RSRP), channel state information (CSI), channel phase (CP) etc. The key generation process of the physical layer key mainly quantizes the measured wireless channel information into keys, and this process does not have high and complex computationally algorithms [11].

### 6.32.2 Solution details

Authentication solution for Inventory and Command Case using L1 measurement to generate a random number.



**Figure 6.32.2-1 authentication solotion using L1 parameters for “inventory and command case”**

1. AIoT NF triggers inventory. AIoT NF could be triggerd by AF.

2. Reader sends R2D Inventory paging message.

3， AIoT device extracts L1 measurements from Inventory paging message, and uses L1 measurements e.g. L1-RSRP to generate RAND\_L1, a random number.

4. AIoT device sends inventory response, including authentication request message, and device ID.

Note that RAND\_L1 is not needed to be sent over the air interface, because reader can generate the same random number.

5. Reader extracts L1 measurements from the inventory response message and uses L1 measurements e.g. L1-RSRP to generate the same RAND\_L1 as AIoT device.

6. Reader sends authentication request message, including device ID and RAND\_L1 to AIoT NF.

7. AIoT NF sends authentication request message, including device ID and RAND\_L1 to AIoT Authentication Function.

8. AIoT authentication function sends authentication vector request to UDM/ARPF for AIoT device.

9. UDM/ARPF generate a random number: RAND. It calculates Kiot with K (e.g. the root key of AIoT device), RAND\_L1, Device ID and RAND.

10. UDM/ARPF calculates MAC with K, RAND\_L1, and RAND.

11. UDM/ARPF sends RAND, MAC, device ID and Kiot to AIoT Authentication Function.

12. AIoT Authentication Function sends Authentication Response including RAND, MAC, device ID and Kiot to AIoT NF.

13. AIoT NF sends Authentication Response including RAND, MAC, device ID to Reader.

14. Reader sends Authentication Response including RAND and MAC to AIoT device in the command message.

15. AIoT device calculates XMAC with RAND, RAN\_L1and K. It verifies XMAC=MAC.

16. AIoT device calculates Kiot with K, RAND\_L1, Device ID and RAND.

Editor’s Note: Whether DL frequency and UL frequency need to be the same is FFS.

Editor’s Note: L1 measurements in this solution is contingent and subject to RAN1 work.

Editor’s Note: Where is the security credential storage and key derivation in Network is FFS.

Editor’s Note: It is FFS whether the key generation for AIoT devices in close proximity result in the same key.

### 6.32.3 Evaluation

TBA.

## 6.33 Solution #33: L1 Security Key Generation

### 6.33.1 Introduction

This solution addresses KI#4. It proposes a L1 security key generation soluton for “inventory and command case”.

Physical layer key generation utilizing the reciprocity and randomness of wireless channels to generate shared keys includes three main stages: channel feature extraction, channel feature quantization and key negotiation [10].

Channel feature parameters are L1 measurements e.g., the reference signal received power (RSRP), channel state information (CSI), channel phase (CP) etc. The key generation process of the physical layer key mainly quantizes the measured wireless channel information into keys, and this process does not have high and complex computationally algorithms [11].

### 6.33.2 Solution details



**Figure 6.33.2-1 Security Key Derivation using L1 measurements for “inventory and command case”**

1. AIoT NF triggers inventory. AIoT NF could be triggerd by AF.

2. Reader sends R2D Inventory paging message.

3. AIoT device extracts L1 measurements from Inventory paging message, and uses L1 measurements e.g. L1-RSRP to generate K\_L1, a physical layer key.

4. AIoT device sends inventory response, inclusing device ID.

Note that K\_L1 is not needed to be sent over the air interface, because reader can generate the same key.

5. Reader extracts L1 measurements from the inventory response message and uses L1 measurements e.g. L1-RSRP to generate the same K\_L1 as AIoT device.

6. Reader sends device ID and K\_L1 to AIoT NF.

7. AIoT NF sends device ID and K\_L1 to UDM/ARPF.

8. UDM/ARPF calculates Kiot with K, K\_L1, and Device ID.

9. UDM/ARPF sends device ID and Kiot to AIoT NF.

10. AIoT NF sends command message which is protected by Kiot.

11. Reader sends command message protected by Kiot to AIoT device.

12. AIoT device calculates Kiot with K, K\_L1, Device ID.

13. AIoT device uses Kiot to decipher or verify command message, depends on the protection methods.

Editor’s Note: Whether DL frequency and UL frequency need to be the same is FFS.

Editor’s Note: L1 measurements in this solution is contingent and subject to RAN1 work.

Editor’s Note: Where is the security credential storage and key derivation in Network is FFS.

Editor’s Note: It is FFS whether the key generation for AIoT devices in close proximity result in the same key.

### 6.33.3 Evaluation

TBA.

## 6.34 Solution #34: PHY key based protecting AIoT device identifiers

### 6.34.1 Introduction

The solution addresses the security requirement of KI#3: Privacy by protecting AIoT device identifiers. To prevent the AIoT deivce ID from being exposed in the air interface, the solution uses the generated physical layer key between AIoT devices and BS/UE Reader to protect the AIoT IDs.

Physical layer key generation utilizing the reciprocity and randomness of wireless channels to generate shared keys includes three main stages: channel feature extraction, quantization and key negotiation [10]. Channel feature parameters are L1 measurements, e.g., the reference signal received power (RSRP), channel state information (CSI), channel phase (CP), etc. The key generation process of the physical layer key mainly quantizes the measured wireless channel information into keys, and this process does not have high and complex computationally algorithms [11].

The solution focuses on the AIoT Inventory operation. Specifically, the AIoT device matches the partial AIoT device ID will performs random-access procedure to establish the AS connection with Reader. During random-access procedure, the AIoT device and Reader extract channel features respectively, and use channel features to generate the shared PHY key to protects AIoT permanent ID.

### 6.34.2 Solution details

**

**Figure 6.34.2-1: PHY key based protecting AIoT device identifiers**

1. The AF sends the AIoT Inventory Request to the AIoT NF via the NEF.

2. The AIoT NF triggers the BS/UE readers to perform AIoT Inventory operation towards the AIoT Devices.

3. The BS/UE Reader sends the AIoT Paging Request to the AioT device.

4. The AIoT device performs the PHY key generation:

a. The AIoT device measures the L1-RSRP according the AIoT Paging Request message

b. The AIoT device quantifies and calculates the PHY key using the measured L1-RSRP.

c. The AIoT device protects its permanent ID with the generated PHY key.

5. The AIoT device sends the AIoT Paging Response to the BS/UE Reader, including the protected AioT Device ID.

6. The BS/UE Reader performs the PHY key generation and obtain the AIoT ID:

a. The BS/UE Reader measures the L1-RSRP according the received the AIoT Paging Response message.

b. The BS/UE Reader quantifies and calculates the PHY key using the measured L1-RSRP.

c. The BS/UE Reader obtains the AIoT device ID with the generated PHY key.

7. The BS/UE Reader forwards the AIoT device ID to the AIoT NF.

8. The AIoT NF sends the AIoT device ID to the AF via the NEF.

Editor’s Note: Whether DL frequency and UL frequency need to be the same is FFS.

Editor’s Note: L1 measurements in this solution is contingent and subject to RAN1 work.

Editor’s Note: it is FFS whether the key generation for AIoT devices in close proximity result in the same key.

Editor’s Note: How to set up measurements and adapt to paging procedure is FFS.

Editor’s Note: The evaluation on the risk of exposing AIOT ID to the Reader is FFS.

### 6.34.3 Evaluation

TBA.

## 6.35 Solution #35: Configurable device/network authentication, data confidentiality, integrity and id privacy protection

### 6.35.1 Introduction

The solution proposes a solution to KI#3, KI#4 and KI#5. It gives a proposal to how a AIoT device can establish a secure connection with a reader, within the RAN procedure to resolve contention. This minimizes the signaling between the AIoT device and reader and thereby also the signaling overhead related to security establishment.

### 6.35.2 Solution details

The solution details how a shared session key can be derived while providing the means for mutual authentication (if required), confidentiality, integrity, reply and privacy protection. The explicit procedure is shown in figure 1. The solution is based on the 3-step contention based random access.

A screenshot of a computer screen

Description automatically generated

Figure 6.35-1: Security context establishment based on 3-step contention based random access

1. The AIoT device and AIoTF is provisioned with a share secret K and a configuration of which primitives to enable like replay protection, integrity protection, privacy protection etc. Furthermore, the AIoT device contains a non-volatile counter and the AIoTF keeps track of the counter.
2. An AF requests the inventory from a AIoT device.
3. The AIoTF draws a random nonce1 and uses the nonce to derive a new key k\_AIoT from the shared secret. K\_AIoT is used to encrypt the device ID and optionally the counter value if reply protection is required. Please note that the same device ID produces different ciphertexts due to the nonce, hereby addressing the privacy concerns.
   1. The Protected Id and nonce1 is send to the AIoT device.
4. The AIoT device derives k\_AIoT and decrypts the protected ID. If the device ID doesn’t match the AIoT device ID rest of the steps can me omitted. If replay protection is enabled, the AIoT device compares the internal counter with count. If the counter values don’t match, the AIoT device can decide to omit rest of the steps or return the correct counter value to get the AIoTF back to sync (Please note that the counter is protected with the session key k\_AIoT\_ses). The decision is taken based on configuration and if the counter is s returned, the procedure restarts at step 3. The AIoT device draws a random number Nonce2 and uses Nonce2 and k\_AIoT to derive the session key k\_AIoT\_ses. Using the session key, the device creates a MAC of nonce1 called ARES. If authentication of the network isn’t required, the AIoT device ID can be encrypted/integrity protected using the session key. The step finalises by incrementing the non-volatile counter.
   1. ARES and Nonce2 and optional protected counter and protected ID are sent to the AIoTF.
5. The AIoTF derives the session k\_AIoT\_ses and calculates the MAC ARES\*. If ARES equals ARES\* the device is authenticated. This step can be omitted if device authentication isn’t required. The AIoTF creates the MAC NRES using Nonce2 and the session key as input. If network authentication isn’t required, calculation of NRES can be omitted. If counter is returned form the AIoT device, the AIoTF re-adjust the internal counter and restarts at step 3.
   1. NRES is send to the AIoT device.
6. The AIoT device calculates NRES\* and compares it to NRES. If the values are equal, the AIoT device have authenticated the network. The device encrypts/integrity protects the AIoT device ID

13-14. The protected ID is sent to the AIoTF.

15. The reader might acknowledge the AIoT device – This is part of the contention based random access procedure.

16-17. The AIoTF decrypts and verifies the integrity of the Protected ID and send it to the AF,

The key k\_AIoT\_ses can further be used to confidentiality and integrity protect command messages. If confidentiality and integrity protection isn’t required, NULL ciphering and integrity protection algorithms can be used.

Editor’s note: How the solution addresses a group of devices including key identification in step 9 is FFS.

Editor’s note: How the constrains of the device influence the capability to preform cryptographic operations and maintain the counter is FFS.

Editor's Note: It is FFS how to prevent power and resource exhaustion in the AIoT devices when all of devices in the paging area always have to decrypt the Device ID.

Editor’s note: How the counter is resynced when the AIoTF and device gets out for sync is FFS.

Editor’s note: The solution needs to be aligned with RAN2 conclusions.

NOTE: According to TR 38.769 [9] legacy is paging is not support and the baseline for the random access procedure is slotted-ALOHA.

NOTE: The device will maintain state as long backscatter is applied, which implies if backscatter is removed, only the configuration, key K and count maintain state. Procedure starts at step 3, when backscatter is reapplied.

### .6.35.3 Evaluation

The solution fulfils the requirements of KI#3 by protecting the device identifier such same device ID produces different ciphertexts in each paging message, hereby avoiding ciphertext attack. Furthermore, when the real device ID is transferred from the device to the AIoTF, the ID is protected by a session key, implying the same ID is producing different ciphertexts for each session, hereby also avoiding ciphertext and likability attacks.

The solution fulfills the requirements of KI#4 by providing the means for confidentiality, integrity and replay protecting. The enablement of the protection profile is based on configuration both in the AIoT device and AIoTF. The solution doesn’t mandate a given profile, but leaves it for deployment and implementation,

The solution fulfills the requirement of KI#5 by providing the means for authentication both one-way and mutual. The selection of authentication scheme is part of configuration of the AIoT and AIoTF.

Editor’s note: Further evaluation is FFS.

## 6.36 Solution #36: AIoT Authentication with dedicated AIoT NF

### 6.36.1 Introduction

To address the requirements in Key issue#5 that the core network entities, if any, that are used for authentication are dedicated for Ambient IoT service, this solution proposes a dedicated AIoT NF to perform authentication for AIoT devices.

### 6.26.2 Solution details

As shown in figure 6.x.1, the authentication is performed by the AIoT device on device side and a dedicated AIoT NF on network side by the dedicated AIoT NF*.*

A diagram of a cloud

Description automatically generated

Figure 6.36. 1 AIoT authentication with dedicated AIoT NF

Editor’s note: The figure needs to be updated regarding the AIoT security domain.

Editor’s note: Whether this solution can be converted to a security architecture clause is FFS.

The AIoT NF can support both one-way and mutual authentication between the AIoT device and the AIoT NF .

If AIoT NF needs to communicate with 5GC, there should be a Security Gateway(SGW) at the border of security domains. The dedicated AIoT NF is in one security domain, can be called AIoT security domain, and the 5GC is in another security domain, called 5GC security domain. The SGW isolates the two domains, and can provide the security properties such as access control, signaling filtering, topology hiding etc.

### 6.36.3 Evaluation

TBD

## 6.37 Solution #37: Mutual Authentication Using AEAD for Inventory and Command case

### 6.37.1 Introduction

The solution addresses KI#5. It proposes a mutual authentication using AEAD between AIoT devices and the network. According to the conclusions in TR 23.700-13, the solution makes the following assumptions:

* An AIoT device shares a permanent identity (i.e., AIoT Device ID) and root key (i.e., K) with the network side or third party (e.g., UDM or Credential Holder's AAA server).

### 6.37.2 Solution details

The message flow of this solution is described below:



Figure 6.37.2-1 Mutual authentication for security credentials stored in the UDM

1-2. Third-party AF sends inventory request to the AIoT function through NEF. The request can include an AIoT Device ID to page the device.

3-4. AIoT function generates a fresh nonce and sends the nonce to the AIoT devices through a Reader.

5. AIoT device calculate KAF and authentication parameters.

5.1 AIoT device computes KAF using key derived function. The input parameters include the K, the Device ID and the nonce.

5.2 The device uses f2 or AES algorithm to compute RES. The input parameters include the K, the identity and the nonce.

5.3 Alternatively, the devices can use AEAD algorithm to compute a message authentication code to protect the authenticity of uplink data (i.e. Device ID). The input parameters include the KAF and uplink data (i.e. Device ID).

6. The device sends inventory response (authentication request) including the identity, RES or message authentication code to the Reader.

7. The Reader forwards the response to the AIoT function, including authentication request.

8. The AIoT function sends the authentication request to the UDM, including nonce.

9. The UDM computes KAF and verifies the RES or message authentication code.

9.1 computes KAF using key derived function. The input parameters include the K, the Device ID and the nonce.

9.2 retrieves the SQN and the K according to the ID.

9.3 computes the anonymity key (AK) using f5 or AES algorithm. The input parameters include the K and the fresh nonce.

9.4 computes MAC using f1 or AES function. The input parameters include the K, the SQN fresh nonce and the command message.

10. The UDM sends KAF, AK⊕SQN and MAC to the AIoT function and increases the SQN by one.

11. The AIoT function stores the KAF corresponding to the identity and sends the Command request to the Reader, the message includes AK⊕SQN and MAC.

12. The Reader forwards the Command request to the AIoT device.

13. The AIoT device computes the AK in the same way as the UDM and decrypts the concealed SQN. If the SQN is valid, then it verifies MAC.

14 – 17. Optionally AIoT device sends command response message.

Editor’s Note: How SQN is validated is FFS.

Editor’s Note: Use of Device ID in Inventory (i.e., paging) and whether it applies to a single or a group of AIoT devices is FFS.

Editor’s Note: Impact of using UDM on the network for credential storage and key generation is FFS.

Editor’s Note: How AES AEAD mode is used in the AKA procedure is FFS.

### 6.37.3 Evaluation

TBD.

## 6.38 Solution #38: Authentication and privacy of AIoT device

### 6.38.1 Introduction

This solution addresses key issue #3: Privacy by protecting AIoT device identifiers and key issue #5: Authentication in Ambient IoT service.

It is assumed that an AIoT device has power or computational resource limitation.

In this solution, encryption is used to protect AIoT device identifier (AIoT ID) and one-way authentication (i.e., network authenticating AIoT device) is performed.

Editor’s Note: Whether one-way authentication is sufficient is FFS.

It is assumed that AIoT device and AF are provisioned with AIoT ID and a key (K).

Editor’s Note: Alignment with conclusion from TR 23.700-13 [4] is FFS.

### 6.38.2 Solution details



Figure 6.38.2-1 Security procedure for AIoT

0. AIoT ID and a key (K) are provisioned to the AIoT device and AF. AF generates KAIoT from K and RANDAF.

1. AF sends AIoT service request message to AIoTF (AIoT Function). This message may be sent via NEF. AIoT service request message includes command, AIoT ID, RANDAF, and KAIoT.

2. AIoTF generates RANDAIoTF. AIoTF derives encryption key (Kenc) and integrity key (Kint), and then encrypts AIoT ID using RANDAIoTF and Kenc.3. AIoTF sends AIoT Service Request including RANDAIoTF, RANDAF, encrypted AIoT ID, and command.

NOTE: If necessary, command may be encrypted using Kenc.

Editor’s Note: Possibility of replay and spoofing attack at step 3 is FFS.

Editor’s Note: How AIoTF knows which AIoT device to send the request is FFS.

4. The AIoT device derives KAIoT from K and RANDAF. The AIoT device derives Kenc and Kint from KAIoT, before decrypting the encrypted AIoT ID. If the decrypted ID is same as the AIoT ID, the AIoT device processes the command.

5. AIoT device responds with AIoT service response. The message includes encrypted AIoT ID, command response, and MAC. MAC is generated from Kint, RANDAIoTF, encrypted ID, and command response.

NOTE: If necessary, command response may be encrypted using Kenc.

Editor’s Note: Possibility of replay and spoofing attack at step 5 is FFS.

6. After AIoTF finds Kenc and Kint from the encrypted AIoT ID, AIoTF checks the authenticity of the message by verifying the MAC. If the verification is successful, AIoTF decrypts the encrypted IE(s) and proceeds to step 7. If the verification fails, AIoTF does not allow the AIoT device to use the network provided by the operator.

7. AIoTF sends AIoT service response to AF. This message includes AIoT ID and command response.

For A-IoT device paging functionality, it is understood that the legacy paging message, legacy paging occasion and legacy DRX from NR are not supported (See TS 38.300 [R2-3] for references for any legacy NR functionality). From RAN2 perspective, it is assumed that the A-IoT device can receive as long as there is enough energy

Editor’s Note: Impact of protected ID on paging occasion is FFS.

Editor's Note: It is FFS how to prevent power and resource exhaustion in the AIoT devices when all of devices in the paging area always have to decrypt the Device ID.

### 6.38.3 Evaluation

TBD

## 6.39 Solution #39: reuse of existing authentication frameworks

### 6.39.1 Introduction

This solution addresses Key Issue #5.

This solution applies to topology 1 and topology 2. The solution takes into account the presence of RAN reader and AIoTF.

### 6.39.2 Solution details

The following authentication frameworks specified in TS 33.501 [12] can be used to perform authentication between an AIoT device and AUSF/UDM (or AAA-S): 5G AKA, EAP-AKA’, EAP-TLS or any other key-generating EAP-method.

The choice of the authentication framework depends on the deployment scenario and is compliant with TS 33.501 [12].

RAN reader and AIoTF do not perform any cryptographic computation of the authentication procedure.

The key hierarchy resulting from the authentication procedure is adapted to AIoT service, e.g. there is no key derived for handover and mobility.

Editor’s Note: It is FFS whether the key hierarchy resulting from the authentication could be further reduced depending on the conclusions of other key issues, e.g. key issue #4 "Protection of information during AIoT service communication".

### 6.39.3 Evaluation

This solution addresses Key Issue #5 by reusing existing authentication frameworks.

Editor’s Note: It is FFS whether the solution applies to all types of AIoT devices.

Editor’s Note: It is FFS whether the number of device interactions are feasible for AIoT.

Editor’s Note: Since all AIoT devices require a subscription, the impact to the UDM is FSS.

Editor’s Note: Impact to the RAN paging message is FFS.

Editor’s Note: It is FFS how the existing mechanisms referred to in this solution fit within RAN2 procedures.

## 6.40 Solution #40: Communication security for reading all information from AIoT device

### 6.40.1 Introduction

This solution addresses KI#4 for the case that network wants to read all information from an AIoT device.

The AIoT device may be a simple device that only contains limited information. The AIoT AF may need to read all information from an AIoT device in this case. This solution proposes to optimize the signaling procedure in this case.

### 6.40.2 Solution details

The following figure shows the call flow for protecting communication for reading all information from AIoT device.



Figure 6.40.2-1: Communication protection for reading all information from AIoT device

0. The UE reader has registered into 5G network with primary authentication performed successfully. The algorithms are preconfigured in AIoT device, e.g., during manufacture phase.

1. The AIoT AF sends AIoT Read Request to the AIoT NF/AMF via the NEF. The filter for memory reading indicates all, i.e., reading all memory information.

2. The AIoT NF/AMF instructs paging with read indication and a NONCENW to the UE reader via CP or via a PDU Session, or to the RAN reader.

3. The RAN reader or UE reader broadcasts an AIoT paging message for the AIoT device with read indication and the NONCENW, which is used by all the AIoT devices that are paged.

4. The AIoT device determines to responds to the AIoT paging message, it detects that the AIoT paging message contains a read indication and a NONCENW, then generates a NONCEDV and derives a KAIoT-enc and a KAIoT-int based on root key, the NONCEDV, and the NONCENW. The AIoT device integrity and confidentiality protects the information and sends an AIoT Information message with the protected payload and the NONCEDV to the UE reader or the RAN reader. The UE reader or the RAN reader forwards the AIoT Information message to the AIoT NF/AMF.

5. The AIoT NF/AMF sends an AIoT Key Request with the KSI indicating the root key used, the NONCEDV and the NONCENW to the AAA-S.

6. The AAA-S derives a KAIoT-enc and a KAIoT-int as the same way the AIoT device computed based on the root key identified by the KSI, the NONCEDV, and the NONCENW.

7. The AAA-S sends an AIoT Key Response with the KAIoT-enc and the KAIoT-int to the AIoT NF/AMF.

8. The AIoT NF/AMF performs integrity check and decryption on the protected payload based on the received keys, and if integrity check succeeds, the AIoT NF/AMF sends an AIoT Read Response with the plain text payload to the AIoT AF via NEF.

Editor’s Note: Clarification and potential risk (e.g. replay attack) evaluation on step 3 is ffs.

Editor’s Note: Clarification on key derivation is ffs.

Editor’s Note: Clarification on KSI usage is ffs.

### 6.40.3 Evaluation

TBA.

## 6.41 Solution #41: Disabling protection for AIoT device

### 6.41.1 Introduction

This solution addresses KI#1 to protect the command for permanently or temporarily disabling an AIoT device. The solution can be based on Topology 1 and 2.

### 6.41.2 Solution details

### 6.41.2.1 Disable an AIoT device permanently or temporarily

The following figure shows the call flow for protecting command for permanently or temporarily disabling an AIoT device.



Figure 6.41.2.1-1: Disable an AIoT device temporarily or permanently

0. The UE reader has registered into 5G network with primary authentication performed successfully.

1. The AIoT AF sends AIoT Disable Request for disabling an AIoT device with disable type (i.e., permanent or temporary) to the AIoT NF/AMF via the NEF.

2. The AIoT NF/AMF instructs paging with status indication to the UE reader via CP or via a PDU Session, or to the RAN reader.

3. The RAN reader or UE reader broadcasts an AIoT paging message for the AIoT device with status indication.

4. The AIoT device determines to responds to the AIoT paging message, it detects that the AIoT paging message contains a status indication, then generates a NONCEDV and sends an AIoT Status Request with the NONCEDV to the UE reader or the RAN reader. The UE reader or the RAN reader forwards the AIoT Status Request to the AIoT NF/AMF. The AIoT NF/AMF generates a NONCENW. The AIoT NF/AMF sends an AIoT Key Request with the NONCEDV and the NONCENW to the AAA-S, the AAA-S derives a KAIoT based on the NONCEDV, and the NONCENW, then sends an AIoT Key Response with the KAIoT to the AIoT NF/AMF.

5. The AIoT NF/AMF sends an AIoT Status Response with the NONCENW and the type to the AIoT device via the UE reader or the RAN reader. The AIoT Status Response is integrity protected by the KAIoT. The AIoT device derives the KAIoT as the same way the AAA-S computed and performs integrity check on the AIoT Status Response, which implicitly authenticate the network.

6. If the integrity check succeeds, the AIoT device may send an AIoT Disable Confirm message with success indication to the AIoT NF/AMF via the UE reader or the RAN reader. The AIoT Disable Confirm message is integrity protected by the KAIoT. The AIoT device disables the RF permanently or still turns on the receiver temporarily according to the type received.7. If the AIoT Disable Confirm message is received, the AIoT NF/AMF verifies the AIoT Disable Confirm message. The AIoT NF/AMF sends an AIoT Disable Response indicating disable success to the AIoT AF, which may be based on whether the verification is successful.

Editor’s Note: Clarification on step 3 is ffs.

Editor’s Note: Evaluation on step 4a is ffs, e.g. whether step 4a is new message.

#### 6.41.2.2 Enable a temporarily disabled AIoT device

The following figure shows the call flow for protecting command for enable an AIoT device, which is temporarily disabled.



Figure 6.41.2.2-1: Enable a temporarily disabled AIoT device

0-3. Same as described in clause 6.41.2.1 step 0-3 with the difference that the AIoT AF may send key derivation parameter to the AIoT NF/AMF, as well as if the AIoT device is temporarily disabled and determines to handle the AIoT paging message with status indication, it enables the RF temporarily.

NOTE: Only temporarily disabled AIoT devices is able to respond to the AIoT paging with status indication.

4-5. Same as described in clause 6.41.2.1 step 4-5 with the difference that the AIoT NF/AMF may send key derivation parameter to the AIoT device via the UE reader or the RAN reader. The AIoT device derives new KAIoT-status based on the key derivation parameter and KAIoT for consequence disable/enable commands.

6. If the integrity check succeeds, the AIoT device enables the RF permanently and sends an AIoT Enable Confirm message with success indication to the AIoT NF/AMF via the UE reader or the RAN reader. The AIoT Enable Confirm message is integrity protected by the KAIoT.

7. The AIoT NF/AMF verifies the AIoT Enable Confirm message, if verification succeeds, the AIoT NF/AMF sends an AIoT Enable Response indicating enable success to the AIoT AF.

Editor’s Note: Clarification on step 3b is ffs.

6.41.3 Evaluation

TBA.

## 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 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.

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.

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

10. 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.

11. 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.

12. 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: Procedure to update set of nonces is FFS

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

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

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

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

## 6.Y Solution #Y: <Solution Name>

### 6.Y.1 Introduction

Editor’s Note: Each solution should list the key issues being addressed.

### 6.Y.2 Solution details

### 6.Y.3 Evaluation

Editor’s Note: Each solution should motivate how the potential security requirements of the key issues being addressed are fulfilled.

# 7 Conclusions

Editor’s Note: This clause contains the agreed conclusions that will form the basis for any normative work.

Annex <X> (informative):  
Change history

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Change history** | | | | | | | |
| **Date** | **Meeting** | **TDoc** | **CR** | **Rev** | **Cat** | **Subject/Comment** | **New version** |
| 04/2024 | SA3#115Adhoc-e | S3-241476 |  |  |  | Initial draft TR | 0.0.0 |
| 04/2024 | SA3#115Adhoc-e | S3-241648 |  |  |  | Incorporated accepted contributions S3-241477, S3-241622, S3-241630, S3-241636 | 0.1.0 |
| 05/2024 | SA3#116 | S3-242536 |  |  |  | Incorporated accepted contributions S3-242649, S3-242534, S3-242535, S3-242539, S3-242540, S3-242541 | 0.2.0 |
| 08/2024 | SA3#117 | S3-243699 |  |  |  | Incorporated accepted contributions S3-243492, S3-243680 | 0.3.0 |
| 10/2024 | SA3#118 | S3-243828 |  |  |  | Incorporated accepted contributions S3-244124, S3-244366, S3-234460, S3-244367, S3-244368. S3-244369, S3-244370, S3-244449, S3-244450, S3-244451, S3-244452, S3-244453, S3-244454, S3-244455, S3-244456, S3-244457, S3-244458, S3-244459, S3-244508, S3-244509, S3-244467, S3-244468, S3-244469, S3-244470, S3-244477, S3-244478, S3-244479, S3-244480, S3-244483, S3-244484, S3-244485, S3-244486, S3-244487, S3-244488, S3-244489 | 0.4.0 |
| 11/2024 | SA3#!19 | S3-245169 |  |  |  | Incorporated accepted contributions S3-244757, S3-244795, S3-244960, S3-244970, S3-245018, S3-245043, S3-245063, S3-245064, S3-245065, S3-245066, S3-245067, S3-245068, S3-245069, S3-245070, S3-245071, S3-245172, S3-245173, S3-245174, S3-245175, S3-245177, S3-245178, S3-245296, S3-245297, S3-245298, S3-245299, S3-245300, S3-245301, S3-245302, S3-245303, S3-245304, S3-245305, S3-245306, S3-245307, S3-245308, S3-245309, S3-245310, S3-245311, S3-245312, S3-245313, S3-245314, S3-245315, S3-245316, S3-245344 | 0.5.0 |