INTERNATIONAL STANDARD

ISO/IEC 29167-11

Second edition
2023-02

Information technology — Automatic identification and data capture techniques —

Part 11: Crypto suite PRESENT-80 security services for air interface communications

Technologies de l'information — Techniques automatiques d'identification et de capture de données —

Partie 11: Interface radio pour services sécurité – Suite cryptographique PRESENT-80 security services for air interface communications

Reference number
ISO/IEC 29167-11:2023(E)

Contents

Page

Foreword

SO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives or www.iec.ch/members_experts/refdocs).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents) or the IEC list of patent declarations received (see https://patents.iec.ch).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see www.iso.org/iso/foreword.html. In the IEC, see www.iec.ch/understanding-standards.

This document was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 31, *Automatic identification and data capture techniques*.

This second edition cancels and replaces the first edition (ISO/IEC 29167-11:2014), which has been technically revised.

The main changes are as follows:

— the Interrogator authentication and Tag-Interrogator mutual authentication has been added;

— the variant of PRESENT that uses a 128-bit key has been added.

A list of all parts in the ISO/IEC 29167 series can be found on the ISO and IEC websites.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html and www.iec.ch/national-committees.

Introduction

The International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC) draw attention to the fact that it is claimed that compliance with this document may involve the use of a patent.

ISO and IEC take no position concerning the evidence, validity and scope of this patent right.

The holder of this patent right has assured ISO and IEC that he/she is willing to negotiate licences under reasonable and non-discriminatory terms and conditions with applicants throughout the world. In this respect, the statement of the holder of this patent right is registered with ISO and IEC. Information may be obtained from the patent database available at www.iso.org/patents or https://patents.iec.ch.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights other than those in the patent database. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

Information technology — Automatic identification and data capture techniques —

Part 11: Crypto suite PRESENT-80 security services for air interface communications

1 Scope

This document defines the crypto suite for PRESENT-80 for the ISO/IEC 18000 series of air interfaces standards for radio frequency identification (RFID) devices. This document provides a common crypto suite for security for RFID devices for air interface standards and application standards. The crypto suite is defined in alignment with existing air interfaces.

This document specifies basic security services that are use the lightweight block cipher PRESENT-80. The variant of PRESENT that takes 128-bit keys is also considered in this document.

This document defines various methods of use for the cipher.

A Tag and an Interrogator can support one, a subset, or all of the specified options, clearly stating what is supported.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 18000 (all parts), *Information technology — Radio frequency identification for item management*

ISO/IEC 19762, *Information technology — Automatic identification and data capture (AIDC) techniques — Harmonized vocabulary*

ISO/IEC 29167-1, *Information technology — Automatic identification and data capture techniques — Part 1: Security services for RFID air interfaces*

3 Terms, definitions, symbols and abbreviated terms

3.1 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 19762 and the following apply.

ISO and IEC maintain terminology databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at https://www.electropedia.org/

3.1.1
bit string
ordered sequence of 0's and 1's

3.1.2
block cipher
family of permutations parameterized by a *cryptographic key* (3.1.3); permutations map bit strings of a fixed length given by the block size to bit strings of the same length

3.1.3
cryptographic key
string of bits of a specified length that is used by the *block cipher* (3.1.2) to transform a *data block* (3.1.4)

3.1.4
data block
string of bits whose length is given by the block size (3.1.3) of the *block cipher* (3.1.2)

3.1.5
PRESENT-*k*-ENC(key, data)
PRESENT encryption of **data**, a 64-bit *data block* (3.1.4), using **key**, a *k*-bit *cryptographic key* (3.1.3)

3.1.6
PRESENT-*k*-DEC(key, data)
PRESENT decryption of **data**, a 64-bit *data block* (3.1.4), using **key**, a *k*-bit *cryptographic key* (3.1.3)

3.1.7
salt
string of bits, typically randomly generated, that is used to diversify a cryptographic computation

3.2 Symbols

$XXXX_2$	Binary notation
$XXXX_h$	Hexadecimal notation
||	Concatenation of syntax elements, transmitted in the order written
Ø	Empty string, typically used to indicate a deliberately empty input or omitted field
|A|	Bit-wise length of the string A expressed as an integer EXAMPLE 1 $\|0000_2\| = 4$ EXAMPLE 2 $\|0000_h\| = 16$ EXAMPLE 3 |Ø| = 0
Field [a:b]	Selection of bits from a string of bits denoted Field The selection ranges from bit "a" through to, and including, bit "b" where Field [0] represents the least significant or rightmost bit. EXAMPLE 1 Field [2:0] represents the selection of the three least significant bits of Field. EXAMPLE 2 Field, without a specified range, indicates the entirety of Field. EXAMPLE 3 Field [-1:0] is an alternative representation of the empty string Ø.
Field [~]	Non-empty string constructed from a string of bits denoted Field
Key.KeyID	Cryptographic key identified and indexed by the numerical value KeyID

3.3 Abbreviated terms

CS	Crypto Suite
CSI	Crypto Suite Indicator
IA	Interrogator authentication
IChallenge	Interrogator challenge
MA	Mutual Authentication
RFU	Reserved for future use
RFID	Radio frequency identification
TA	Tag authentication
TID	Tag Identification number

4 Conformance

4.1 Air interface protocol specific information

An Interrogator or Tag shall comply with all relevant clauses of this document, except those marked as “optional”.

Relevant conformance test methods are provided in ISO/IEC 19823-11[1].

4.2 Interrogator conformance and requirements

The Interrogator shall implement the mandatory commands defined in this document and conform to the relevant part of the ISO/IEC 18000 series.

The Interrogator can implement any subset of the optional commands defined in this document.

The Interrogator shall not:

— implement any command that conflicts with this document; or

— require the use of an optional, proprietary or custom command to meet the requirements of this document.

4.3 Tag conformance and requirements

The Tag shall implement the mandatory commands defined in this document for the supported types and conform to the relevant part of the ISO/IEC 18000 series.

The Tag can implement any subset of the optional commands defined in this document.

The Tag shall not:

— implement any command that conflicts with this document; or

— require the use of an optional, proprietary or custom command to meet the requirements of this document.

5 Introduction of the PRESENT-80 cryptographic suite

PRESENT-80 is a block cipher that uses an 80-bit key and is designed to be suitable for extremely constrained environments such as RFID Tags.

The details of the operation of the PRESENT-80 cipher are provided in Annex C. The background to the development of PRESENT-80 and its design principles are described in Reference [3].

A variant of PRESENT-80, denoted PRESENT-128, supports keys of length 128 bits.

Guidance on the appropriate variant to use in a given application lies outside the scope of this document. A thorough security and risk assessment is advised before deployment. Errors and error-handling for this cryptographic suite shall be in accordance with Annex B.

Test vectors for parts of this document are provided in Annex D.

Over-the-air protocol commands that use this cryptographic suite shall be in accordance with Annex E.

6 Parameter and variable definitions

Table 1 lists the variables and constants that are used in this document.

Table 1 — PRESENT-80 cryptographic suite variables and constants

Parameter	Description
IChallenge	A random challenge generated by the Interrogator
TChallenge	A random challenge generated by the Tag
TRnd	A random salt value generated by the Tag
IRnd	A random salt value generated by the Interrogator
CTAM	A pre-defined constant set to the two-bit value 00_2
CIAM	A pre-defined constant set to the two-bit value 01_2
CMAM1	A pre-defined constant set to the two-bit value 10_2
CMAM2	A pre-defined constant set to the two-bit value 11_2
PurposeIAM	Purpose bits for Interrogator authentication If PurposeIAM[3:3] = 0_2, the bits [2:0] are RFU with value 000_2. If PurposeIAM[3:3] = 1_2, the bits [2:0] are manufacturer defined.
PurposeMAM	Purpose bits for Mutual authentication If PurposeMAM[3:3] = 0_2, the bits [2:0] are RFU with value 000_2. If PurposeMAM[3:3] = 1_2, the bits [2:0] are manufacturer defined.
Key.0 ... Key.15	A set of up to 16 keys Key.0 through to Key.15 Not all key values need to be specified. However, Key.j shall not be specified when there remains an unspecified Key.i with i<j. If there is only one key on the Tag, it shall be identified as Key.0.

7 Crypto suite state diagram

After power-up and after a reset, the cryptographic engine transitions into the **Initial** state. The cryptographic engine shall follow the state transition in Table A.1. The state progressions defined by the state transition table in Annex A is illustrated in Figure 1.

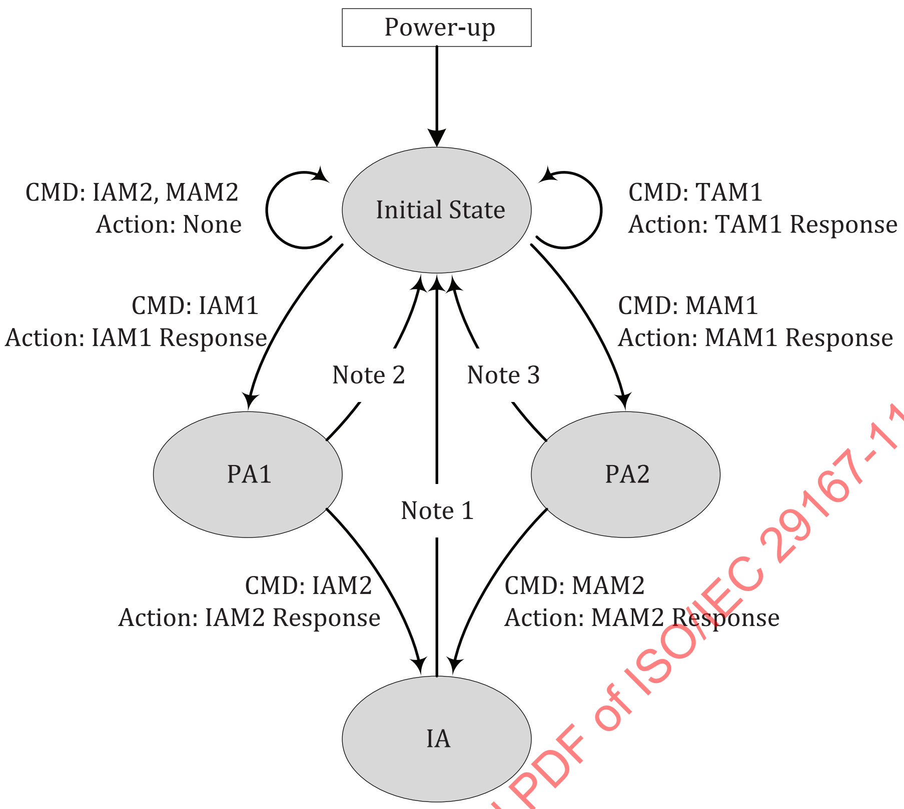


Figure 1 — Cryptographic engine state diagram

NOTE 1 For all of TAM1, IAM1, MAM1, IAM2, MAM2 and errors return to Initial State without action.

NOTE 2 For all of TAM1, IAM1, MAM1, MAM2 and errors return to Initial State without action.

NOTE 3 For all of TAM1, IAM1, MAM1, IAM2 and errors return to Initial State without action.

8 Initialization and resetting

After power-up and after a reset the cryptographic engine transitions into the **Initial** state.

Implementations of this suite shall ensure that all memory used for any intermediate results is cleared

a) after the completion of each cryptographic protocol,

b) if some cryptographic protocol is abandoned or incomplete, and

c) after reset.

9 Authentication

9.1 Introduction

This document supports Tag authentication, Interrogator authentication and Tag-Interrogator mutual authentication.

9.2 Message and response formatting

Messages and responses are part of the security commands described in the air interface specification. The following subclauses of this document describe the formatting of message and response for a

Tag authentication method, an Interrogator authentication method and a Tag-Interrogator mutual authentication method.

Different authentication methods are identified using the AuthMethod field. The values for this field are given in Table 2.

Table 2 — Values and descriptions for AuthMethod[1:0]

Value	Description
00_2	Tag authentication
01_2	Interrogator authentication
10_2	Tag-Interrogator authentication
11_2	Vendor defined

9.3 Tag authentication: AuthMethod "00"

9.3.1 General

Tag authentication uses a challenge-response protocol. This is illustrated in Figure 2.

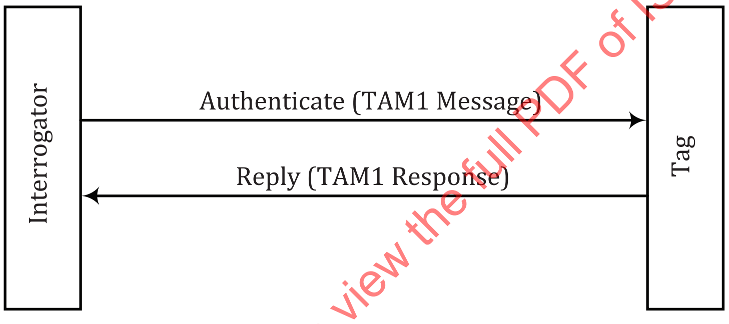


Figure 2 — Tag authentication via a challenge-response scheme

9.3.2 TAM1 message

The Interrogator shall generate a random IChallenge that is carried in the TAM1 message. The format of the TAM1 message is given in Table 3.

The Interrogator may choose optional extended functionality for the TAM1 message. The Interrogator may identify the cryptographic key Key.KeyID that should be used and, for Key.KeyID, whether it is 80 or 128 bits long.

NOTE 1 If a Tag supports anything other than a single 80-bit key, then extended functionality is required.

NOTE 2 Determining Key.KeyID is a matter of key management that lies outside the scope of this document.

NOTE 3 The variant(s) of PRESENT deployed on a device are manufacturer dependent.

NOTE 4 Appropriate mechanisms to generate IChallenge lie outside the scope of this document.

Table 3 — TAM1 message format

Field	Payload							
	Always included					In addition, for the case $E = 1_2$		
	AuthMethod	RFU	Extension	TID	Challenge	Key	KeyLength	E-RFU
Length (bits)	2	2	1	1	42	4	1	3
Value	00_2	00_2	E	T	IChallenge	KeyID	L	000_2
The fields shaded in gray are present only if E = 1_2.								

9.3.3 Intermediate Tag processing

The Tag shall accept the TAM1 message at any time (unless occupied by internal processing and not capable of receiving messages); i.e. upon receipt of the message with valid parameters the Tag shall abort any cryptographic protocol that has not yet been completed and shall remain in the **Initial** state.

A Tag conforming to this document shall support the case of AuthMethod=00_2.

The actions of the Tag when AuthMethod=00_2 are defined in this subclause. Actions of the Tag for other values of AuthMethod are defined in 9.4 and 9.5.

If the value of the field RFU is not 00_2, the Tag shall set a "Not Supported" crypto suite error.

The Tag shall parse the value of E.

— If E=0_2:

  — Extended functionality is not requested by the Interrogator.

  — An 80-bit key, for which Key.ID=0, shall be used for Tag authentication.

  — The TAM1 message payload has length 48 bits.

— If E=1_2:

  — Extended functionality is requested by the Interrogator.

  — The fields Key, KeyLength, and E-RFU (see below) reveal the extended functionality requested by the Interrogator.

  — The TAM1 message payload has length 56 bits.

— A Tag shall support at least one of the options E=0_2 or E=1_2. The Tag shall set a "Not Supported" crypto suite error for a value of E that is not supported by the Tag.

The Tag shall parse the value of T.

— If T=0_2, the Interrogator is requesting the output from the cryptographic operation. The reply to the TAM1 message consists of the output from the cryptographic operation.

— If T=1_2, the Interrogator is requesting some or all of the TID in addition to the output from the cryptographic operation.

  — If the Tag supports the option T=1_2 then the reply to the TAM1 message consists of some or all of the TID concatenated with the output from the cryptographic operation. There is no restriction on the number, selection or location of TID bits returned. The choice and specification is manufacturer defined.

  — If the Tag does not support T=1_2 then the Tag shall set a "Not Supported" crypto suite error.

— A Tag shall support the option T=0_2 and may support the option T=1_2.

If present, the Tag parses KeyID, identifying the choice of cryptographic key.

— The Tag shall use Key.KeyID to compute TAM1 response.

— If the Tag does not support Key.KeyID then the Tag shall set a "Not Supported" crypto suite error.

If present, the Tag parses L, the value of KeyLength.

— If $L=0_2$ then PRESENT-80 shall be used in the computation of TAM1 response. If Key.KeyID does not have length 80 bits, then the Tag shall set a "Not Supported" crypto suite error.

— If $L=1_2$ then PRESENT-128 shall be used in the computation of TAM1 response. If Key.KeyID does not have length 128 bits, then the Tag shall set a "Not Supported" crypto suite error.

If present, the Tag checks that the value of the field $\text{E-RFU}=000_2$. If not, the Tag shall set a "Not Supported" crypto suite error.

9.3.4 TAM1 response

If the TAM1 message is successfully parsed by the Tag, the Tag shall generate a random salt TRnd of length 20 bits.

The Tag shall use Key.KeyID of length k and PRESENT-k encryption to form a 64-bit string TResponse:

$$\text{TResponse} = \text{PRESENT-}k\text{-ENC} (\text{Key.KeyID}, \text{CTAM} \,||\, \text{TRnd} \,||\, \text{IChallenge}).$$

NOTE 1 In the case that $E=0_2$, KeyID takes the default value 0 and the only key on the Tag – Key.0 – is an 80-bit key.

NOTE 2 Appropriate mechanisms to generate TRnd lie outside the scope of this document.

NOTE 3 Only one input block of 64 bits is encrypted and so only one invocation of PRESENT is required.

The TAM1 Response to the Interrogator is given in Table 4.

Table 4 — TAM1 Response format

Field	**Payload**	
	Case $T=0_2$ **Response**	**Case** $T=1_2$ **Response**
Length (bits)	64	65 to 160 (manufacturer dependent)
Value	TResponse	TID[~] || TResponse

9.3.5 Final Interrogator processing

After receiving TAM1 response, the Interrogator shall use Key.KeyID to compute the 64-bit string R:

$$R = \text{PRESENT-}k\text{-DEC} (\text{Key.KeyID}, \text{TResponse}).$$

a) The Interrogator shall check that R[41:0] = IChallenge.

b) The Interrogator should check that R[63:62] = CTAM.

If these verification steps are successfully completed, the Interrogator concludes that the Tag and Interrogator possess matching values of Key.KeyID. When combined with an appropriate key management scheme – the definition of which falls outside the scope of this document – the Interrogator concludes that the Tag is authentic.

9.4 Interrogator authentication: AuthMethod “01”

9.4.1 General

Interrogator authentication uses a challenge-response protocol. This is illustrated in Figure 3.

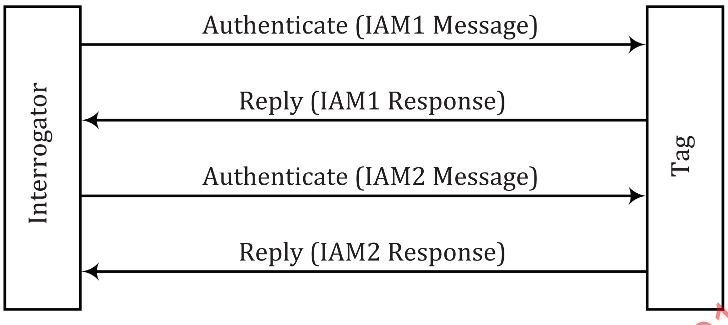


Figure 3 — Interrogator authentication via a challenge-response scheme

9.4.2 IAM1 message

The Interrogator sends message IAM1 to the Tag thereby prompting the Tag to start a challenge-response exchange. The format of the IAM1 message is given in Table 5.

NOTE 1 The key(s) deployed on a device are manufacturer dependent.

NOTE 2 Interrogator authentication requires the use of a 128-bit key.

NOTE 3 The number of key(s) deployed on a device are manufacturer dependent.

NOTE 4 Determining Key.KeyID is a matter of key management that lies outside the scope of this document.

Table 5 — IAM1 message format

Field	**Payload**			
	AuthMethod	**Step**	**RFU**	**Key**
Length (bits)	2	2	4	4
Value	01_2	00_2	0000_2	KeyID

9.4.3 Intermediate Tag processing #1

The Tag shall accept this message at any time (unless occupied by internal processing and not capable of receiving messages); i.e. upon receipt of the message with valid parameters the Tag shall abort any cryptographic protocol that has not yet been completed and shall remain in the **Initial** state.

NOTE A Tag conforming to this document can support the case of AuthMethod=01_2.

If Interrogator authentication is not supported on the Tag, i.e. if "01_2" is not a valid value for AuthMethod, then the Tag shall set a "Not Supported" crypto suite error.

If the value of Step is not "00_2" then the Tag shall set a "Not Supported" crypto suite error.

If the value of RFU is not "0000_2" then the Tag shall set a "Not Supported" crypto suite error.

The Tag shall use Key.KeyID to process message IAM2. If the Tag does not support Key.KeyID, or if Key.KeyID does not have length 128 bits, then the Tag shall set a "Not Supported" crypto suite error.

9.4.4 IAM1 response

If the IAM1 message is successfully parsed by the Tag, the Tag shall generate a random challenge TChallenge of length 42 bits and send this to the Interrogator. The IAM1 response is illustrated in Table 6.

Table 6 — IAM1 response format

Field	**Payload**
	Challenge
Length (bits)	42
Value	TChallenge

9.4.5 Intermediate Interrogator processing

The Interrogator sends the message IAM2 to the Tag.

9.4.6 IAM2 message

The Interrogator shall generate a random salt IRnd of length 16 bits.

The Interrogator shall use Key.KeyID of length 128 bits and PRESENT-128 decryption to form a 64-bit string IResponse:

IResponse = PRESENT-128-DEC (Key.KeyID, CIAM || PurposeIAM || IRnd || TChallenge).

The Interrogator sends IResponse to the Tag as part of the IAM2 message, see Table 7.

NOTE 1 Only one input block of 64 bits is encrypted and so only one invocation of PRESENT-128 is required.

NOTE 2 Appropriate mechanisms to generate TRnd lie outside the scope of this document.

Table 7 — IAM2 message format

Field	**Payload**			
	AuthMethod	**Step**	**RFU**	**InterrogatorResponse**
Length (bits)	2	2	4	64
Value	01_2	01_2	0000_2	IResponse

9.4.7 Intermediate Tag processing #2

The Tag shall only accept the IAM2 message when the cryptographic engine is in state **PA1** (see Clause 7).

If Interrogator authentication is not supported on the Tag, i.e. if "01_2" is not a valid value for AuthMethod, then the Tag shall set a "Not Supported" crypto suite error.

If the value of Step is not "01_2" then the Tag shall set a "Not Supported" crypto suite error.

If the value of RFU is not "0000_2" then the Tag shall set a "Not Supported" crypto suite error.

The Tag shall use Key.KeyID to compute the 64-bit string R:

R = PRESENT-128-ENC (Key.KeyID, IResponse).

a) The Tag shall check that R[41:0] = TChallenge.

b) The Tag should check that R[63:62] = CIAM.

If the checks performed by the Tag are successful, then the Tag concludes that the Tag and Interrogator possess matching values of Key.KeyID. When combined with an appropriate key management scheme – the definition of which falls outside the scope of this document – the Tag concludes that the Interrogator is authentic and TStatus is set to 1_2. Otherwise TStatus is set to 0_2.

The Tag shall set PurposeIAM = R[61:58].

9.4.8 IAM2 response

The Tag shall return the value of TStatus to the Interrogator using the IAM2 response illustrated in Table 8.

— If TStatus = 1_2 then the cryptographic state machine moves to the state **IA** (see Clause 7).

— If TStatus = 0_2 then the cryptographic state machine moves to the **Initial** state (see Clause 7).

Table 8 — IAM2 response format

Field	**Payload**	
	Status	**RFU**
Length (bits)	1	3
Value	TStatus	000_2

9.4.9 Final Interrogator processing

The Interrogator receives IAM2 Response.

If the value of TStatus is 1_2 then the Interrogator assumes that the Tag is in the state **IA** (see Clause 7).

If, under conditions laid out in the over-the-air protocol, there is no response from the Tag or if the returned value of TStatus is 0_2 then the Interrogator shall abandon the cryptographic protocol.

NOTE The RFU field will be available in a future version of this document.

9.5 Mutual authentication: AuthMethod “10”

9.5.1 General

Mutual Interrogator-Tag authentication uses an interleaved challenge-response protocol. This is illustrated in Figure 4.

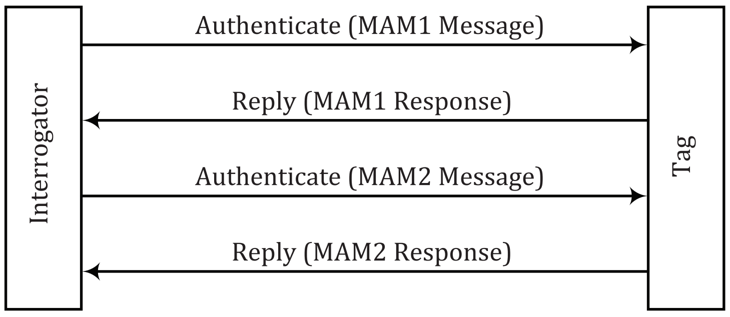


Figure 4 — Interrogator-Tag mutual authentication via an interleaved challenge-response scheme

9.5.2 MAM1 message

The Interrogator shall generate a random 42-bit challenge IChallenge that is carried in the MAM1 message. The format of the MAM1 message is illustrated in Table 9.

NOTE 1 The variant(s) of PRESENT deployed on a device are manufacturer dependent.

NOTE 2 Mutual authentication requires the use of a 128-bit key.

NOTE 3 The key(s) deployed on a device are manufacturer dependent.

NOTE 4 Determining Key.KeyID is a matter of key management and lies outside the scope of this document.

NOTE 5 Appropriate mechanisms to generate IChallenge lie outside the scope of this document.

Table 9 — MAM1 message format

Fields	Payload				
	AuthMethod	Step	RFU	Key	Challenge
Length (bits)	2	2	4	4	42
Value	10_2	00_2	0000_2	KeyID	IChallenge

9.5.3 Intermediate Tag processing #1

The Tag shall accept MAM1 message at any time (unless occupied by internal processing and not capable of receiving messages); i.e. upon receipt of the message with valid parameters the Tag shall abort any cryptographic protocol that has not yet been completed and shall remain in the **Initial** state.

NOTE 1 A Tag conforming to this document can support the case of AuthMethod=10_2.

If Tag-Interrogator mutual authentication is not supported on the Tag, i.e. if "10_2" is not a valid value for AuthMethod, then the Tag shall set a "Not Supported" crypto suite error.

If the value of Step is not "00_2", then the Tag shall set a "Not Supported" crypto suite error.

If the value of RFU is not "0000_2" then the Tag shall set a "Not Supported" crypto suite error. The Tag shall use Key.KeyID to process the response to messages MAM1 and MAM2. If the Tag does not support Key.KeyID, or if Key.KeyID does not have length 128 bits, then the Tag shall set a "Not Supported" crypto suite error.

If the MAM1 message is successfully parsed by the Tag, then the Tag shall generate a random challenge TChallenge of length 42 bits.

The Tag shall use Key.KeyID to compute the 64-bit string R:

R = PRESENT-128-ENC (Key.KeyID, CMAM1 || TChallenge [41:22] || IChallenge).

The remainder of TChallenge is concatenated with R to give TResponse:

TResponse = TChallenge [21:0] || R.

NOTE 2 Only one input block of 64 bits is encrypted and so only one invocation of PRESENT is required.

NOTE 3 Appropriate mechanisms to generate TChallenge lie outside the scope of this document.

9.5.4 MAM1 response

The Tag returns the value of TResponse to the Interrogator using the MAM1 response illustrated in Table 10.

Table 10 — MAM1 response format

Field	Payload
	Response
Length (bits)	86
Value	TResponse

9.5.5 Intermediate Interrogator processing

After receiving MAM1 Response the Interrogator shall use Key.KeyID to compute the 64-bit string T:

$$T = \text{PRESENT-128-DEC} \,(\, \text{Key.KeyID}, \text{TResponse}[63{:}0] \,).$$

a) The Interrogator shall check that T[41:0] = IChallenge.

b) The Interrogator should check that T[63:62] = CMAM1.

If these verification steps are not successful, the Interrogator shall abandon the cryptographic protocol. Otherwise the Interrogator concludes that the Tag and Interrogator possess matching values of Key. KeyID. When combined with an appropriate key management scheme – the definition of which falls outside the scope of this document – the Interrogator concludes that the Tag is authentic.

9.5.6 MAM2 message

If the cryptographic protocol has not been abandoned, the Interrogator shall generate a random salt IRnd of length 16 bits.

The Interrogator shall form a 64-bit string IResponse:

IResponse = PRESENT-128-DEC (Key.KeyID, CMAM2 || PurposeMAM || IRnd || T[57:42] || TResponse[85:64]).

The Interrogator sends IResponse to the Tag as part of the MAM2 message, see Table 11.

Table 11 — MAM2 message format

Field	Payload			
	AuthMethod	Step	RFU	Interrogator Response
Length (bits)	2	2	4	64
Value	10_2	01_2	0000_2	IResponse

9.5.7 Intermediate Tag processing #2

The Tag shall only accept this message when the cryptographic engine is in the state **PA2**, see Clause 7.

If Tag-Interrogator mutual authentication is not supported on the Tag, i.e. if "10_2" is not a valid value for AuthMethod, then the Tag shall set a "Not Supported" crypto suite error.

If the value of Step is not “01_2” then the Tag shall set a "Not Supported" crypto suite error.

If the value of RFU is not “0000_2” then the Tag shall set a "Not Supported" crypto suite error.

Assuming that the MAM2 message is successfully parsed by the Tag, the Tag shall compute the 64-bit string S:

$$S = \text{PRESENT-128-ENC} \,(\, \text{Key.KeyID}, \text{IResponse} \,).$$

The Tag shall check that S[41:0] = TChallenge.

The Tag should check that S[63:62] = CMAM2.

If the checks performed by the Tag are successful then the Tag concludes that the Tag and Interrogator possess matching values of Key.KeyID. When combined with an appropriate key management scheme – the definition of which falls outside the scope of this document – the Tag concludes that the Interrogator is authentic and TStatus is set to 1_2. Otherwise, TStatus is set to 0_2.

The Tag shall set PurposeMAM = S[61:58].

9.5.8 MAM2 response

The Tag shall prepare and send a MAM2 response as specified in Table 12.

If TStatus = 1_2 then the cryptographic engine transitions to state **IA** (see Clause 7).

Table 12 — MAM2 response format

Field	Payload	
	Status	RFU
Length (bits)	1	3
Description	TStatus	000_2

9.5.9 Final Interrogator processing

The Interrogator receives MAM2 response.

If the value of TStatus is 1_2 then the Interrogator assumes that the Tag is in the state **IA** (see Clause 7).

If, under conditions laid out in the over-the-air protocol, there is no response from the Tag or if the returned value of TStatus is 0_2 then the Interrogator shall abandon the cryptographic protocol.

NOTE The RFU field will be available in a future version of this document and can be used to provide advanced security services to mutually authenticated devices.

10 Communication

Secure communication techniques are not supported in this document.

11 Key table and Key update

This document does not support the KeyUpdate command.

A Tag can maintain a record for each key in a key table. The format, memory location, size and values of any key table are manufacturer defined.

A key can be identified by a KeyID, the identification number of a key within the key table. The value of KeyID shall start with 0_h and increment by one for each successive key in the table. The value of KeyID shall not exceed F_h.

Annex A
(normative)

Crypto suite state transition table

Table A.1 — Crypto Suite State transition table

Start State	Transition	End State	Action
Initial	TAM1	**Initial**	Send TAM1 Response
Initial	IAM2, MAM2, improper, or faulty command	**Initial**	Cryptographic Suite Error
Initial	IAM1	**PA1**	Send IAM1 Response
Initial	MAM1	**PA2**	Send MAM1 Response
PA1	TAM1, IAM1, MAM1, MAM2, improper, or faulty command	**Initial**	Cryptographic Suite Error
PA1	IAM2	**IA**	Send IAM2 Response
PA2	TAM1, IAM1, MAM1, IAM2, improper, or faulty command	**Initial**	Cryptographic Suite Error
PA2	MAM2	**IA**	Send MAM2 Response
IA	TAM1, IAM1, MAM1, IAM2, MAM2, improper, or faulty command	**Initial**	Cryptographic Suite Error

Annex B
(normative)

Errors and error handling

A Tag that encounters an error during the execution of a cryptographic suite operation may send an error reply to the Interrogator. The details of these error replies are defined in the respective air interface standards.

Table B.1 contains a list of the errors that can result from the operation of this cryptographic suite. Annex E defines how to translate these errors into an error code for the air interface.

Table B.1 — Error conditions

Error	Error condition
Not Supported	Supplied parameter values are either not supported by this cryptographic suite or not supported by this implementation.
Cryptographic suite error	A conflict in the protocol flow has been detected.

Annex C
(informative)

Description of PRESENT

PRESENT is a block cipher with an SPN structure. It can take both 80- and 128-bit keys and it was designed from first principles to be suitable for constrained hardware environments while offering excellent levels of security. The cipher has been widely analyzed in the cryptographic community and it is standardized in ISO/IEC 29192-2[4].

PRESENT was originally published, along with design criteria, in Reference [3].

PRESENT consists of 31 rounds, see Figure C.1. The block length is 64 bits and two key lengths of 80 and 128 bits are supported.

```
generateRoundKeys()
for $i$ = 1 to 31 do
    addRoundKey(state,$K_i$)
    sBoxLayer(state)
    pLayer(state)
end for
addRoundKey(state,$K_{32}$)
```

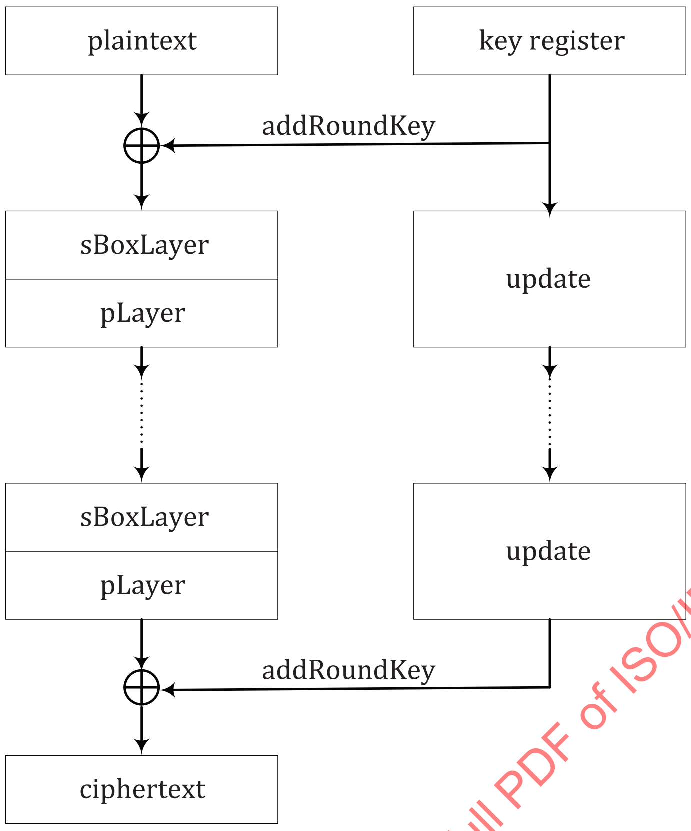


Figure C.1 — Top-level overview of PRESENT

Each of the 31 rounds consists of an xor operation to introduce a round key K_i for $1 \le i \le 32$, where K_{32} is used for post-whitening, a linear bitwise permutation and a non-linear substitution layer. The non-linear layer uses a single 4-bit S-box which is applied 16 times in parallel in each round. Throughout, bits are numbered from zero with bit zero on the right of a block or word. See Figure C.2.

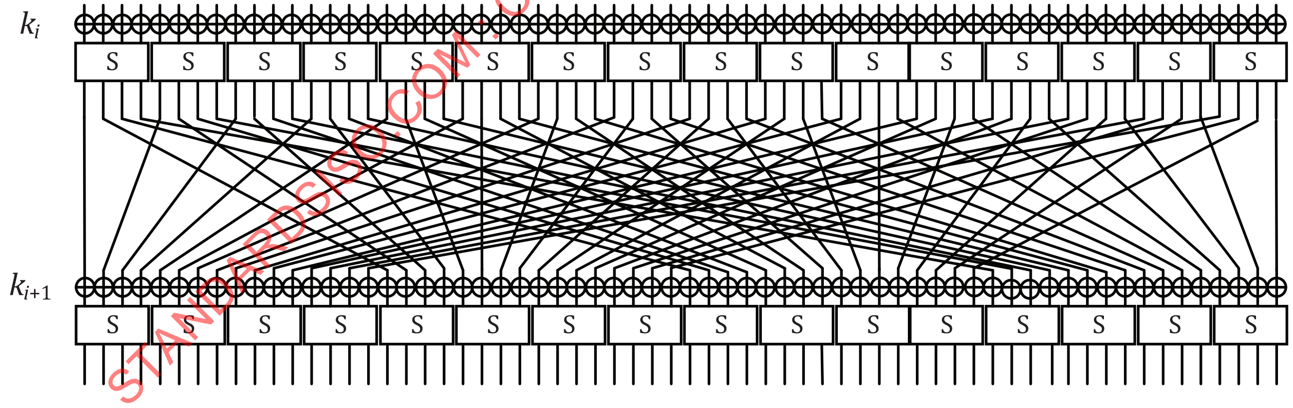


Figure C.2 — Round structure of PRESENT

addRoundKey. Given round key $K_i = \kappa_{63} \dots \kappa_0$ for $1 \le i \le 32$ and current state $b_{63} \dots b_0$, addRoundKey consists of the operation for $0 \le j \le 63$, $b_j \to b_j \oplus \kappa_j$.

sBoxlayer. The S-box used in PRESENT is a 4-bit to 4-bit S-box. The action of this box in hexadecimal notation is given in Table C.1.

Table C.1 — S-box used in PRESENT

Input	0_h	1_h	2_h	3_h	4_h	5_h	6_h	7_h	8_h	9_h	A_h	B_h	C_h	D_h	E_h	F_h
Output	C_h	5_h	6_h	B_h	9_h	0_h	A_h	D_h	3_h	E_h	F_h	8_h	4_h	7_h	1_h	2_h

For sBoxLayer the current state $b_{63} \ldots b_0$ is considered as sixteen 4-bit words $w_{15} \ldots w_0$ where $w_i = b_{4*i+3}$ || b_{4*i+2} || b_{4*i+1} || b_{4*i} for $0 \le i \le 15$ and the output nibble $S[w_i]$ provides the updated state values in the obvious way.

pLayer. The bit permutation used in PRESENT is given by the following table. Bit i of state is moved to bit position $P(i)$. The action of the permutation layer is given in Table C.2.

Table C.2 — Permutation layer used in PRESENT

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
P(i)	0	16	32	48	1	17	33	49	2	18	34	50	3	19	35	51
i	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
P(i)	4	20	36	52	5	21	37	53	6	22	38	54	7	23	39	55
i	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
P(i)	8	24	40	56	9	25	41	57	10	26	42	58	11	27	43	59
i	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
P(i)	22	28	44	60	13	29	45	61	14	30	46	62	15	31	47	63

generateRoundKeys: PRESENT-80.

The user-supplied key is stored in a key register K and represented as $k_{79}k_{78} \ldots k_0$. At round i the 64-bit round key $K_i = \kappa_{63}\kappa_{62} \ldots \kappa_0$ consists of the 64 leftmost bits of the current contents of register K. Thus, at round i we have that $K_i = \kappa_{63}\kappa_{62} \ldots \kappa_0 = k_{79}k_{78} \ldots k_{16}$.

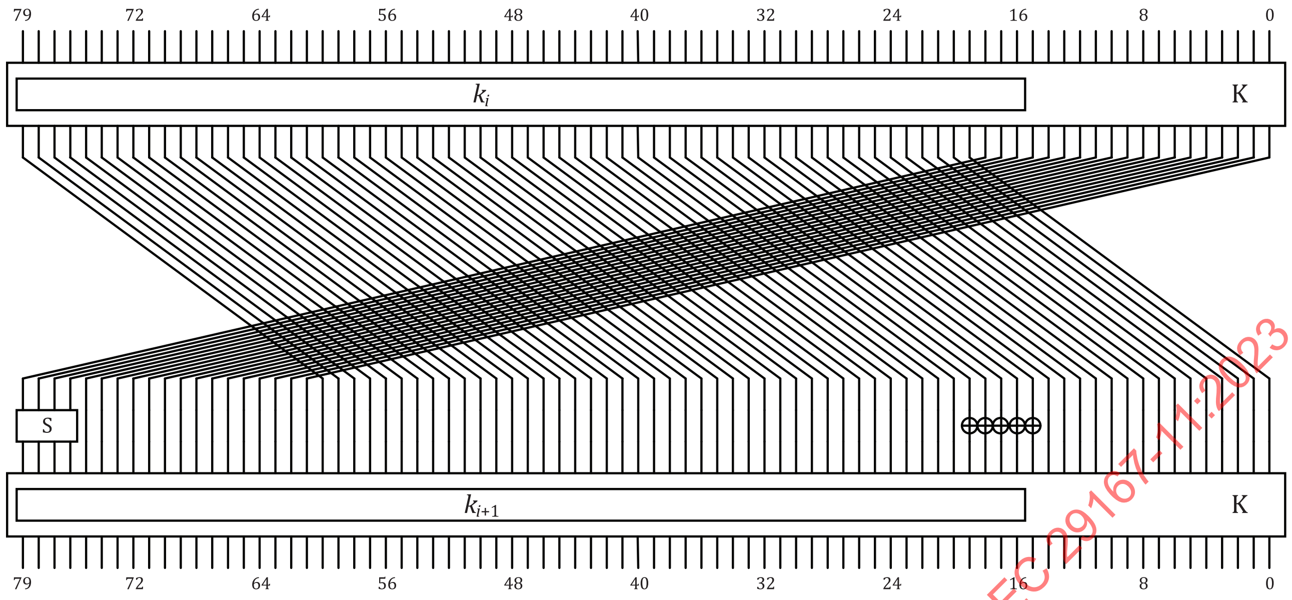
After extracting the round key K_i, the key register $K = k_{79}k_{78} \ldots k_0$ is updated as follows.

a) $[k_{79}k_{78} \ldots k_0] = [k_{18}k_{17} \ldots k_{20}k_{19}]$

b) $[k_{79}k_{78}k_{77}k_{76}] = S[k_{79}k_{78}k_{77}k_{76}]$

c) $[k_{19}k_{18}k_{17}k_{16}k_{15}] = [k_{19}k_{18}k_{17}k_{16}k_{15}] \oplus$ round_counter

Thus, the key register is rotated by 61 bit positions to the left, the left-most four bits are passed through the PRESENT S-box, and the round_counter value i is exclusive-ored with bits $k_{19}k_{18}k_{17}k_{16}k_{15}$ of K with the least significant bit of round_counter on the right. See Figure C.3.



NOTE Round key bits k_i and k_{i+1} are extracted from key register K with k_{i+1} computed from k_i using S-box S and a five-bit round_counter.

Figure C.3 — Key schedule for PRESENT-80

generateRoundKeys: PRESENT-128.

The user-supplied key is stored in a key register K and represented as $k_{127}k_{126}...k_0$. At round i the 64-bit round key $K_i = \kappa_{63}\kappa_{62}...\kappa_0$ consists of the 64 leftmost bits of the current contents of register K. Thus, at round i we have that $K_i = \kappa_{63}\kappa_{62}...\kappa_0 = k_{127}k_{126}...k_{64}$.

After extracting the round key K_i, the key register $K = k_{127}k_{126}...k_0$ is updated as follows.

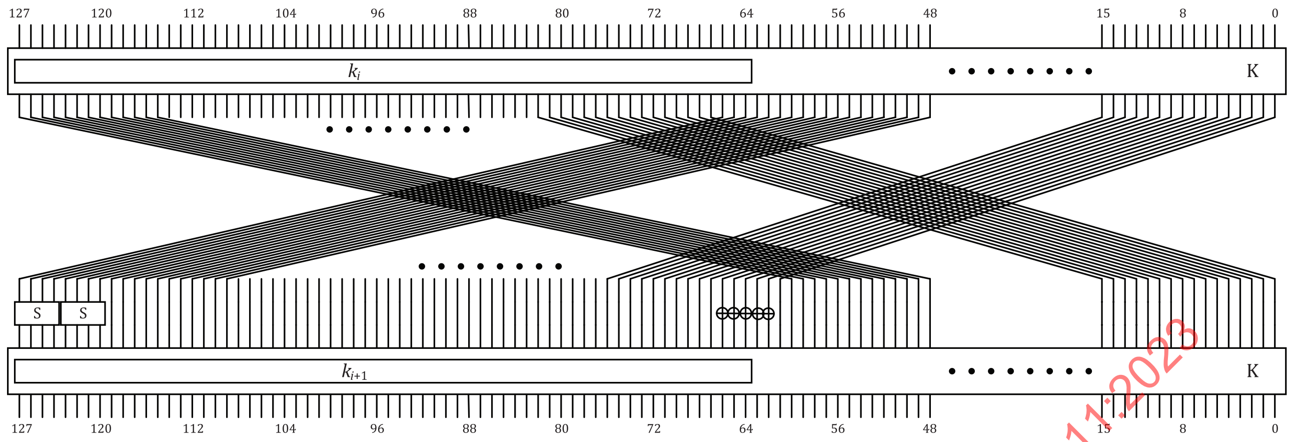
a) $[k_{127}k_{126}...k_1k_0] = [k_{66}k_{65}...k_{68}k_{67}]$

b) $[k_{127}k_{126}k_{125}k_{124}] = S[k_{127}k_{126}k_{125}k_{124}]$

c) $[k_{123}k_{122}k_{121}k_{120}] = \mathrm{S}[k_{123}k_{122}k_{121}k_{120}]$

d) $[k_{66}k_{65}k_{64}k_{63}k_{62}] = [k_{66}k_{65}k_{64}k_{63}k_{62}] \oplus$ round_counter

Thus, the key register is rotated by 61 bit positions to the left, the left-most eight bits are passed through two PRESENT S-boxes, and the round_counter value i is exclusive-ored with bits $k_{66}k_{65}k_{64}k_{63}k_{62}$ of K with the least significant bit of round_counter on the right. See Figure C.4.



NOTE Round key bits k_i and k_{i+1} are extracted from key register K with k_{i+1} computed from k_i using two instances of S-box S and a five-bit round_counter.

Figure C.4 — Key schedule for PRESENT-128