INTERNATIONAL STANDARD

ISO/IEC 29167-14

First edition
2015-10-15

Information technology — Automatic identification and data capture techniques —

Part 14: Crypto suite AES OFB security services for air interface communications

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

Partie 14: Services de sécurité par suite cryptographique AES-OFB pour communications d'interface radio

Reference number
ISO/IEC 29167-14:2015(E)

COPYRIGHT PROTECTED DOCUMENT

ISO copyright office
Ch. de Blandonnet 8 • CP 401
CH-1214 Vernier, Geneva, Switzerland
Tel. +41 22 749 01 11
Fax +41 22 749 09 47
copyright@iso.org
www.iso.org

Contents

Page

Foreword

ISO (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. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

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).

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).

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

For an explanation on the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the WTO principles in the Technical Barriers to Trade (TBT) see the following URL: Foreword - Supplementary information

The committee responsible for this document is ISO/IEC JTC 1, *Information technology*, Subcommittee SC 31, *Automatic identification and data capture techniques*.

ISO/IEC 29167 consists of the following parts, under the general title *Information technology — Automatic identification and data capture techniques*:

— *Part 1: Security services for RFID air interfaces*

— *Part 10: Crypto suite AES-128 security services for air interface communications*

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

— *Part 12: Crypto suite ECC-DH security services for air interface communications*

— *Part 13: Crypto suite Grain-128A security services for air interface communications*

— *Part 14: Crypto suite AES OFB security services for air interface communications*

— *Part 16: Crypto suite ECDSA-ECDH security services for air interface communications*

— *Part 17: Crypto suite cryptoGPS security services for air interface communications*

— *Part 19: Crypto suite RAMON security services for air interface communications*

The following parts are under preparation:

— *Part 15: Crypto suite XOR security services for air interface communications*

— *Part 20: Air interface for security services — Cryptographic Suite Algebraic Eraser*

Introduction

This part of ISO/IEC 29167 describes a cryptographic suite that is applicable to the ISO/IEC 18000 standard. The ISO/IEC 18000 series of standards on RFID for item management do not contain any strong cryptographic security. The unique item identifier (UII) of tags is transmitted during the identification/singulation process to every reader that is able to communicate according to the standard. Sensitive data that are communicated from the interrogator, such as passwords and certain data written to memory, could be cover-coded with a one-time pad obtained from the tag. The tag sends this one-time pad over the air in plain text allowing an attacker to easily intercept all communications. Additionally, passwords are limited in length, providing limited security for the system. This part of ISO/IEC 29167 will fill this security gap for applications requiring a high level of security. Furthermore, it is applicable to applications requiring a large amount of data to be communicated between interrogators and tags.

This part of ISO/IEC 29167 covers the air interface for RFID tags that have a security module on board and its corresponding interrogators. Any other means of security is not addressed in this part of ISO/IEC 29167. A security module according to this part of ISO/IEC 29167 is either a means to provide read or write access limitations, password protection or a crypto engine. The use of a crypto engine is the typical case and all others are less likely.

The International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC) draw attention to the fact that it is claimed that compliance with this part of ISO/IEC 29167 can involve the use of patents concerning radio-frequency identification technology given in the clauses identified below.

ISO and IEC take no position concerning the evidence, validity and scope of these patent rights.

The holders of these patent rights have assured the ISO and IEC that they are willing to negotiate licences under reasonable and non-discriminatory terms and conditions with applicants throughout the world. In this respect, the statements of the holders of these patent rights are registered with ISO and IEC.

Information on the declared patents may be obtained from:

Contact details	
Patent holder:	
Electronics Telecommunication Reseach Institute	
Contact for license application:	
Name & Department:	Ickchan, Lee, Intellectual Property Management Team
Address:	138 Gajeongno, Yuseong-gu
Address:	Daejeon, 305-700, Korea
Tel.	+82-42-860-6904
Fax	+82-42-860-3831
E-mail	ickchanlee@etri.re.kr
URL (optional)	www.etri.re.kr
Patent Holder:	
Legal Name	Impinj, Inc.
Contact for license application:	
Name & Department	Stacy Jones
Address	701 N. 34th Street, Suite 300
Address	Seattle, WA 98103, USA
Tel.	+1.206 834 1032
Fax	+1.206 517 5262

E-mail	stacy.jones@impinj.com
URL (optional)	www.impinj.com

The latest information on IP that might be applicable to this part of ISO/IEC 29167 can be found at www.iso.org/patents.

Information technology — Automatic identification and data capture techniques —

Part 14: Crypto suite AES OFB security services for air interface communications

1 Scope

This part of ISO/IEC 29167 defines the cryptographic suite for AES using OFB mode (AES OFB) for the ISO/IEC 18000-63 air interface standard for radio frequency identification (RFID) devices. Its purpose is to provide a common cryptographic suite for security for RFID devices that can be referenced by ISO committees for air interface standards and application standards.

This part of ISO/IEC 29167 specifies a cryptographic suite for AES OFB for air interface for RFID systems. The cryptographic suite is defined in alignment with existing air interfaces.

This part of ISO/IEC 29167 defines various authentication methods and 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 Conformance

2.1 Claiming conformance

To claim conformance with this part of ISO/IEC 29167, an interrogator or tag shall comply with all relevant clauses of this part of ISO/IEC 29167, except those marked as “optional”.

2.2 Interrogator conformance and obligations

To conform to this part of ISO/IEC 29167, an interrogator shall implement the mandatory commands defined in this part of ISO/IEC 29167, and conform to ISO/IEC 18000.

To conform to this part of ISO/IEC 29167, an interrogator may implement any subset of the optional commands defined in this part of ISO/IEC 29167.

To conform to this part of ISO/IEC 29167, the interrogator shall not implement any command that conflicts with this part of ISO/IEC 29167, or require the use of an optional, proprietary, or custom command to meet the requirements of this part of ISO/IEC 29167.

2.3 Tag conformance and obligations

To conform to this part of ISO/IEC 29167, a tag shall implement the mandatory commands defined in this part of ISO/IEC 29167 for the supported types, and conform to ISO/IEC 18000.

To conform to this part of ISO/IEC 29167, a tag may implement any subset of the optional commands defined in this part of ISO/IEC 29167.

To conform to this part of ISO/IEC 29167, a tag shall not implement any command that conflicts with this part of ISO/IEC 29167, or require the use of an optional, proprietary, or custom command to meet the requirements of this part of ISO/IEC 29167.

3 Normative references

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 18000-63, *Information technology — Radio frequency identification for item management — Part 63: Parameters for air interface communications at 860 MHz to 960 MHz Type C*

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

4 Terms and definitions

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

4.1
AES OFB mode
output feedback mode using a block cipher AES

Note 1 to entry: It makes a synchronous stream cipher of a block cipher. In this part of ISO/IEC 29167, the key stream generated by AES OFB mode is described as the keystream in order to emphasize the key hierarchy consisting of master key and keystream.

4.2
ciphertext
usable data that are formatted as output from a mode

4.3
plaintext
usable data that are formatted as input to a mode

5 Symbols and abbreviated terms

5.1 Symbols

C_j	*j*-th ciphertext block
$DEC_K(X)$	decryption function under the key *K* applied to the data block *X*
$ENC_K(X)$	encryption function under the key *K* applied to the data block *X*
j	index to a sequence of data blocks or data segments ordered from left to right
K	secret key
n	number of data blocks or data segments in the plaintext
P_j	*j*-th plaintext block
||	concatenation of syntax elements, transmitted in the order written

5.2 Abbreviated terms

AES	Advanced Encryption Standard
CSI	Cryptographic Suite Identifier

IV Initialization Vector

MK Master Key

OFB Output FeedBack

UII Unique Item Identifier

XOR eXclusive OR

6 Cipher introduction

6.1 General

The Advanced Encryption Standard (AES) algorithm is a symmetric block cipher standardized as ISO/IEC 18033-3. The Output Feedback (OFB) mode is a confidentiality mode that features the iteration of the forward cipher on an Initialization Vector (IV) to generate a sequence of output blocks that are exclusive-ORed with the plaintext to produce the ciphertext, and vice versa.

6.2 Encryption in AES OFB mode

Figure 1 shows the AES OFB encryption process. The encryption process shall be implemented as the XOR operation between the plaintext and the keystream. For a given tuple of n plaintext messages P_1, ..., P_n and a keystream K, the encryption process is defined by $ENC_K(P_1, ..., P_n) = (P_1 \text{ XOR } K_1, ..., P_n \text{ XOR } K_n)$, where K_j is the *j-th* block of the keystream K and has the same bit-length as P_j. The decryption process is exactly the same with the encryption process.

The operation mode is the AES OFB mode. The encryption process shown in Figure 1 operates as if it is a synchronous stream cipher. In addition, encryption process and decryption process are exactly the same because of the symmetry of the XOR operation. Therefore, a tag can perform both of encryption and decryption with only one encryption module. This OFB mode has an advantage of decreasing the burden of security operations of the tag.

— Keystream: generated through the AES encryption module initiated by an Initialization Vector (IV) and the master key. Keystream generation in the AES OFB mode is based on the iterative operation of AES encryption module (refer to Clause 12 for keystream generation).

— ChInt, ChTag or AuthData is a Challenge data used by the Authenticate command.

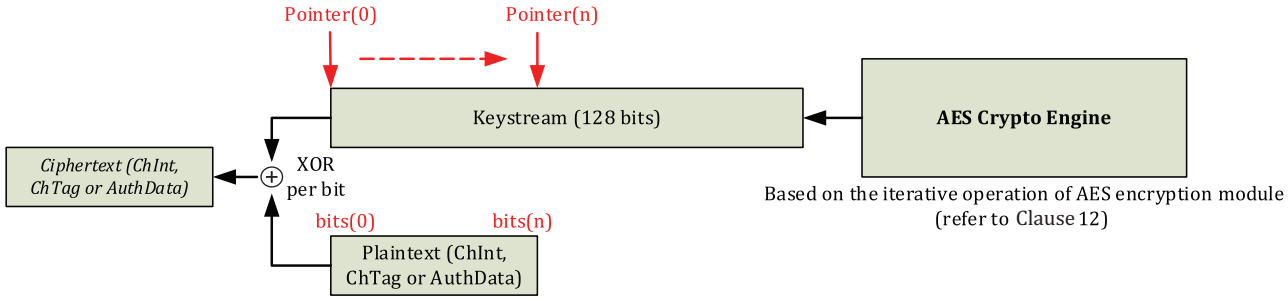

Figure 1 — Encryption of Authentication messages in AES OFB mode

6.3 Decryption in AES OFB mode

Figure 2 shows the AES OFB decryption process. The decryption process shall be implemented as the XOR operation between the ciphertext and the keystream. For a given tuple of n ciphertext messages C_1, ..., C_n and a keystream K, the decryption process is defined by $DEC_K(C_1, ..., C_n) = (C_1 \text{ XOR } K_1, ..., C_n \text{ XOR } K_n)$, where K_j is the *j-th* block of the keystream K and has the same bit-length as C_j.

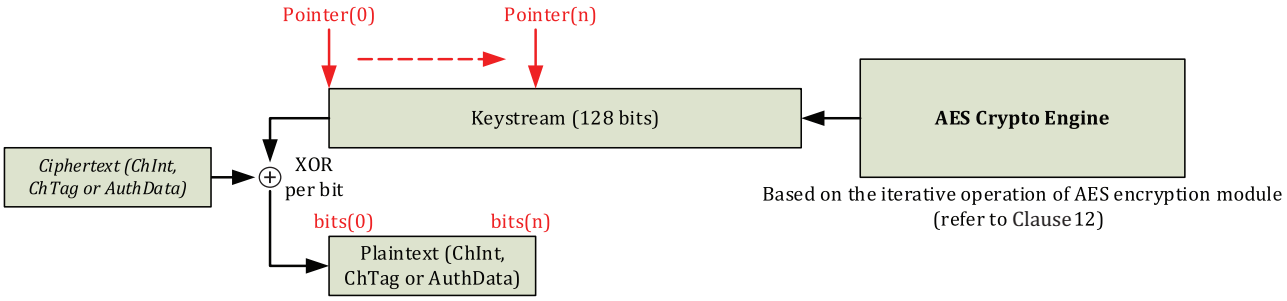

Figure 2 — Decryption of Authentication messages in AES OFB mode

7 Parameter definitions

The security parameters used in this part of ISO/IEC 29167 are described in Table 1.

Table 1 — Security parameters and descriptions

Parameter	Description
ChTag	Challenge of tag for challenge-response protocol (variable length, minimum length=16 bits, maximum length=128 bits)
ChInt	Challenge of interrogator for challenge-response protocol (variable length, minimum length=16 bits, maximum length=128 bits)
ChLen	Length of the challenge number in words. The minimum value of the ChLen is 1 and the maximum value of the ChLen is 15. The default value of the ChLen is 4. NOTE Length of a word is 16 bits.
AuthData	Challenge used in challenge-response protocol of Tag authentication via the server
RnTag[63:0]	64-bit random number (nonce) of tag (prevention of replay and other attacks) (The safe methods to avoid random number inference shall be used for implementation of this part of ISO/IEC 29167. That is, this part of ISO/IEC 29167 requires the Tag to generate a random number. The random number should contain sufficient entropy. However, this part of ISO/IEC 29167 does not specify a minimum. This part of ISO/IEC 29167 recommends that a random number complies with the random bit generator concepts and requirements of NIST Special Publication 800-90A.)
RnInt[63:0]	64-bit random number (nonce) of interrogator (prevention of replay and other attacks) (The safe methods to avoid random number inference shall be used for implementation of this part of ISO/IEC 29167. That is, this part of ISO/IEC 29167 requires the Interrogator to generate a random number. The random number should contain sufficient entropy. However this part of ISO/IEC 29167 does not specify a minimum. This part of ISO/IEC 29167 recommends that a random number complies with the random bit generator concepts and requirements of NIST Special Publication 800-90A.)
IV	Initialization Vector for keystream refreshment
Key[KeyID]	128-bit AES key with the ID number=KeyID
AES OFB ENC	AES OFB encryption
AES OFB DEC	AES OFB decryption
Command	Interrogator command
Response	Tag response

8 State diagram

After power-up or reset, the crypto suite transitions to its Ready state. Figure 3 shows the state diagram for the crypto engine. During the authentication procedure, a Tag does not reply to a command having an invalid handle or invalid CRC; the next Tag state is maintaining the current state.

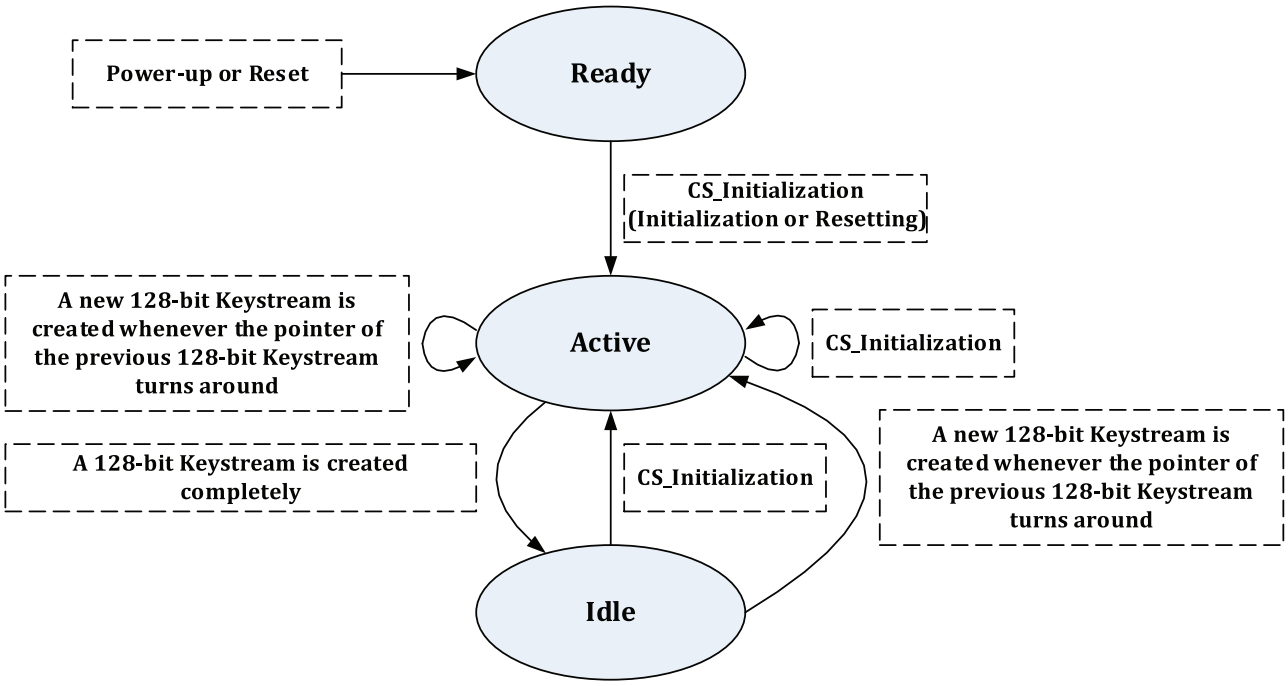

Figure 3 — State diagram for the crypto engine

9 Initialization and resetting

This cryptographic suite is initiated by the interrogator according to the authentication procedure. The crypto engine is reset by the CS_Initialization of Authenticate command.

10 Authentication

10.1 General

10.1.1 Authentication types

Authentication message format shall be implemented as shown in Table 2. This part of ISO/IEC 29167 describes the Message field of the Authenticate command and the Response field of the Tag reply.

Table 2 — Contents of Message field

	AuthMethod	Step	Flags	RnLen or ChLen	RnInt or ChInt
# of bits	3	2	3	4	16 ~ 128

This part of ISO/IEC 29167 defines the message format as follows.

— AuthMethod: defines the Authentication types (see Table 3).

— Step: orders the authenticate command.

— Flag: reserved for use.

— RnLen: defines the length of RnInt/RnTag in words.

— ChLen: defines the length of ChInt/ChTag in words.

Table 3 — Authentication types

AuthMethod	Value	Action	Remark
3 Bits (Authentication Methods)	000	Tag Authentication	
	001	Interrogator Authentication	
	010	Mutual Authentication	
	011	Proprietary use	Refer to Annex F
	100-110	RFU	Reserved for Future Use
	111	CS Initialization	Initialization Vector

10.1.2 CS_Initialization (Authentication type: AuthMethod "111", Mandatory)

10.1.2.1 General

The cryptographic suite shall be initiated by a fresh Initialization Vector (IV) and the master key. The first 128-bit IV results from the concatenation of the RnInt and the RnTag. The RnInt is immediately followed by the RnTag. RnLen is the length of RnInt/RnTag in words and its value is 4. In other words, the length of RnInt/RnTag is fixed to the 4 words (64 bits). If the value of RnLen field is not 4, the tag ignores the command.

10.1.2.2 Use of 'Authenticate' command

The source of Authentication command format is shown in Table 4.

Table 4 — Contents of Message field

AuthMethod	Step	Flags	RnLen	RnInt
111	**00**	**000**	**0100**	**64-bit random number**

10.1.2.3 Tag response

The Tag response format is shown in Table 5.

The response of CS_Initialization includes the Secure Parameter related to the key, KeyIndex and RnTag. Secure Parameter (Table 6) includes the Key information, Length of KeyIndex and the method using secure channel. Length of KeyIndex is the KeyIndex size in words. If this value is zero, KeyIndex does not exist. KeyIndex is the information related with the key pool of the tag and the interrogator.

Table 5 — Contents of Response field

Secure Parameter **16 bits**	KeyIndex **variable**	Message **64 bits**
Information related to the key and KeyIndex length	KeyIndex	Random number (RnTag, the same length of RnInt)

Table 6 — Secure Parameter format

KeyID[7:0]	RFU	Flag[2:0]	Length of KI(KeyIndex)
8	1	3	4
KeyID	0	[2:0]	[4:0]

— KeyID: it specifies the key.

— RFU: Reserved for future use.

— Flag: it shows the method supporting the secure channel by the tag (Table 7).

— Length of KI: it shows the length of KeyIndex in words.

Table 7 — Secure Channel Information

Name	Value	Description
Flag[2:0]	000	Do not support the Secure Channel
Flag[2:0]	001	Support Secure Channel
Flag[2:0]	010~111	Reserved for Future Use

10.2 Tag authentication (Authentication type: AuthMethod = “000”, Mandatory)

10.2.1 Tag authentication

Tag authentication is performed after an Inventory process.

Tag authentication is the process where an interrogator authenticates a tag. For this process, the interrogator generates randomly the challenge interrogator (ChInt), and encrypts the challenge interrogator (ChInt), and transmits the encrypted challenge interrogator (Enc(ChInt)) to the tag. The ChLen is the length of the challenge number in words and the minimum length is 16 bits.

When the tag receives the encrypted challenge interrogator (Enc(ChInt)) from the interrogator, the tag decrypts the encrypted challenge interrogator(Enc(ChInt)), and re-encrypts the decrypted challenge interrogator and transmits the re-encrypted challenge interrogator(Enc(ChInt)) to the interrogator. The encrypted challenge interrogator (3) from the interrogator is different from the re-encrypted challenge interrogator (4) transmitted by the tag because the pointer of keystream used for encrypting the ChInt by the interrogator is different from the pointer of keystream used for re-encrypting the ChInt by the tag.

Then, when the interrogator receives the re-encrypted challenge interrogator (Enc(ChInt)) from the tag, the interrogator decrypts the re-encrypted challenge interrogator (Enc(ChInt)), and compares the decrypted challenge interrogator with the generated challenge interrogator (ChInt), and performs the authentication for the tag.

When the decrypted challenge interrogator is determined to be identical to the generated challenge interrogator (ChInt), the interrogator determines that the tag authentication succeeded. Otherwise, the interrogator determines that the tag authentication failed. A tag remains at the current State regardless of the success or failure of the tag authentication.

For initializing an encryption engine, the interrogator randomly generates the random interrogator information (RnInt), and transmits the random interrogator (RnInt) to the tag. Also, the tag randomly generates the random tag (RnTag) and initializes an encryption engine of tag using the random interrogator (RnInt) and random tag (RnTag). The interrogator also uses the random interrogator (RnInt) and random tag (RnTag) that are transmitted by the tag to initiate its encryption engine.

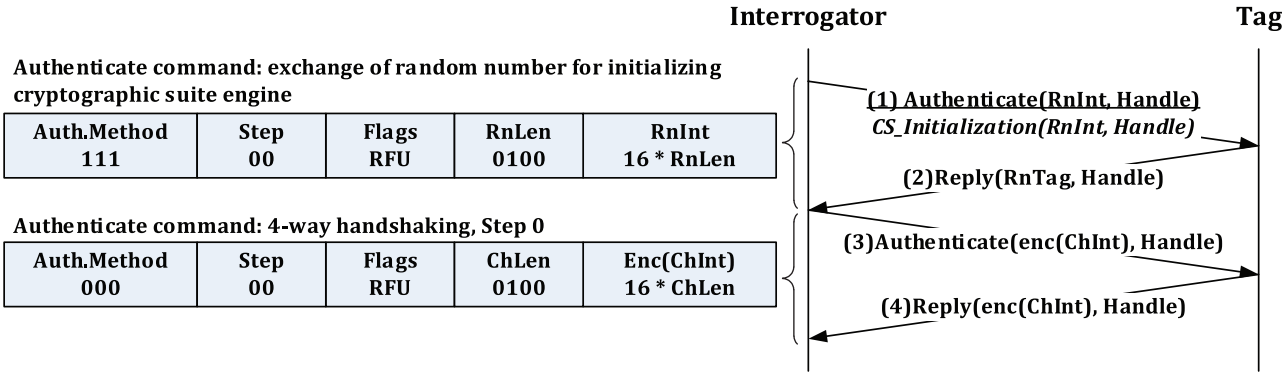

Figure 4 — Tag Authentication Diagram

10.2.2 Commands and responses for tag authentication

10.2.2.1 General

This Clause describes the process of a tag authentication (from (1) to (4) in Figure 4) between tag and interrogator. Figure 4 shows a tag authentication method. According to the 'AuthMethod' and 'Step', the 'Authenticate' commands are used for exchanging random numbers and sending a challenge number. The commands and responses for the functions are as follows.

10.2.2.2 Authenticate Command (CS_Initialization)

In Figure 4, the first 'Authenticate' command (Table 8) is to exchange random numbers between tag and interrogator for initializing the IV. If the interrogator performed already this function at the current inventory, this step can be omitted. And its format is as follows. RnLen is the length of RnInt/RnTag in words and its value is 4, so the length of RnInt/RnTag is fixed to the 4 words (64 bits). If the value of RnLen field is not 4, the tag ignores the command. CS_Initialization refers to 10.1.2.

Table 8 — Authenticate (CS_Initialization)

AuthMethod	Step	Flags	RnLen	RnInt
111	00	000	0100	64-bit random

10.2.2.3 Response (CS_Initialization)

The following is the response (Table 9) to the first Authenticate command. CS_Initialization refers to 10.1.2.

Table 9 — Response (CS_Initialization)

Secure Parameter	KeyIndex	RnTag
		64-bit random

10.2.2.4 Authenticate Command (Tag Authentication)

In Figure 4, the second 'Authenticate' command (Table 10) is used for sending a challenge number. And its format is as follows. The ChLen is the length of the challenge number in words and the minimum length is 16 bits. The ChInt is generated randomly by the interrogator. The interrogator encrypts the ChInt and transmits the encrypted ChInt (Enc(ChInt)) to the tag.

Table 10 — Authenticate (Tag Authentication)

AuthMethod	Step	Flags	ChLen	Enc(ChInt)
000	00	000	4-bit length	Ciphertext

10.2.2.5 Response (Tag Authentication)

The following is the response (Table 11) to the second Authenticate command. The length of 'ChInt' shall be the same as that of the received 'ChInt'. The tag decrypts and re-encrypts the received ChInt. And then, the tag transmits the encrypted ChInt (Enc(ChInt)) to interrogator through the response.

Table 11 — Response (Tag Authentication)

Enc(ChInt)
Ciphertext

When the interrogator receives the re-encrypted challenge interrogator from the tag, the interrogator decrypts the re-encrypted challenge interrogator, and compares the decrypted challenge interrogator with the generated challenge interrogator.

When the decrypted challenge interrogator is determined to be identical to the generated challenge interrogator, the interrogator determines that the tag authentication succeeded. Otherwise, the interrogator determines that the tag authentication failed.

10.3 Interrogator authentication (Authentication type: AuthMethod = "001", Optional)

10.3.1 Interrogator authentication

Interrogator authentication is performed after an Inventory process.

Interrogator authentication is the process where a tag authenticates an interrogator. For this process, the interrogator transmits a request for challenge tag to a tag. Then the tag generates randomly the challenge tag (ChTag), and encrypts the challenge tag (ChTag) and transmits the encrypted challenge tag (Enc(ChTag)) to the interrogator. The ChLen is the length of the requested challenge number in words and the minimum length is one word (i.e. 16 bits).

When the interrogator receives the encrypted challenge tag (Enc(ChTag)) from the tag, the interrogator decrypts the encrypted challenge tag (Enc(ChTag)), and re-encrypts the decrypted challenge tag and transmits the re-encrypted challenge tag (Enc(ChTag)) to the tag. The encrypted challenge tag (4) from the tag is different from the re-encrypted challenge tag (5) transmitted by the interrogator because the pointer of keystream used for encrypting the ChTag by the tag is different from the pointer of keystream used for re-encrypting the ChTag by the interrogator.

When the tag receives the re-encrypted challenge tag (Enc(ChTag)) from the interrogator, the tag decrypts the re-encrypted challenge tag (Enc(ChTag)), and compares the decrypted challenge tag with the generated challenge tag (ChTag), and performs the authentication for the interrogator.

When the decrypted challenge tag is determined to be identical to the generated tag (ChTag), the tag determines that the interrogator authentication succeeded. Otherwise, the tag determines that the interrogator authentication failed. If the interrogator authentication fails, the tag does not transmit the tag response.

When the authentication for the interrogator succeeds, the interrogator can delete, change, or add a Master Key (MK) of the tag. Also, the interrogator can change the current operating mode of the tag.

In the process of interrogator authentication, the interrogator can initialize cryptographic suite (encryption engine) of interrogator. For the initialization of cryptographic suite, the interrogator randomly generates the random interrogator (RnInt) and transmits the random interrogator (RnInt) to the tag.

When the tag receives the random interrogator (RnInt) from the interrogator, the tag randomly generates the random tag (RnTag) and initializes cryptographic suite (encryption engine and IV) of tag using the random interrogator (RnInt) and random tag (RnTag). And then the tag transmits the random tag (RnTag) to the interrogator. When the interrogator receives the random tag (RnTag), the interrogator initializes cryptographic suite (encryption engine and IV) of interrogator using the random interrogator (RnInt) and random tag (RnTag).

Also, when the authentication for the interrogator succeeds, the tag shall transition to the secured State.

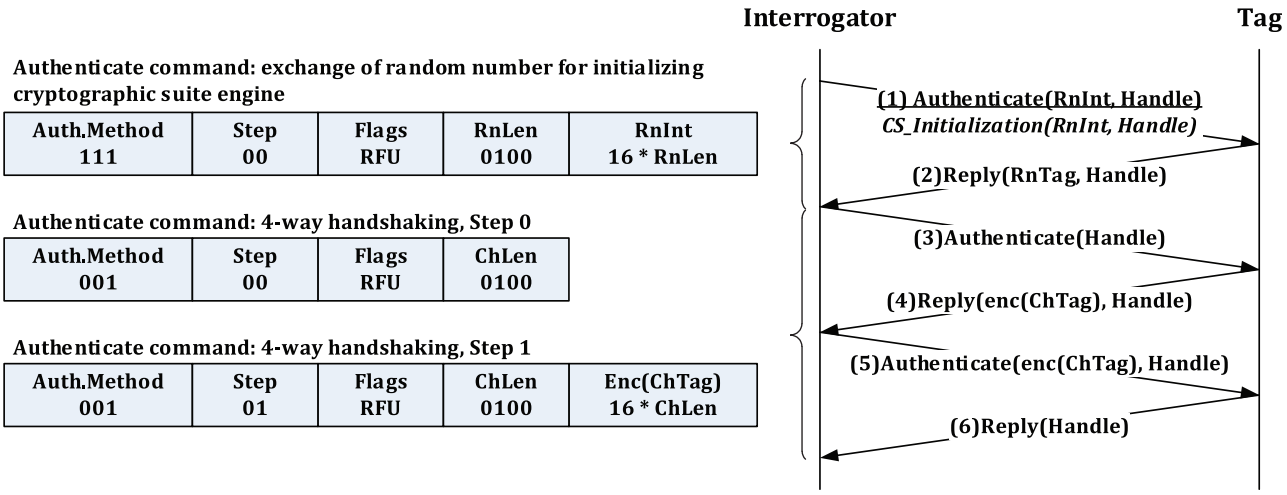

Figure 5 — Interrogator Authentication Diagram

10.3.2 Commands and responses for interrogator authentication

10.3.2.1 General

This Clause describes the process of an interrogator authentication (from (1) to (6) in Figure 5) between tag and interrogator. Figure 5 shows an interrogator authentication method. According to the 'AuthMethod' and 'Step', the 'Authenticate' commands are used for exchanging random numbers and sending a challenge number. The commands and responses for the functions are as follows.

10.3.2.2 Authenticate Command (CS_Initialization)

In Figure 5, the first 'Authenticate' command (Table 12) is to exchange random numbers between tag and interrogator for initializing the IV. If the interrogator performed already this function at the current inventory, this step can be omitted. And its format is as follows. RnLen is the length of RnInt/RnTag in words and its value is 4. In other words, the length of RnInt/RnTag is fixed to the 4 words (64 bits). If the value of RnLen field is not 4, the tag ignores the command. CS_Initialization refers to 10.1.2.

Table 12 — Authenticate (CS_Initialization)

AuthMethod	Step	Flags	RnLen	RnInt
111	00	000	0100	64-bit random

10.3.2.3 Response (CS_Initialization)

The following is the response (Table 13) to the first Authenticate command. CS_Initialization refers to 10.1.2.

Table 13 — Response (CS_Initialization)

Secure Parameter	KeyIndex	RnTag 64-bit random

10.3.2.4 Authenticate Command (Interrogator Authentication 1)

In Figure 5, the second 'Authenticate' command (Table 14) is used for requesting a challenge number. And its format is as follows. The ChLen is the length of the requested challenge number in words and the minimum length is 16 bits.

Table 14 — Authenticate (Interrogator Authentication 1)

AuthMethod	Step	Flags	ChLen
001	00	000	4-bit length

10.3.2.5 Response (Interrogator Authentication 1)

The following is the response (Table 15) to the second command. The length of the challenge number can be derived by the 'Length' field. The tag generates randomly the challenge tag (ChTag), and encrypts the challenge tag (ChTag) and transmits the encrypted challenge tag (Enc(ChTag)) to the interrogator.

Table 15 — Response (Interrogator Authentication 1)

Enc(ChTag)
Ciphertext

10.3.2.6 Authenticate Command (Interrogator Authentication 2)

In Figure 5, the third 'Authenticate' command (Table 16) is used for sending a ciphertext of the tag challenge number. And its format is as follows. The length of 'Enc(ChTag)' shall be the same as that of the received 'Enc(ChTag)'. The ChLen is the length of the challenge number in words and the minimum length is 16 bits.

Table 16 — Authenticate (Interrogator Authentication 2)

AuthMethod	Step	Flags	ChLen	Enc(ChTag)
001	01	000	4-bit length	Ciphertext

When the tag receives the re-encrypted challenge tag from the interrogator, the tag decrypts the re-encrypted challenge tag, and compares the decrypted challenge tag with the generated challenge tag.

When the decrypted challenge tag is determined to be identical to the generated tag, the tag determines that the interrogator authentication succeeded. Otherwise, the tag determines that the interrogator authentication failed.

10.3.2.7 Response (Interrogator Authentication 2)

The following is the response (Table 17) to the third command. It has no payload field. If the interrogator authentication succeeds, the tag transmits the tag response as follows. Otherwise, the tag does not transmit the tag response.

Table 17 — Response (Interrogator Authentication 2)

Response
nothing

10.4 Mutual authentication (Authentication type: AuthMethod = “010”, Mandatory)

10.4.1 Mutual authentication

Mutual authentication is performed after an Inventory process.

Mutual authentication is the process where a tag and an interrogator authenticate each other. This authentication consists of the processes for tag authentication and interrogator authentication as the tag authentication is followed by the interrogator authentication.

First, the interrogator randomly generates the challenge interrogator (ChInt), and encrypts the challenge interrogator, and then, transmits a request for challenge tag and the encrypted challenge interrogator (Enc(ChInt)) to the tag. The ChLen is the length of the challenge number in words and the minimum length is 16 bits.

When the tag receives the above request from the interrogator, the tag decrypts the encrypted challenge interrogator (Enc(ChInt)) and re-encrypts the decrypted challenge interrogator. Also, the tag randomly generates the challenge tag (ChTag), and encrypts the challenge tag (ChTag). Then the tag transmits the re-encrypted challenge interrogator (Enc(ChInt)) and the encrypted challenge tag (Enc(ChTag)) to the interrogator. The encrypted challenge interrogator (3) from the interrogator is different from the re-encrypted challenge interrogator (4) transmitted by the tag because the pointer of keystream used for encrypting the ChInt by the interrogator is different from the pointer of keystream used for re-encrypting the ChInt by the tag.

When the interrogator receives the re-encrypted challenge interrogator (Enc(ChInt)) from the tag, the interrogator decrypts the re-encrypted challenge interrogator (Enc(ChInt)), and compares the decrypted challenge interrogator with the generated challenge interrogator (ChInt), and performs the authentication for the tag.

When the decrypted challenge interrogator is determined to be identical to the generated challenge interrogator (ChInt), the interrogator determines that the tag authentication succeeded. Otherwise, the interrogator determines that the tag authentication failed.

When the authentication for the tag succeeds, the interrogator decrypts the encrypted challenge tag (Enc(ChTag)), and re-encrypts the decrypted challenge tag, and transmits the re-encrypted challenge tag (Enc(ChTag)) to the tag. The encrypted challenge tag (4) from the tag is different from the re-encrypted challenge tag (5) transmitted by the interrogator because the pointer of keystream used for encrypting the ChTag by the tag is different from the pointer of keystream used for re-encrypting the ChTag by the interrogator.

When the tag receives the re-encrypted challenge tag (Enc(ChTag)) from the interrogator, the tag decrypts the re-encrypted challenge tag (Enc(ChTag)) and compares the decrypted challenge tag with the generated challenge tag (ChTag), and performs the authentication for the interrogator.

When the decrypted challenge tag is determined to be identical to the generated tag (ChTag), the tag determines that the interrogator authentication succeeded. Otherwise, the tag determines that the interrogator authentication failed. If the mutual authentication fails, the tag does not transmit the tag response.

When the authentication for the interrogator succeeds, the interrogator can delete, change, or add a Master Key (MK) of the tag. Also the interrogator can change the current operating mode of the tag.

In the process of interrogator authentication, the interrogator can initialize cryptographic suite (encryption engine) of interrogator. For the initialization of cryptographic suite (encryption engine), the interrogator randomly generates the random interrogator (RnInt), and transmits the random interrogator (RnInt) to the tag.

When the tag receives the random interrogator (RnInt) from the interrogator, the tag randomly generates the random tag (RnTag) and initializes cryptographic suite (encryption engine and IV) of tag using the random interrogator (RnInt) and random tag (RnTag). And then the tag transmits the random

tag to the interrogator. When the interrogator receives random tag (RnTag), the interrogator initializes cryptographic suite (encryption engine and IV) of interrogator using the random interrogator (RnInt) and random tag (RnTag).

When the authentication for the interrogator succeeds, the tag shall transition to the secured state.

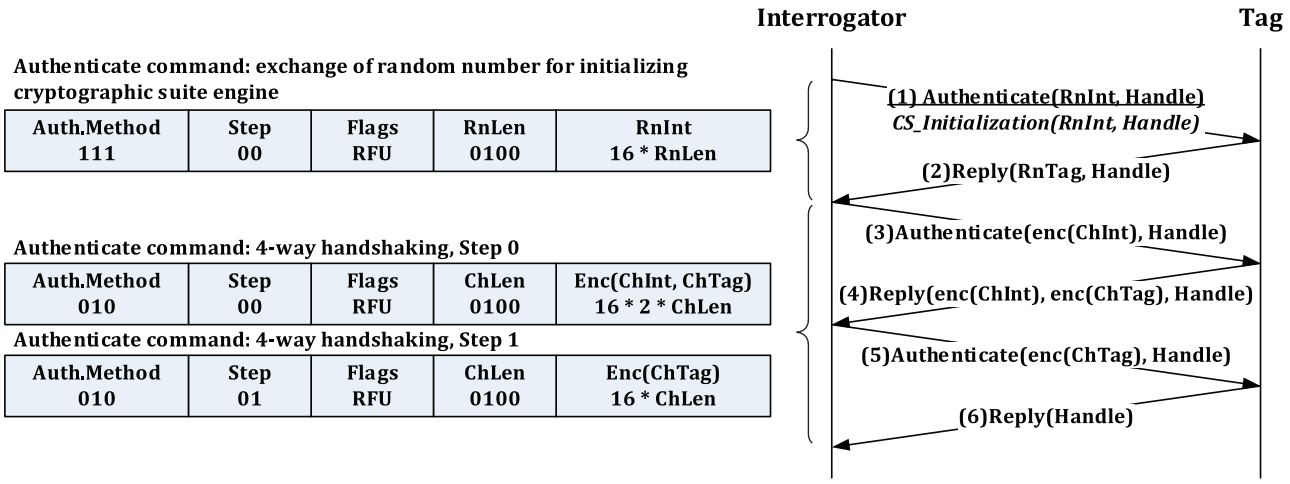

Figure 6 — Mutual Authentication Diagram

10.4.2 Commands and responses for mutual authentication

10.4.2.1 General

This Clause describes the process of a mutual authentication (from (1) to (6) in Figure 6) between tag and interrogator. Figure 6 shows a mutual authentication method. According to their 'AuthMethod' and 'Step', the commands are used for random number exchange and 4-way handshaking. The commands and responses for the functions are as follows.

10.4.2.2 Authenticate Command (CS_initialization)

In Figure 6, the first 'Authenticate' command (Table 18) is to exchange random numbers between tag and interrogator for initializing the IV. If the interrogator performed already this function at the current inventory, this step can be omitted. And its format is as follows. RnLen is the length of RnInt/RnTag in words and its value is 4. In other words, the length of RnInt/RnTag is fixed to the 4 words (64 bits). If the value of RnLen field is not 4, the tag ignores the command. CS_Initialization refers to 10.1.2.

Table 18 — Authenticate (CS_Initialization)

AuthMethod	Step	Flags	RnLen	RnInt
111	00	000	0100	64-bit random

10.4.2.3 Response (CS_initialization)

The following is the response (Table 19) to the first Authenticate command. CS_Initialization refers to 10.1.2.

Table 19 — Response (CS_Initialization)

Secure Parameter	KeyIndex	RnTag
		64-bit random

10.4.2.4 Authenticate Command (Mutual Authentication 1)

In Figure 6, the second 'Authenticate' command (Table 20) is used for sending an interrogator challenge number and receiving a tag challenge number. And its format is as follows. The ChLen is the length of the challenge number in words. The interrogator randomly generates the challenge interrogator (ChInt), and encrypts the challenge interrogator, and then, transmits a request for challenge tag and the encrypted challenge interrogator (Enc(ChInt)) to the tag.

Table 20 — Authenticate (Mutual Authentication 1)

AuthMethod	Step	Flags	ChLen	Enc(ChInt)
010	00	000	4-bit length	Ciphertext

10.4.2.5 Response (Mutual Authentication 1)

The following is the response (Table 21) to the second command. The tag decrypts the encrypted challenge interrogator (Enc(ChInt)) and re-encrypts the decrypted challenge interrogator. Also, the tag randomly generates the challenge tag (ChTag), and encrypts the challenge tag (ChTag). Then the tag transmits the re-encrypted challenge interrogator (Enc(ChInt)) and the encrypted challenge tag (Enc(ChTag)) to the interrogator. The length of 'Enc(ChInt)' and 'Enc(ChTag)' shall be the same as that of the received 'Enc(ChInt)', respectively.

Table 21 — Response (Mutual Authentication 1)

Enc(ChInt)	Enc(ChTag)
Ciphertext	Ciphertext

The interrogator receives the re-encrypted challenge interrogator from the tag, the interrogator decrypts the re-encrypted challenge interrogator, and compares the decrypted challenge interrogator with the generated challenge interrogator.

When the decrypted challenge interrogator is determined to be identical to the generated challenge interrogator, the interrogator determines that the tag authentication succeeded. Otherwise, the interrogator determines that the tag authentication failed.

10.4.2.6 Authenticate Command (Mutual Authentication 2)

In Figure 6, the third 'Authenticate' command (Table 22) is used for sending a ciphertext of the tag challenge number. And its format is as follows. The interrogator decrypts the encrypted challenge tag (Enc(ChTag)), and re-encrypts the decrypted challenge tag, and transmits the re-encrypted challenge tag (Enc(ChTag)) to the tag. The length of 'Enc(ChTag)' shall be the same as that of the received 'Enc(ChTag)'. The ChLen is the length of the challenge number in words.

Table 22 — Authenticate (Mutual Authentication 2)

AuthMethod	Step	Flags	ChLen	Enc(ChTag)
010	01	000	4-bit length	Ciphertext

When the tag receives the re-encrypted challenge tag from the interrogator, the tag decrypts the re-encrypted challenge tag, and compares the decrypted challenge tag with the generated challenge tag.

When the decrypted challenge tag is determined to be identical to the generated tag, the tag determines that the interrogator authentication succeeded. Otherwise, the tag determines that the interrogator authentication failed.

10.4.2.7 Response (Mutual Authentication 2)

The following is the response (Table 23) to the third command. It has no payload field. If the mutual authentication succeeds, the tag transmits the tag response as follows. Otherwise, the tag does not transmit the tag response.

Table 23 — Response (Mutual Authentication 2)

Response
nothing

11 Communication

This Cryptographic suite does not support secure communication.

12 Key management and key update

12.1 Master key selection

Each keystream shall be generated from the master key. A tag conforming to this part of ISO/IEC 29167 shall have a master key. The master key can be updated by an authorized interrogator.

The master key of a tag is stored in the key field of tag memory. The key index related with the master key is stored in the key index field. It is recommended that the first master key is set by a tag manufacturer. The master key blocks shall be protected from unauthorized Read/Write/KeyUpdate commands.

Figure 7 shows a master key selection method.

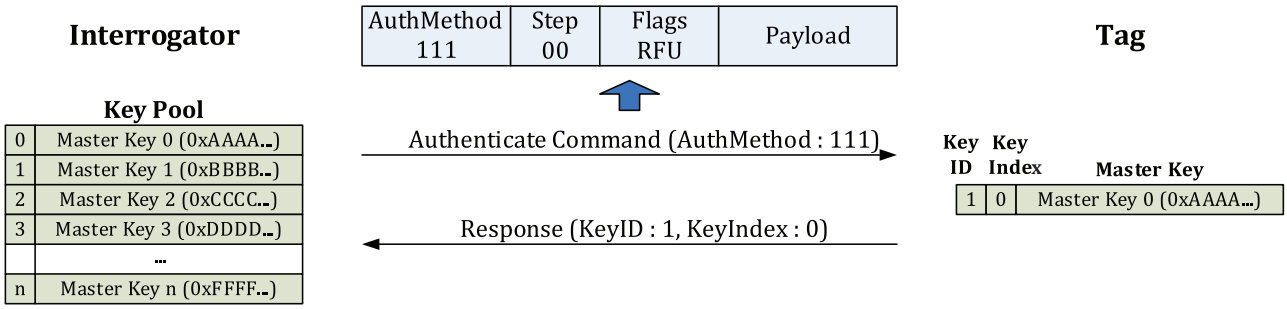

Figure 7 — Master Key Selection

This Cryptographic suite supports the multiple master keys within a tag. If a tag has the multiple master keys within a tag (for examples, KeyID = 1, KeyID = 3, KeyID = 5), an interrogator chooses the KeyID as shown in Figure 8. CS_Initialization refers to 10.1.2.

— An interrogator transmits the CS_Initialization (Authenticate command, AuthMethod: 111) to a tag for choosing the KeyID and generating the keystream. A tag transmits the first Key (KeyID = 1) response of the CS_Initialization.

— If an interrogator does not have the authority on the first Key (KeyID = 1), an interrogator transmits again the CS_Initialization to a tag. A tag transmits the second Key (KeyID = 3) response of the second CS_Initialization. For example, a tag transmits sequentially the KeyID about the CS_Initialization.

— If an interrogator does not have the authority on the second Key (KeyID = 3), an interrogator transmits again the CS_Initialization to a tag. A tag transmits the third Key (KeyID = 5) response of the third CS_Initialization.

— If an interrogator has the authority on the third Key (KeyID = 5), an interrogator generates a keystream using the response.

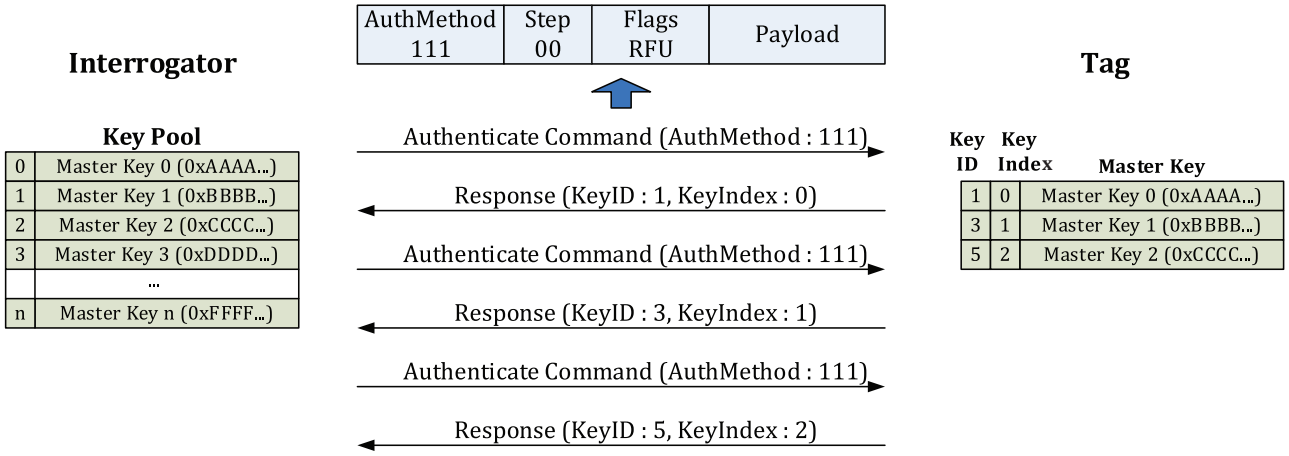

Figure 8 — KeyID Selection in multiple keys within a Tag

12.2 Keystream generation

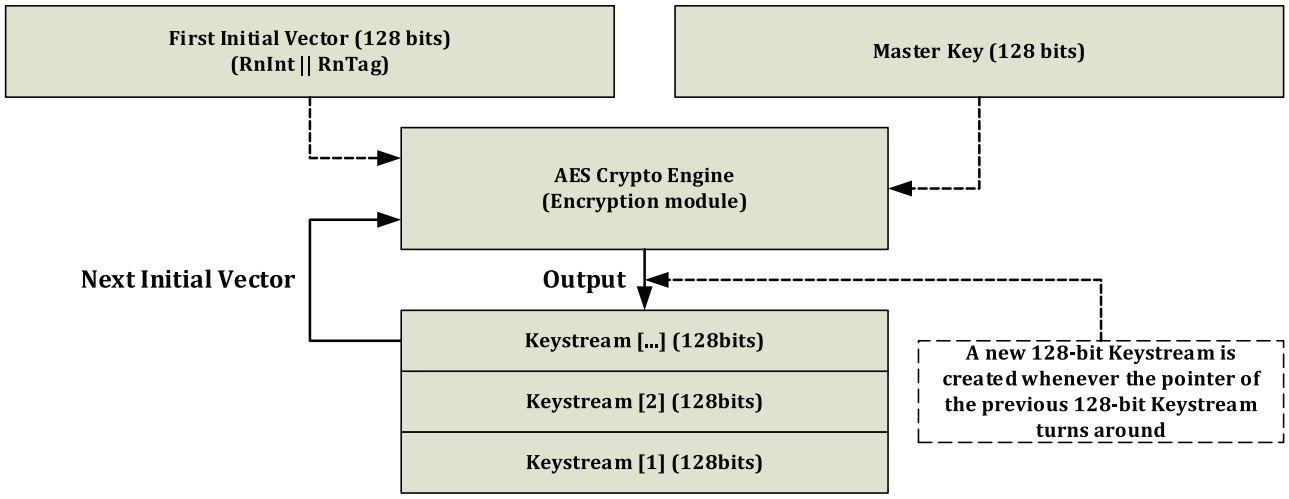

Figure 9 — Keystream Generation

Figure 9 shows the generating method of a keystream based on the AES encryption module. The master key is a secret key and plays an important role in generating a keystream.

The AES encryption module that generates a keystream is initiated by an Initialization Vector (IV) and the master key. The first IV is concatenated by the interrogator and the tag during 'CS_Initialization' operation. The RnInt is a random number transmitted to a tag from an interrogator and the RnTag is a random number transmitted to an interrogator from a tag. The length of the RnInt and the RnTag is 64 bits respectively. The 128-bit IV consists of the concatenation of the RnInt and the RnTag. For the first time, the AES crypto engine generates the first keystream by encrypting the 128-bit IV using the master key. And then, it generates the second keystream by encrypting the first keystream using the master key. The AES crypto engine replaces the target data by the previous keystream in every generating routine of a new keystream. Thus, it means that the AES crypto engine generates different keystreams in each generating routine. The AES crypto engine generates firstly two keystreams in advance to prevent the exhaustion of the keystream. And then, the AES crypto engine generates a new keystream whenever one keystream of two keystreams is exhausted. During the above process, the Interrogator and the tag shall generate a random number. The random number should contain sufficient entropy; however, this part of ISO/IEC 29167 does not specify a minimum.

12.3 Key update

12.3.1 General

Interrogators and Tags may implement the KeyUpdate command for the CSI of AES OFB mode; if they do, they shall implement it as shown in this Clause. KeyUpdate allows an Interrogator to delete, write or overwrite a key stored in a Tag. Table 24 shows the KeyUpdate command. The key update operation is only allowed in protected environments.

KeyUpdate has the following fields:

— SenRep specifies whether a Tag backscatters a reply message or stores the message in a ResponseBuffer.

— Length is the message length, in bits.

— KeyID specifies the key to be updated.

— Message contains WordPtr and 'KeyIndex' or 'key'.

12.3.2 Command

Table 24 — Command format

WordPtr	UpData
8	16*n
word pointer	Length of KeyIndex, KeyIndex or Key

UpData includes the Length of KeyIndex, KeyIndex and the Key.

— Length of KeyIndex is the KeyIndex size in words. If this value is zero, KeyIndex does not exist. Length of KeyIndex uses only the lower 4 bits of 16 bits. Thus, it supports the length of 1 ~ 15 words.

— KeyIndex is the information related with the key pool of the tag and the interrogator.

The WordPtr (Word Pointer) indicates the Length of KeyIndex, KeyIndex or the Key and the order of KeyUpdate command. The WordPtr field has two types as shown in Table 25.

Table 25 — WordPtr fields

bit	0	1	2	3	4	5	6	7
WordPtr	0 (processing)	0 (Length of Key-Index, KeyIndex)/1 (Key)	Word pointer for Length of KeyIndex, KeyIndex and Key					
	1 (complete)	0 (update)/1 (delete)	RFU		Mask		Action	
			-		skip/apply	skip/apply	Key write	permalock

The bit 0 of the WordPtr has a value of 0 or 1. The value of 0 means that the data still remains to be updated and the value of 1 means the final message for update.

In the case of the value of 0, bit 1 of WordPtr indicates the Length of KeyIndex, KeyIndex or Key. The value of 0 in bit 1 means the UpData includes the Length of KeyIndex and KeyIndex. The value of 1 in bit 1 means the UpData includes Key. The remained bits of the WordPtr indicate the starting address to be updated, which is Length of KeyIndex, KeyIndex or the Key blocks. The user chooses whether UpData is updated by a word or by words according to the value of Length field (Length = WordPtr (8 bits) + UpData (16 bits) × n) of KeyUpdate command.

If bit 0 of the WordPtr is 1, bit 1 of WordPtr indicates the update or deletion of Key, bit 4 and bit 5 of WordPtr are Mask bits, and bit 6 and bit 7 are Action bits. At this time, the UpData shall be CRC-16 value

for the transmitted Length of KeyIndex, KeyIndex and the Key blocks. Thus, the transmitted data is confirmed by the CRC-16 value.

The functionality of the various Action fields in the final KeyUpdate command is described in Table 26.

Table 26 — Action fields functionality

Action		Meaning
Key write	per-malock	
0	0	The Key blocks are writable or readable by the KeyUpdate command from the secured state.
0	1	The Key blocks are permanently writable by the KeyUpdate command from the secured state and not readable from the any state by any command.
1	0	The Key blocks are not writable by the KeyUpdate command from the secured state and not readable from the any state by any command.
1	1	The Key blocks are permanently not writable or readable from the any state by any command.

A tag shall interpret these bit values as follows:

— Mask (0): Ignore the associated Action field and retain the current lock setting.

— Mask (1): Implement the associated Action field and overwrite the current lock setting.

— Action (Key write, 0): De-assert write-protection through the KeyUpdate command.

— Action (Key write, 1): Assert write-protection through the KeyUpdate command.

The order of KeyUpdate command is as follows.

WordPtr	Updata
(0×00)	KeyIndex length (2 word)
(0×01)	KeyIndex(0)
(0×02)	KeyIndex(1)
(0×40)	Key (0)
(0×41)	Key (1)
(0×42)	Key (2)
(0×43)	Key (3)
(0×44)	Key (4)
(0×45)	Key (5)
(0×46)	Key (6)
(0×47)	Key (7)
(0×8X)	CRC CheckSum

Figure 10 — UpData of KeyUpdate command (an example of a word)

WordPtr	Updata	
(0x00)	length (2 word)	KeyIndex (0 ~ 1)
(0x40)	Key (0 ~3)	
(0x44)	Key (4 ~ 7)	
(0x8X)	CRC CheckSum	

a) Example of the variable words

WordPtr	Updata	
(0x00)	length (2 word)	KeyIndex(0 ~ 1)
(0x40)	Key (0 ~ 7)	
(0x8X)	CRC CheckSum	

b) Another example of the variable words

Figure 11 — UpData of KeyUpdate command

The UpData of KeyUpdate command operates in a word (Figure 10) or the variable words [Figure 11 a) and b)]. The interrogator indicates the length of WordPtr and UpData in the Length field of KeyUpdate command.

If the Interrogator performs the deletion of Key, it transmits only the WordPtr in Message field of KeyUpdate command and is not existed the UpData field. At this time, the value of WordPtr is 0xC0 [Figure 12a)] means that the complete bit and the delete bit are set to 1).

Also, if the Interrogator performs the modification of Key Action, it transmits the marked WordPtr (Action field) of the final KeyUpdate command or it transmits only the WordPtr (Action field) in Message field of KeyUpdate command. If the Interrogator transmits only the WordPtr of KeyUpdate command, the UpData shall be NULL. At this time, the value of WordPtr is 0x8X [Figure 12b) means that the complete bit and the related bit are set to 1].

In this case, the example of the KeyUpdate command is as follows.

WordPtr
(0xC0)

Figure 12a — The deletion of Key

WordPtr
(0x8X)

Figure 12b — The modification of Key Action

Annex A
(normative)

Crypto suite state transition tables

The crypto suite state transitions are described in Table A.1.

Table A.1 — Crypto suite state transition table

Current State	Command	Action	Next State
Ready	CS_Initialization	Initializes the encryption engine	**Active**
Active or Idle	CS_Initialization	Initializes the encryption engine	**Active**
Active	Tag Authentication	Verify Tag key	**Active**
Active	Interrogator Authentication	Verify Interrogator key	**Active**
Active	Mutual Authentication	Verify Tag key and Interrogator key	**Active**
Idle	Tag Authentication	Verify Tag key (a pointer of keystream remains)	**Idle**
		Verify Tag key (a pointer of keystream have exhausted)	**Active**
Idle	Interrogator Authentication	Verify Interrogator key (a pointer of keystream remains)	**Idle**
		Verify Interrogator key (a pointer of keystream have exhausted)	**Active**
Idle	Mutual Authentication	Verify Tag key and Interrogator key (a pointer of keystream remains)	**Idle**
		Verify Tag key and Interrogator key (a pointer of keystream have exhausted)	**Active**

Any combination of Current States and Transitions not listed in Table A.1 shall result in an error and consequently a transition to the Ready state.

All other errors resulting from the execution of commands shall result in an error and consequently a transition to the Ready state.

Annex B
(normative)

Error Codes

The error codes are described in Table B.1.

Table B.1 — Error Codes

Error-Code Support	Error Code	Error-Code Name	Error Description
Error-specific	00000000_2	Other error	Catch-all for errors not covered by other codes.
	00000011_2	Memory overrun	The specified memory location does not exist or the UII length field is not supported by the tag.
	00000100_2	Memory locked	The specified memory location is locked and/or permalocked and is either not writeable or not readable.
	00001011_2	Insufficient power	The tag has insufficient power to perform the memory-write operation.
	00010001_2	Sensor scheduling configuration	Number of specific time scheduling records exceeds maximum possible number.
	00010010_2	Tag busy	An interrogator tries to read or write sensor specific data. The tag is currently busy accessing the sensor or storing sensor related data (e.g. a new measurement value) and therefore is temporarily unable to serve the request.
	00010011_2	Measurement type not supported	The sensor does not support the specified measurement type.
Non-specific	00001111_2	Non-specific error	The tag does not support error-specific codes.
Security Error-specific	00100001_2	No Key	The Key does not exist.
	00100010_2	CS not initialized or Crypto Suite Error	The crypto engine is not initialized because the Authenticate (CS_initialization) command has not been executed.
	00100011_2	Insufficient Privileges	The specified key is the Insufficient Privileges.
	00100100_2	CRC Checksum error	The error of CRC checksum is occurred in the KeyUpdate command.

Annex C
(normative)

Cipher description

The Advanced Encryption Standard (AES) block cipher is described in detail in Reference.[6]

Annex D
(informative)

AES OFB test vectors

The first five test cases test AES-128 OFB encryption engine. And, the additional two test cases test encrypting and decrypting the four continuous blocks using AES-128 OFB encryption engine.

Each test case includes the key, Initialization Vector (RnInt and RnTag), Keystream, the plaintext (ChInt, ChTag, or AuthData, etc), and the resulting ciphertext. The values of keys and data are either hexadecimal numbers (prefixed by "0x"). The computed cyphertext values are all hexadecimal numbers.

Case #1: Encrypting 16 bytes (1 block) using AES OFB-encrypt with 128-bit key

Key: 0xaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa

RnInt: 0x1234123412341234, RnTag : 0x1234123412341234

IV: 0x12341234123412341234123412341234

Keystream: 0x85b8176673693b793700475876e6fb08

Plaintext: 0x123456789abcde123456789abcde1234

Ciphertext: 0x978c411ee9d5e56b03563fc2ca38e93c

Case #2: Encrypting 16 bytes (1 block) using AES OFB-encrypt with 128-bit key

Key: 0xc651ecbafb7cf8e75a339a0d5825175e

RnInt: 0x67734acdf24654e8, RnTag : 0x7663c932315a05e9

IV: 0x67734acdf24654e87663c932315a05e9

Keystream: 0x13519bd8f5cdda9473c763284c995f29

Plaintext: 0x69ff29abe3c21b8d2e9f66b7a35d58d4

Ciphertext: 0x7aaeb273160fc1195d58059fefc407fd

Case #3: Encrypting 16 bytes (1 block) using AES OFB-encrypt with 128-bit key

Key: 0xb29b11743d70a1fc01ea965cb03254db

RnInt: 0xabc6548221873ee1, RnTag : 0x3e976b38ecfb3c18

IV: 0xabc6548221873ee13e976b38ecfb3c18

Keystream: 0x095ae589eb942f9ddbed0e1a2013ac7d

Plaintext: 0xcdb40e41dce941677edc8f2a3bafec5c

Ciphertext: 0xc4eeebc8377d6efaa53181301bbc4021

Case #4: Encrypting 16 bytes (1 block) using AES OFB-encrypt with 128-bit key

Key: 0x1afb3ad13c75615c99b1b3f7a1cad064

RnInt: 0x0243aa290594bef9, RnTag : 0xa895f1efe9e5cb47

IV: 0x0243aa290594bef9a895f1efe9e5cb47

Keystream: 0xa664f6e6e72b05821b2c9617820ffd09

Plaintext: 0xfefafbe67cd889bb0feb05003a0b48bd

Ciphertext: 0x589e0d009bf38c3914c79317b804b5b4

Case #5: Encrypting 16 bytes (1 block) using AES OFB-encrypt with 128-bit key

Key: 0x231cc55a4b3b2409d41b3be347bb197d

RnInt: 0x1fa864735e63649e, RnTag : 0xaaf2afcd485c229b

IV: 0x1fa864735e63649eaaf2afcd485c229b

Keystream: 0xeeddb2d94c7f97fa268b5df3c17a97d7

Plaintext: 0x1e7b14c5797011dcac103350156fbaf5

Ciphertext: 0xf0a6a61c350f86268a9b6ea3d4152d22

Case #6: Encrypting the 4 continuous block using AES OFB-encrypt with 128-bit key

Key: 0x7cc254f81be8e78d765a2e63339fc99a

IV: 0x67c6697351ff4aec29cdbaabf2fbe346

Block #1

Keystream: 0x80ed55152009abb9971985f536a5cc40

Plaintext: 0x66320db73158a35a255d051758e95ed4

Ciphertext: 0xe6df58a2115108e3b24480e26e4c9294

Block #2

IV: 0x80ed55152009abb9971985f536a5cc40

Keystream: 0xfa8e67e38d34fa4f7965e9736b58f0cd

Plaintext: 0xabb2cdc69bb454110e827441213ddc87

Ciphertext: 0x513caa251680ae5e77e79d324a652c4a

Block #3

IV: 0xfa8e67e38d34fa4f7965e9736b58f0cd

Keystream: 0x7673ab0ba17f911070fea363be4deaea

Plaintext: 0x70e93ea141e1fc673e017e97eadc6b96

Ciphertext: 0x069a95aae09e6d774effddf45491817c

Block #4

IV: 0x7673ab0ba17f911070fea363be4deaea

Keystream: 0x999b4ff25da506dc993e3284893d0b71

Plaintext: 0x8f385c2aecb03bfb32af3c54ec18db5c

Ciphertext: 0x16a313d8b1153d27ab910ed06525d02d

Case #7: Decrypting the 4 continuous block using AES OFB-encrypt with 128-bit key

Key: 0x7cc254f81be8e78d765a2e63339fc99a

IV: 0x67c6697351ff4aec29cdbaabf2fbe346

Block #1

Keystream: 0x80ed55152009abb9971985f536a5cc40

Ciphertext: 0xe6df58a2115108e3b24480e26e4c9294

Plaintext: 0x66320db73158a35a255d051758e95ed4

Block #2

IV: 0x80ed55152009abb9971985f536a5cc40

Keystream: 0xfa8e67e38d34fa4f7965e9736b58f0cd

Ciphertext: 0x513caa251680ae5e77e79d324a652c4a

Plaintext: 0xabb2cdc69bb454110e827441213ddc87

Block #3

IV: 0xfa8e67e38d34fa4f7965e9736b58f0cd

Keystream: 0x7673ab0ba17f911070fea363be4deaea

Ciphertext: 0x069a95aae09e6d774effddf45491817c

Plaintext: 0x70e93ea141e1fc673e017e97eadc6b96

Block #4

IV: 0x7673ab0ba17f911070fea363be4deaea

Keystream: 0x999b4ff25da506dc993e3284893d0b71

Ciphertext: 0x16a313d8b1153d27ab910ed06525d02d

Plaintext: 0x8f385c2aecb03bfb32af3c54ec18db5c

Annex E
(normative)

Protocol specific operation

E.1 Protocol specific information

The AES OFB cryptographic suite provides security services for UHF air interface protocols as described in E.2. ISO/IEC 29167-1 defines the Cryptographic suite Identifier (CSI) for AES OFB to be 000100_2 and it is expanded to the 8-bit value 04_h for use by all air interface protocols in this Annex.

E.2 Security services for ISO/IEC 18000-63

In this part of ISO/IEC 29167, the AES OFB cryptographic suite provides the following security services using the protocol commands shown in Table E.1. The mandatory security services shall be implemented in a compliant tag.

- Authentication (The operations of 'Authenticate' command are specified by 'Authentication Method' which value is given to 'AuthMethod')
  - Tag Authentication (Authentication type: AuthMethod = "000", Mandatory)
  - Interrogator Authentication (Authentication type: AuthMethod = "001", Optional)
  - Mutual Authentication (Authentication type: AuthMethod = "010", Mandatory)
  - Tag Authentication via Server (Authentication type: AuthMethod = "011", Optional)
  - CS_Initialization (Authentication type: AuthMethod "111", Mandatory)
- Challenge (Mandatory)
- KeyUpdate (Optional)

The *Challenge and/or Authenticate* command shall be used for all authentication methods implemented by a Tag. Tags shall implement Tag authentication, Mutual authentication, and CS_Initialization (Initialization Vector). Tags may implement Interrogator authentication, Tag authentication via server, and Challenge.

Table E.1 — Protocol Commands

Protocol Command	Feature	Mandatory, Prohibited, or Vendor defined
Challenge	Tag and Interrogator implement the command.	Vendor defined
	Tag ignores all other commands from an Interrogator during execution.	Mandatory
	Tag and Interrogator support sending ResponseBuffer during reply to an *ACK* command.	Mandatory
	Tag supports security timeout for a crypto error.	Vendor defined
Authenticate	Tag and Interrogator implement the command.	Mandatory

Table E.1 *(continued)*

Protocol Command	Feature	Mandatory, Prohibited, or Vendor defined
	Tag and Interrogator support sending the response during reply to the *Authenticate* command.	Mandatory
	Tag supports security timeout for a crypto error.	Vendor defined
	During authentication, a Tag does not reply to a command having an invalid handle or invalid CRC, the next Tag state is maintaining the current state. The crypto engine is reset by the CS_Initialization of Authenticate command.	Mandatory
SecureComm	This Cryptographic suite does not support this command.	Vendor defined
AuthComm	This Cryptographic suite does not support this command.	Prohibited
KeyUpdate	Tag and Interrogator implement the command.	Vendor defined
	Supported without encapsulation.	Vendor defined

E.3 Challenge (Mandatory)

E.3.1 General

Challenge command (Table E.2) allows an Interrogator to instruct multiple Tags to (simultaneously but independently) precompute cryptographic value(s) for use in a subsequent authentication and to store the precomputed value(s) for subsequent use. The generic nature of the Challenge command allows it to support a wide variety of cryptographic suites.

Challenge has the following fields:

— IncRepLen specifies whether the Tag omits or includes length in its stored reply. If IncRepLen=0 then the Tag omits length from its stored reply; if IncRepLen = 1 then the Tag includes length in its stored reply.

— Immed specifies whether a Tag concatenates response to its UII when replying to an ACK. If immed=0 then the Tag does not concatenate response to its UII when replying to an ACK but backscatters response by a ReadBuffer command; if immed=1 then the Tag backscatters UII + result when replying to an ACK.

— CSI selects the cryptographic suite that Tag and Interrogator use for the Challenge.

— Length is the message length, in bits.

— Message includes parameters for the authentication.

E.3.2 Command

Table E.2 — Command format

RnLen	ChLen	RnInt	ChInt
4	4	64	16 × n
Word number of RnInt (0100)	Word number of ChInt (n)	Random number	Random challenge

The Message consists of the length of random interrogator (RnLen), the length of challenge interrogator (ChLen), random interrogator (RnInt), and challenge interrogator (ChInt). RnLen and RnInt are for initializing an encryption engine. ChLen and ChInt are for tag authentication via the procedure of Mutual Authentication. Thus, Challenge command initializes an encryption engine and performs the first step of Mutual Authentication as shown in Figure E.1.

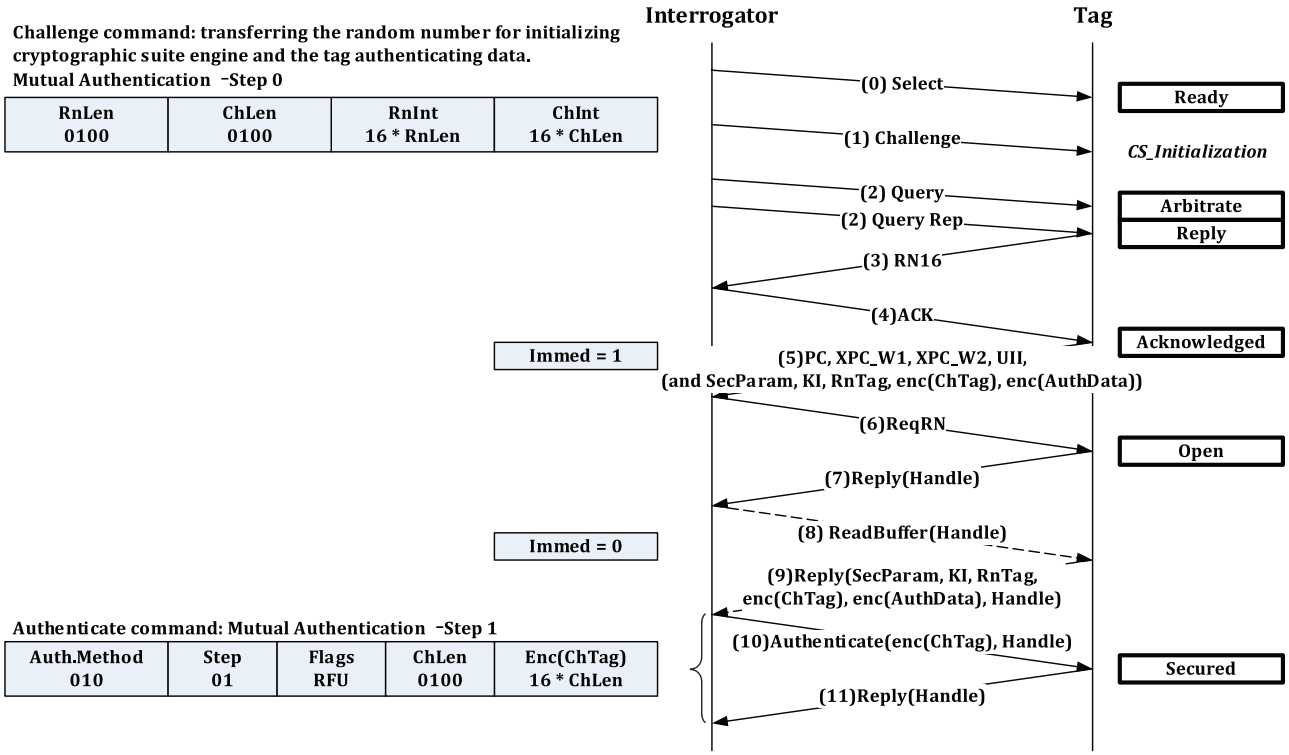

Figure E.1 — Diagram for 18000-63 Challenge

When the tag receives the Challenge command, the tag randomly generates the random tag (RnTag) for initializing an encryption engine and the challenge tag (ChTag) for authenticating the interrogator, and the tag generates the tag authentication data (AuthData) using the challenge interrogator (ChInt) and the challenge tag (ChTag). The authentication data (AuthData) is XOR value between the challenge interrogator (ChInt) and the challenge tag (ChTag), i.e. AuthData = ChInt $\oplus$ ChTag). Then, the tag encrypts the challenge tag (ChTag) and the tag authentication data (AuthData) and stores them to the buffer.

The interrogator receives the stored data through ACK command or ReadBuffer command according to the immed of Challenge command. The interrogator initializes an encryption engine and starts a tag authentication as the first step of Mutual Authentication by reading the stored data.

The interrogator initializes an encryption engine using the RnInt and RnTag, and the interrogator decrypts the challenge tag (ChTag) and the tag authentication data (AuthData). Then, the interrogator compares the decrypted tag authentication data (AuthData)) and the calculated authentication data using the challenge interrogator (ChInt) and the decrypted challenge tag (ChTag).

When the decrypted authentication data (AuthData) is determined to be identical to the calculated authentication data, the interrogator determines that the tag authentication succeeded. Otherwise, the interrogator determines that the tag authentication failed.

The above procedure is the first step of Mutual Authentication, and the interrogator completes the second step of Mutual Authentication through the Authenticate command.

When the authentication for the tag succeeds, the interrogator re-encrypts the decrypted challenge tag, and transmits the re-encrypted challenge tag (Enc(ChTag)) to the tag.

When the tag receives the re-encrypted challenge tag (Enc(ChTag)) from the interrogator, the tag decrypts the re-encrypted challenge tag (Enc(ChTag)) and compares the decrypted challenge tag with the generated challenge tag (ChTag) and performs the authentication for the interrogator.

When the decrypted challenge tag is determined to be identical to the generated tag (ChTag), the tag determines that the interrogator authentication succeeded. Otherwise, the tag determines that the interrogator authentication failed.

E.4 Example of Tag Authentication using *Authenticate*

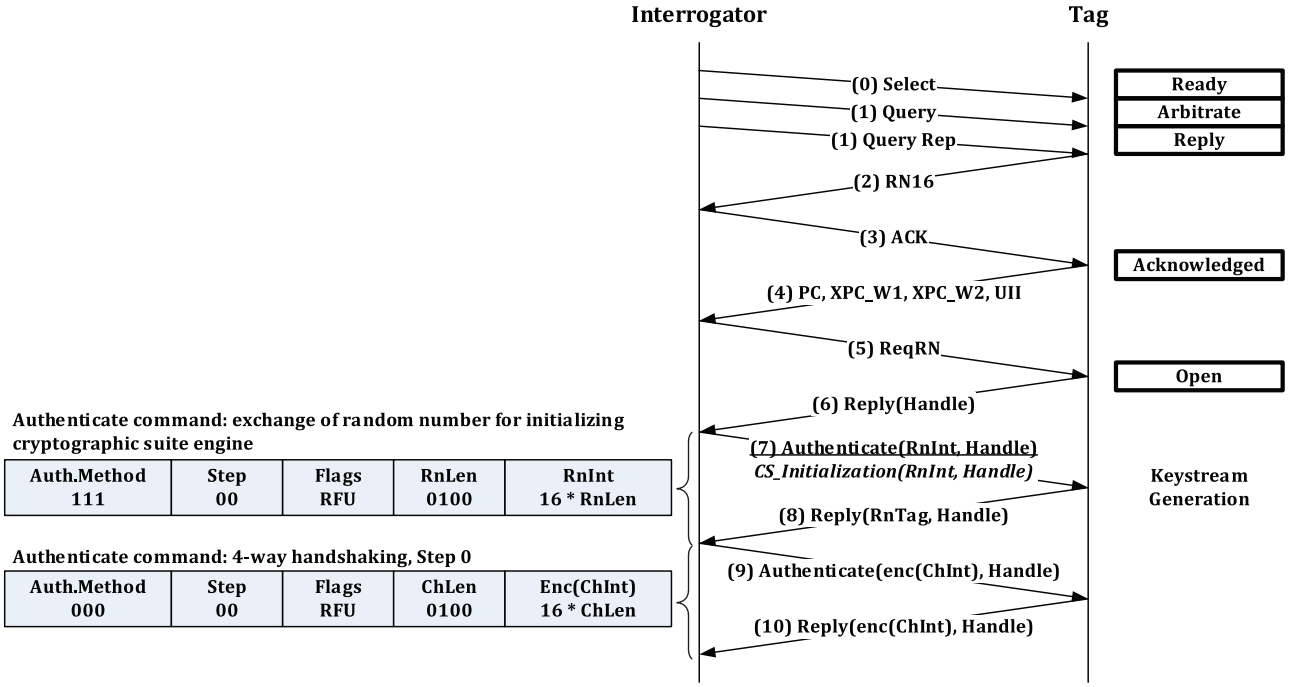

Figure E.2 — Diagram for 18000-63 Tag Authentication using Authenticate

E.5 Example of Interrogator Authentication using *Authenticate*

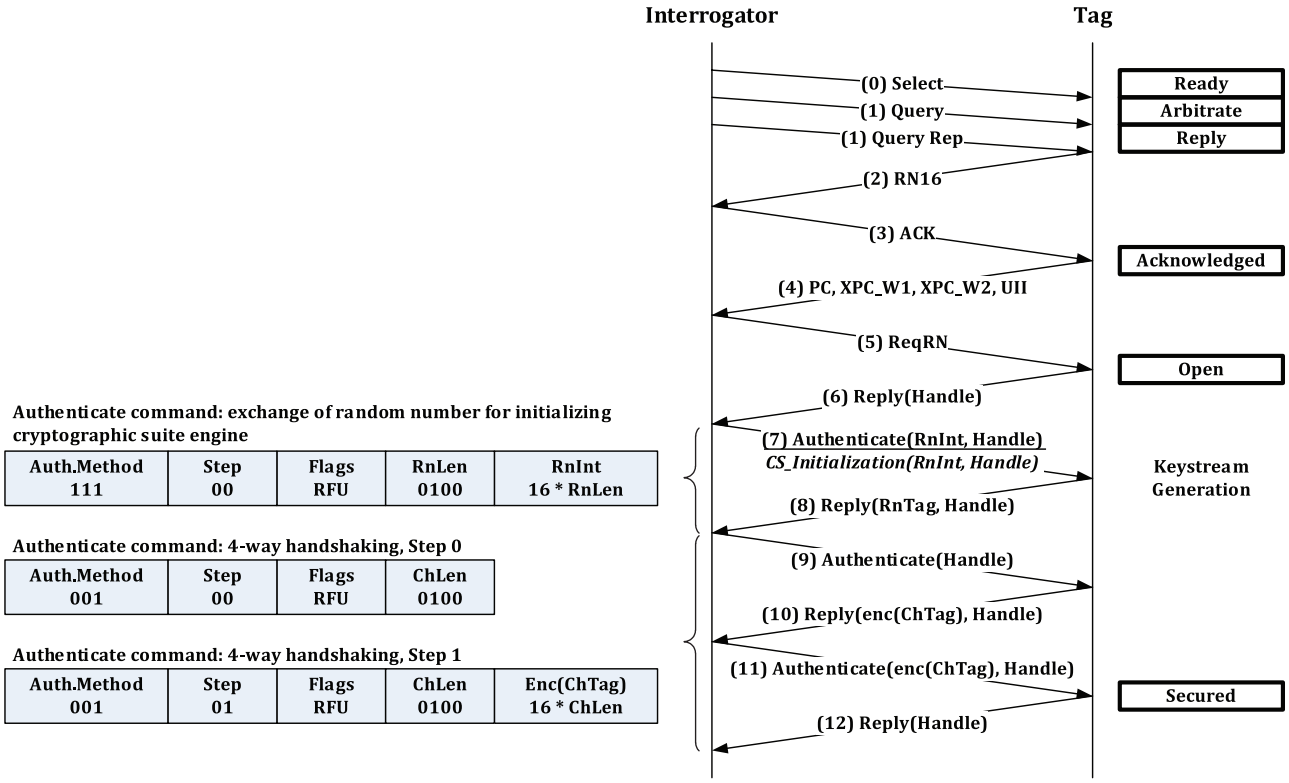

Figure E.3 — Diagram for 18000-63 Interrogator Authentication using *Authenticate*

E.6 Example of Mutual Authentication using *Authenticate*

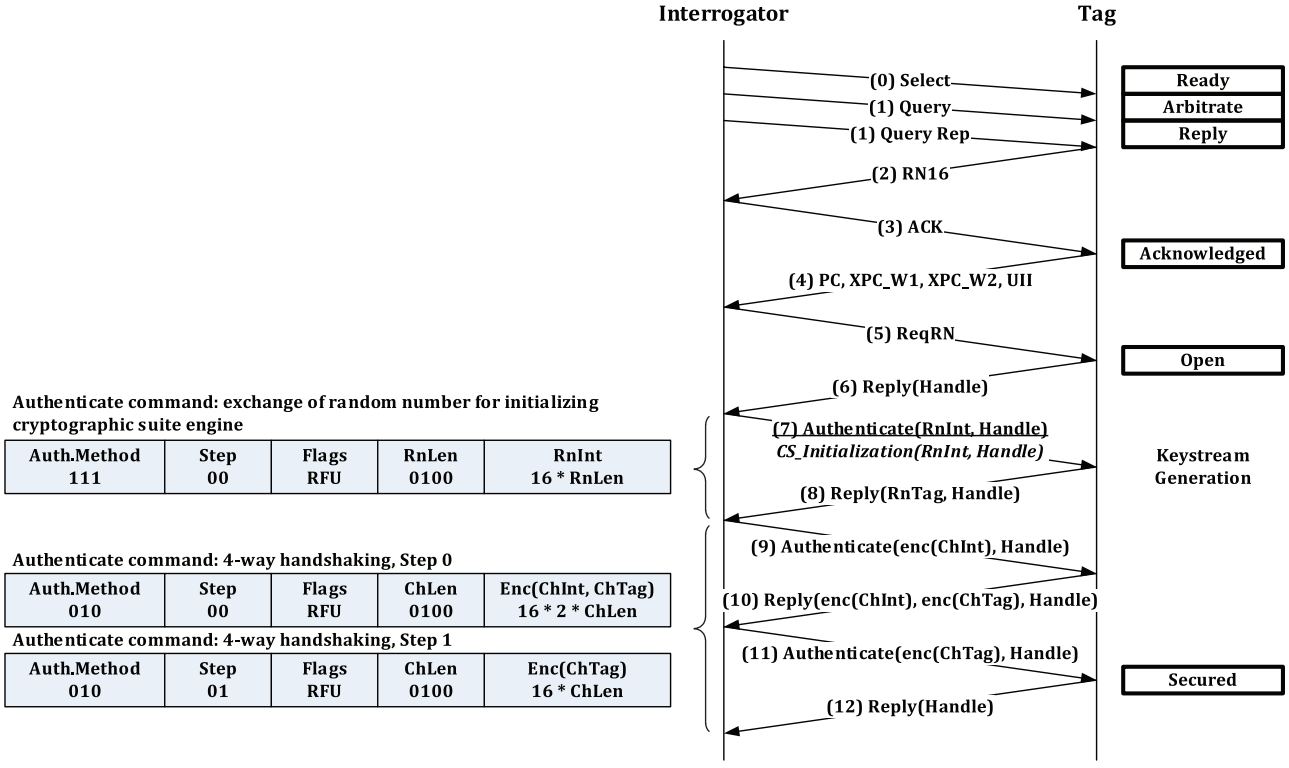

Figure E.4 — Diagram for 18000-63 Mutual Authentication using *Authenticate*

E.7 Summary of protocol security commands for ISO/IEC 18000-63

a) For each authentication method supported by crypto suite, it is specified whether the challenge command shall be supported.

b) The execution time for an authentication shall be below 5 ms.

c) The tag shall ignore commands from an interrogator during execution of a cryptographic operation.

d) The tag shall not support sending the contents of the response buffer in the reply to an ACK command.

e) The tag shall support sending the contents of the read buffer in the reply to a READ_BUFFER command.

f) The tag may support a security timeout following a crypto error.

g) For each authentication method supported by crypto suite, it is specified whether the authenticate command may be supported.

h) A Tag in any cryptographic state other than initial (i.e. state after power-up) shall reset its cryptographic engine and transition to the open state upon receiving an invalid command. (Invalid commands means crypto commands with incorrect handle or CRC error.)

i) For each Error Condition defined in the Cryptographic Suite:

   1) the Tag shall transition to the arbitrate state;

   2) the Tag may send an Error Code in case of a transition to the arbitrate state.

j) The Tag shall remain in its current state after a Tag Authentication. The Tag shall transition to the secured state after a successful Mutual or Interrogator Authentication.

k) The tag shall support at least one of the commands *Challenge* and *Authenticate*

l) For each encapsulation method supported by crypto suite, it is specified whether the AuthComm command shall not be supported.

m) For each encapsulation method supported by crypto suite, it is specified whether the SecureComm command shall not be supported.

n) The KeyUpdate command may be supported.

o) The KeyUpdate command, if supported, shall not be encapsulated

Annex F
(informative)

Tag authentication via server

F.1 Tag authentication via server (AuthMethod = “011”)

Tag authentication using the authentication server is performed after an Inventory process (from (0) to (6) in Figure F.1) of ISO/IEC 18000-63. We assume that the channel (from (11) to (12) in Figure F.1) between an interrogator and an authentication server is secure and out of scope of this part of ISO/IEC 29167.

Tag authentication using the authentication server is the process where an interrogator authenticates a tag via the authentication server which provides the functions of authentication. This type of authentication can be adopted for the case that an interrogator does not authenticate a tag.

For this process, an interrogator generates the challenge interrogator, and transmits the challenge interrogator to a tag. The ChLen is the length of the challenge number in words and the minimum length is 16 bits.

When the tag receives the challenge interrogator (ChInt), the tag randomly generates the challenge tag (ChTag), and generates the authentication data (AuthData) using the challenge interrogator (ChInt) and the challenge tag (ChTag). The authentication data (AuthData) is XOR value between the challenge interrogator (ChInt) and the challenge tag (ChTag), i.e., AuthData = ChInt $\oplus$ ChTag). Then, the tag encrypts the challenge tag and the authentication data and transmits them to the interrogator.

When the interrogator receives the encrypted challenge tag (Enc(ChTag)) and the encrypted authentication data(Enc(AuthData)), the interrogator transmits the challenge interrogator (ChInt), the encrypted challenge tag (Enc(ChTag)) and the encrypted authentication data (Enc(AuthData)) to an authentication server. That is, the interrogator transmits all together the PC, XPC_W1, XPC_W2, UII of the tag, the random interrogator (RnInt), the random tag (RnTag) for initializing cryptographic suite and the challenge interrogator (ChInt), the encrypted challenge tag (Enc(ChTag)) and the encrypted authentication data (Enc(AuthData)) to the authentication server. We assume that the channel between an interrogator and an authentication server is secure and out of scope of this part of ISO/IEC 29167.

Then, the authentication server decrypts the encrypted challenge tag (Enc(ChTag)) and the encrypted authentication data (Enc(AuthData)). Also, the authentication server generates another authentication data using the decrypted challenge tag and challenge interrogator (that is, the authentication server calculates XOR between the challenge tag (ChTag) and challenge interrogator (ChInt) for itself), and performs the authentication of the tag by comparing a decrypted authentication data (AuthData) with the calculated authentication data. Finally, the authentication server notifies the result of the authentication of the tag to the interrogator.

When the decrypted authentication data (AuthData) is determined to be identical to the calculated authentication data, the authentication server determines that the tag authentication succeeded. Otherwise, the authentication server determines that the tag authentication failed. A tag remains at the current State after the tag authentication using the authentication server.

For initializing cryptographic suite, the interrogator randomly generates the random interrogator (RnInt) and transmits the random interrogator (RnInt) to the tag. Also, the tag randomly generates the random tag (RnTag) and initializes the encryption engine and IV of tag using the random interrogator (RnInt) and random tag (RnTag). Also, the tag transmits the random tag (RnTag) to the interrogator then the interrogator transmits the random interrogator (RnInt) and random tag (RnTag) to the authentication server. The authentication server initializes the encryption engine and IV of the authentication server using the random interrogator (RnInt) and random tag (RnTag).

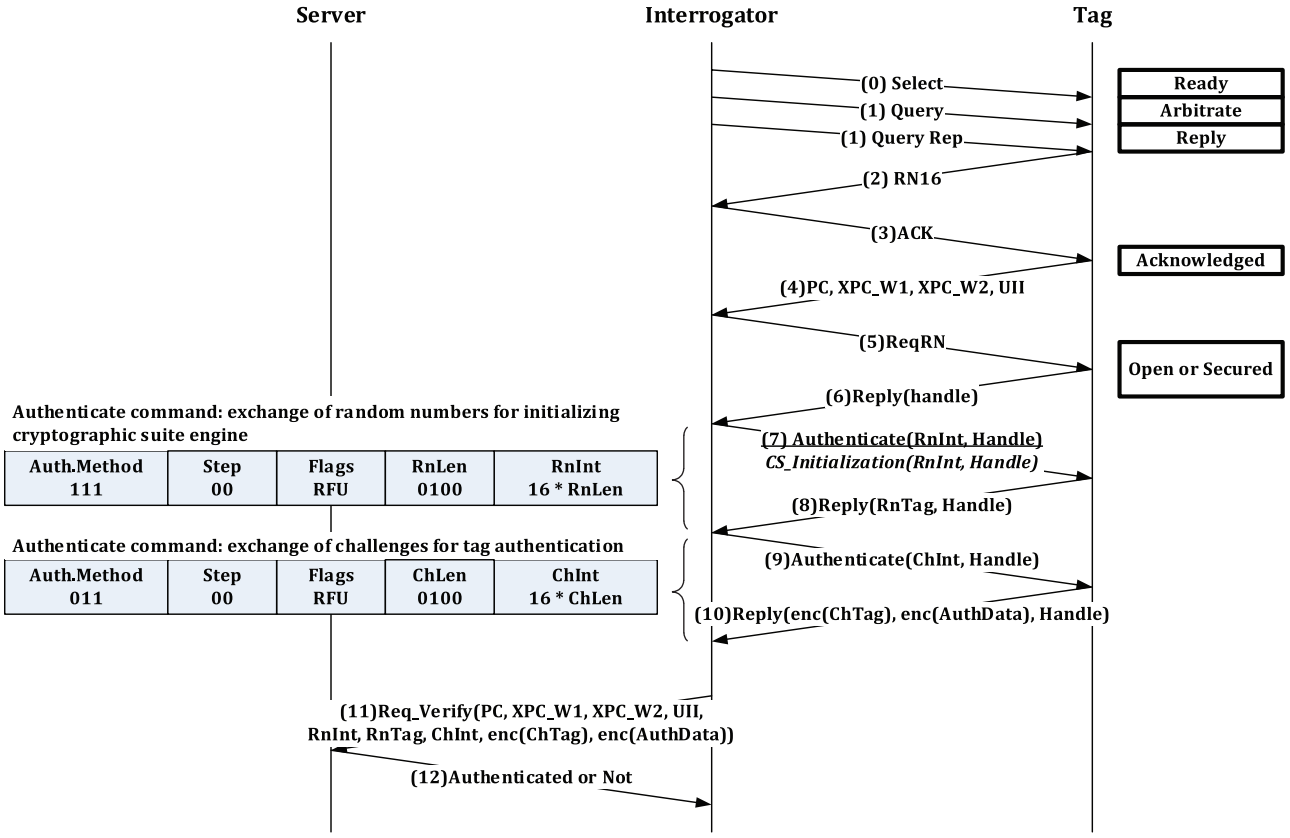

Figure F.1 — Diagram for Tag Authentication via Server

F.2 Commands and responses for tag authentication via server

F.2.1 General

This Clause describes the process of a tag authentication using an authentication server (from (7) to (12) in Figure F.1) between tag and interrogator. Figure F.1 shows a tag authentication method using an authentication server which is located in back-end system. According to the 'AuthMethod' and 'Step', the 'Authenticate' commands are used for exchanging random numbers and sending a challenge number. The commands and responses for the functions are as follows.

F.2.2 Authenticate Command (CS_Initialization)

In Figure F.1, the first 'Authenticate' command (Table F.1) is to exchange random numbers between tag and interrogator for initializing the IV. If the interrogator performed already this function at the current inventory, this step can be omitted. And its format is as follows. RnLen is the length of RnInt/RnTag in words and its value is 4. In other words, the length of RnInt/RnTag is fixed to the 4 words (64 bits). If the value of RnLen field is not 4, the tag ignores the command. CS_Initialization refers to 10.1.2.

Table F.1 — Authenticate (CS_Initialization)

AuthMethod	Step	Flags	RnLen	RnInt
111	00	000	0100	64-bit random

F.2.3 Response (CS_Initialization)

The following is the response (Table F.2) to the first Authenticate command. CS_Initialization refers to 10.1.2.

Table F.2 — Response (CS_Initialization)

Secure Parameter	KeyIndex	RnTag 64-bit random

F.2.4 Authenticate Command (Tag Authentication via Server)

In Figure F.1, the second 'Authenticate' command (Table F.3) is used for sending an interrogator challenge number and receiving a tag challenge number. And its format is as follows. The ChLen is the length of the challenge number in words.

Table F.3 — Authenticate (Tag Authentication via Server)

AuthMethod	Step	Flags	ChLen	ChInt
011	00	000	4-bit length	Plaintext

F.2.5 Response (Tag Authentication via Server)

The following is the response (Table F.4) to the second command. The length of 'Enc(ChTag)' and 'Enc(AuthData)' shall be the same as that of the received 'ChInt', respectively. 'AuthData' consists of XOR between 'ChInt' and 'ChTag'. A tag remains at the current State after the tag authentication using the authentication server.

Table F.4 — Response (Tag Authentication via Server)

Enc(ChTag) Ciphertext	Enc(AuthData) Authentication data (Ciphertext)

Bibliography

[1] ISO/IEC 15408-1, *Information technology — Security techniques — Evaluation criteria for IT security — Part 1: Introduction and general model*

[2] ISO/IEC 15408-2, *Information technology — Security techniques — Evaluation criteria for IT security — Part 2: Security functional components*

[3] ISO/IEC 15408-3, *Information technology — Security techniques — Evaluation criteria for IT security — Part 3: Security assurance components*

[4] ISO/IEC 15962, *Information technology — Radio frequency identification (RFID) for item management — Data protocol: data encoding rules and logical memory functions*

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

[6] NIST Special Publication 800-90A, Recommendation for Random Number Generation Using Deterministic Random Bit Generators, January 2012.

[7] ISO/IEC 18033-3, *Information technology — Security techniques — Encryption algorithms — Part 3: Block ciphers*

[8] NIST Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation: Methods and Techniques

ICS 35.040

Price based on 35 pages