
**Information technology — Automatic
identification and data capture
techniques —**

**Part 22:
Crypto suite SPECK security services
for air interface communications**

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

*Partie 22: Services de sécurité par suite cryptographique SPECK
pour communications par interface radio*



STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 29167-22:2018



COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2018

All rights reserved. Unless otherwise specified, or required in the context of its implementation, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO's member body in the country of the requester.

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

Published in Switzerland

Contents

Page

Foreword	v
Introduction	vi
1 Scope	1
2 Normative references	1
3 Terms, definitions, symbols and abbreviated terms	1
3.1 Terms and definitions	1
3.2 Symbols	2
3.3 Abbreviated terms	3
4 Conformance	3
4.1 Air interface protocol specific information	3
4.2 Interrogator conformance and obligations	3
4.3 Tag conformance and obligations	4
5 Introducing the SPECK cryptographic suite	4
6 Parameter and variable definitions	4
7 Crypto suite state diagram	5
8 Initialization and resetting	6
9 Authentication	6
9.1 General	6
9.2 Message and response formatting	6
9.3 Tag authentication (AuthMethod "00")	7
9.3.1 General	7
9.3.2 TAM1 message	7
9.3.3 Intermediate Tag processing	8
9.3.4 TAM1 response	8
9.3.5 Final Interrogator processing	8
9.4 Interrogator authentication (AuthMethod "01")	9
9.4.1 General	9
9.4.2 IAM1 message	9
9.4.3 Intermediate Tag processing #1	10
9.4.4 IAM1 response	10
9.4.5 Intermediate Interrogator processing	10
9.4.6 IAM2 message	10
9.4.7 Intermediate Tag processing #2	11
9.4.8 IAM2 response	11
9.4.9 Final Interrogator processing	12
9.5 Mutual authentication (AuthMethod "10")	12
9.5.1 General	12
9.5.2 MAM1 message	13
9.5.3 Intermediate Tag processing #1	13
9.5.4 MAM1 response	14
9.5.5 Intermediate Interrogator processing	14
9.5.6 MAM2 message	14
9.5.7 Intermediate Tag processing #2	15
9.5.8 MAM2 response	15
9.5.9 Final Interrogator processing	16
10 Communication	16
10.1 General	16
10.2 Message and response formatting	17
10.3 Transforming a payload prior to encapsulation	17
10.3.1 General	17

10.3.2	Encapsulating an Interrogator command	19
10.3.3	Cryptographically protecting a Tag reply	20
10.4	Processing an encapsulated or cryptographically-protected reply	20
10.4.1	General	20
10.4.2	Recovering an encapsulated Interrogator command	21
10.4.3	Recovering a cryptographically-protected Tag response	22
11	Key table and key update	22
Annex A (normative)	Crypto suite state transition table	24
Annex B (normative)	Errors and error handling	25
Annex C (normative)	Description of SPECK and SILC v3	26
Annex D (informative)	Test vectors	30
Annex E (normative)	Protocol specific information	42
Bibliography		45

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 29167-22:2018

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 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 the following URL: www.iso.org/iso/foreword.html.

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

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

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.

Introduction

This document specifies a variety of security services provided by the lightweight block cipher SPECK. While SPECK supports various key and block sizes, the cipher versions that are supported in this cryptographic suite take the following block/key sizes in bits: 64/96, 96/96, 64/128, 128/128, and 128/256.

The International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC) draw attention to the fact that it is claimed that compliance with this document may 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
Impinj, Inc. 400 Fairview Ave N, # 1200 Seattle, WA 98109 USA

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

The latest information on IP that may be applicable to this document can be found at www.iso.org/patents.

Information technology — Automatic identification and data capture techniques —

Part 22:

Crypto suite SPECK security services for air interface communications

1 Scope

This document defines the crypto suite for SPECK for the ISO/IEC 18000 air interfaces standards for radio frequency identification (RFID) devices. Its purpose is to provide a common crypto suite for security for RFID devices that can be referred to by ISO committees for air interface standards and application standards. The crypto suite is defined in alignment with existing air interfaces.

SPECK is a symmetric block cipher that is parameterized in both its block length and key length. In this document, a variety of block/key length options are supported.

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 18000-63, *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, *Information technology — Automatic identification and data capture (AIDC) techniques — Harmonized vocabulary*

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 terminological databases for use in standardization at the following addresses:

- ISO Online browsing platform: available at <https://www.iso.org/obp>
- IEC Electropedia: available at <http://www.electropedia.org/>

3.1.1

bit string

ordered sequence of 0s and 1s

3.1.2

block cipher

family of permutations that is parameterized by a cryptographic key and, optionally, the block size

3.1.3

block size

number of bits in a *data block* (3.1.6) that is an input (or output) of the *block cipher* (3.1.2)

3.1.4

cryptographic key

string of bits of length given by *key size* (3.1.7) that is used by the *block cipher* (3.1.2) to transform some *data block* (3.1.6)

3.1.5

command

<message> data that the Interrogator sends to the Tag with Message as parameter

3.1.6

data block

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

3.1.7

key size

length in bits of the *cryptographic key* (3.1.4) that is used by the *block cipher* (3.1.2)

3.1.8

message

part of the *command* (3.1.5) that is defined by the crypto suite

3.1.9

nonce

data block (3.1.6) that, within the parameters of typical use, can be assumed to be non-repeating

3.1.10

SPECK-b/k-ENC(key, data)

SPECK encryption of a *b*-bit *data block* (3.1.6) using a *k*-bit *cryptographic key* (3.1.4)

3.1.11

SPECK-b/k-DEC(key, data)

SPECK decryption of a *b*-bit *data block* (3.1.6) using a *k*-bit *cryptographic key* (3.1.4)

3.1.12

Reply

<response> data that the Tag returns to the Interrogator with *Response* (3.1.13) as parameter

3.1.13

Response

part of the *Reply* (3.1.12) (stored or sent) that is defined within the crypto suite

3.2 Symbols

XXXX₂

Binary notation

XXXX_h

Hexadecimal notation

||

Concatenation of syntax elements, transmitted in the order written

∅

The empty string, typically used to indicate a deliberately empty input or omitted field

|A|

The bit-wise length of the string A expressed as an integer

Example 1: $|0000_2| = 4$.

Example 2: $|0000_h| = 16$.

Example 3: $|\emptyset| = 0$.

fix1(A) The string obtained by fixing the first (leftmost) bit to 1₂

Example 1: **fix1**(0000₂) = 1000₂.

Example 2: **fix1**(0000_h) = 8000_h.

Example 3: **fix1**($\emptyset$) = $\emptyset$.

msb_n(A) The *n*-bit binary string obtained by taking the first (leftmost) *n* bits of the binary representation of A

Example 1: **msb**₃(1010₂) = 101₂.

Example 2: **msb**₇(ABCD_h) = 1010101₂.

Example 3: **msb**₇($\emptyset$) = $\emptyset$.

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 $\emptyset$.

Key.KeyID The cryptographic key identified and indexed by the numerical value KeyID

3.3 Abbreviated terms

CS	Crypto Suite
CSI	Crypto Suite Indicator
RFU	Reserved for future use

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".

4.2 Interrogator conformance and obligations

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

An Interrogator may 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 obligations

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

A Tag may implement any subset of the optional commands defined in this document.

A 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 Introducing the SPECK cryptographic suite

SPECK is a lightweight Feistel block cipher that is suitable for extremely constrained environments such as RFID Tags. The details of the operation of the SPECK cipher are described in [Annex C](#).

The background for the development of SPECK and its design principles are described in Reference [3].

SPECK is parameterized in terms of the block size, denoted b , and the key size denoted k . A particular variant of SPECK will be denoted SPECK- b/k throughout this document. While Reference [3] offers many different choices to the block and key size, this cryptographic suite only supports the five parameter combinations given in [Table 1](#):

Table 1 — Variants of SPECK- b/k supported in this document

	SPECK-64/96	SPECK-64/128	SPECK-96/96	SPECK-128/128	SPECK-128/256
Block size (b bits)	64	64	96	128	128
Key size (k bits)	96	128	96	128	256

It is possible that not all variants of SPECK will be cryptographically suited to all applications. Guidance on the appropriate variant for a given application lies outside the scope of this document and a thorough security and risk assessment is advised before deployment.

Test vectors for the components of this document are provided in [Annex D](#).

6 Parameter and variable definitions

[Table 2](#) lists the variables and constants that are used in this document.

Table 2 — SPECK cryptographic suite variables and constants

Parameter	Description
ICChallenge-b/k	A challenge generated at random by the Interrogator. The length of ICChallenge- b/k depends on the values of b and k .
TChallenge-b/k	A challenge generated at random by the Tag. The length of TChallenge- b/k depends on the values of b and k .
TRnd-b/k	A salt value generated at random by the Tag. The length of TRnd- b/k depends on the values of b and k .
IRnd-b/k	A salt value generated at random by the Interrogator. The length of IRnd- b/k depends on the values of b and k .
C_TAM-b/k	A pre-defined constant. The length and value of C_TAM- b/k depends on the values of b and k .
C_IAM-b/k	A pre-defined constant. The length and value of C_IAM- b/k depends on the values of b and k .
C_MAM-b/k	A pre-defined constant. The length and value of C_MAM- b/k depends on the values of b and k .
Key.0 ... Key.255	A set of up to 256 keys Key.0 through to Key.255. Not all key values need to be specified. However Key. j shall not be specified when there remain unspecified Key. i with $i < j$.

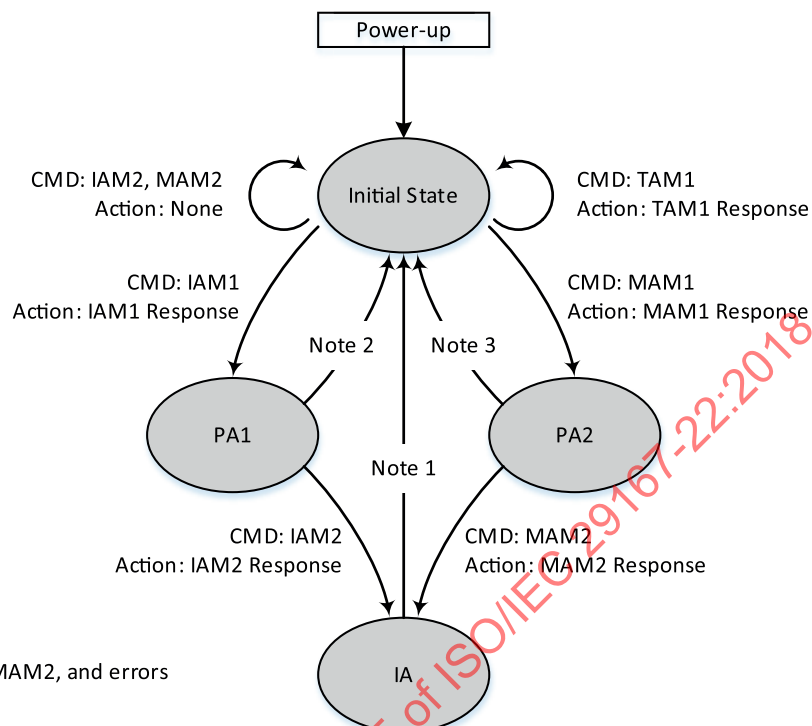
Table 3 gives the values of C_TAM- b/k , C_IAM- b/k and C_MAM- b/k that are used in this document. For a given choice of operational parameters, the length of these constants depends on the block size b .

Table 3 — Values of C_TAM- b/k , C_IAM- b/k , and C_MAM- b/k for different values of b and k and different parameter sets PS

b/k	64/96	64/128	96/96	128/128	128/256
C_TAM-b/k	11 ₂	11 ₂	FF _h	FFFF _h	FFFF _h
C_IAM-b/k	10 ₂	10 ₂	FE _h	FFFE _h	FFFE _h
C_MAM-b/k for PS=00₂	01 ₂	01 ₂	FD _h	FFFD _h	FFFD _h
C_MAM-b/k for PS=01₂	1 _h	1 _h	D _h	FD _h	FD _h

7 Crypto suite state diagram

After power-up and after a reset, the Cryptographic Suite shall transition into the **Initial** state, state transitions shall be defined by Annex A, and error handling shall be defined by Annex B. See Figure 1.



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

Figure 1 — Crypto suite state diagram

8 Initialization and resetting

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

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

- after the completion of each cryptographic protocol,
- if some cryptographic protocol is abandoned or incomplete, and
- after reset.

9 Authentication

9.1 General

This document supports Tag authentication, Interrogator authentication and Mutual authentication.

This clause describes the details of the messages and responses that are exchanged between the Interrogator and Tag for each of the authentication methods.

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.

9.3 Tag authentication (AuthMethod “00”)

9.3.1 General

Tag authentication uses a challenge-response protocol. See Figure 2.

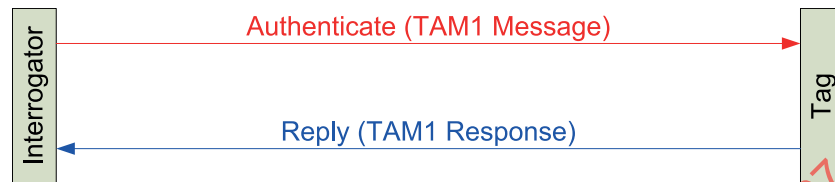


Figure 2 — Tag authentication via a challenge-response scheme

The parameter set PS defined in Table 4 gives the lengths of different fields for different block and key sizes.

NOTE The parameter set PS=00₂ closely matches other parts of the ISO/IEC 29167 series, most notably 29167-10. This provides some drop-in compatibility between SPECK-128/128 and AES-128.

Table 4 — Parameter set PS = 00₂ for Tag authentication

Parameter set PS= 00 ₂					
<i>b/k</i>	64/96	64/128	96/96	128/128	128/256
$t = \text{IChecksum-}b/k $	42	42	56	80	80
$r = \text{TRnd-}b/k $	20	20	32	32	32
$c = \text{C_TAM-}b/k $	2	2	8	16	16

9.3.2 TAM1 message

The Interrogator shall generate a random Interrogator challenge (IChecksum-*b/k*) that is carried in the TAM1 message. The Interrogator shall also indicate the variant of SPECK to be used.

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

NOTE 2 Mechanisms to generate the random Interrogator challenge lie outside the scope of this document.

Table 5 — TAM1 message format

Field	Payload							
	AuthMethod	Step	RFU	BlockSize	KeySize	KeyID	PS	Challenge
Length (bits)	2	2	2	2	2	8	2	<i>t</i>
Value	00 ₂	00 ₂	00 ₂	00 ₂ : <i>b</i> =64 01 ₂ : <i>b</i> =96 10 ₂ : <i>b</i> =128 11 ₂ : RFU	00 ₂ : <i>k</i> =96 01 ₂ : <i>k</i> =128 10 ₂ : <i>k</i> =256 11 ₂ : RFU	variable	00 ₂	IChecksum- <i>b/k</i>

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.

The Tag shall check if the Step is "00₂". If the value of Step is different, the Tag shall return a "Not Supported" error.

The Tag shall check if the RFU is "00₂". If the value of RFU is different, the Tag shall return a "Not Supported" error.

The Tag shall check whether the values of BlockSize and KeySize are supported by the Tag. If at least one of these checks is failed, the Tag shall return a "Not Supported" error.

The Tag shall check whether the values of BlockSize and KeySize are supported by Key.KeyID and that Key.KeyID is authorized for use in Tag authentication. If either (or both) of these checks is (are) failed, the Tag shall return a "Not Supported" error.

The Tag shall check whether the parameter set PS is supported. If the parameter set PS is not supported, the Tag shall return a "Not Supported" error.

Assuming that the TAM1 message is successfully parsed by the Tag, the Tag shall prepare the TAM1 response.

9.3.4 TAM1 response

The Tag shall generate a random salt TRnd- b/k of length r bits where r is given for the parameter set in [Table 3](#).

The Tag shall use Key.KeyID and SPECK encryption to form a b -bit string TResponse such that:

$$\text{TResponse} = \text{SPECK-}b/k\text{-ENC} (\text{Key.KeyID}, C_{\text{TAM-}b/k} \parallel \text{TRnd-}b/k \parallel \text{IChallenge-}b/k).$$

The Tags shall return TResponse to the Interrogator.

NOTE 1 Only one input block of b bits is encrypted and so only one invocation of SPECK- b/k is required.

NOTE 2 Appropriate mechanisms to generate TRnd- b/k lie outside the scope of this document.

Table 6 — TAM1 response format

Field	Payload
Length (bits)	Tag Response
Value	b
	TResponse

9.3.5 Final Interrogator processing

After receiving TAM1 response the Interrogator shall use Key.KeyID to compute the b -bit string S where:

$$S = \text{SPECK-}b/k\text{-DEC} (\text{Key.KeyID}, \text{TResponse}).$$

1. The Interrogator shall check that $S[t-1:0] = \text{IChallenge-}b/k$.
2. The Interrogator may check that $S[b-1:b-c] = C_{\text{TAM-}b/k}$.

If these verification steps are successfully completed, the Interrogator may conclude 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 may conclude that the Tag is authentic.

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

9.4 Interrogator authentication (AuthMethod “01”)

9.4.1 General

Interrogator authentication uses a challenge-response protocol. See Figure 3.

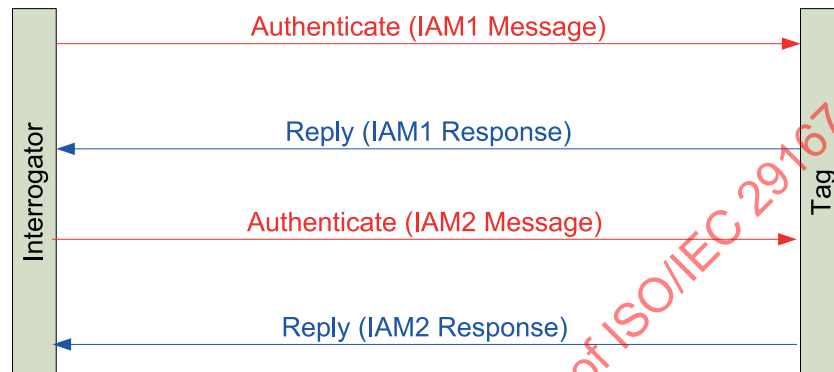


Figure 3 — Interrogator authentication via a challenge-response scheme

The parameter set in Table 7 gives the lengths of specific data fields for different choices of block and key size.

NOTE The parameter set PS=00₂ closely matches other parts of the ISO/IEC 29167 series, most notably 29167-10. This provides some drop-in compatibility between SPECK-128/128 and AES-128.

Table 7 — Parameter set PS = 00₂ for Interrogator authentication

Parameter set PS= 00 ₂					
b/k	64/96	64/128	96/96	128/128	128/256
$t = TChallenge-b/k $	42	42	56	80	80
$r = IRnd-b/k $	20	20	32	32	32
$c = I_MAM-b/k $	2	2	8	16	16

9.4.2 IAM1 message

The Interrogator shall send an initial message IAM1 to the Tag prompting the Tag to start a challenge-response exchange.

The Interrogator shall also indicate the variant of SPECK to be used.

NOTE The variant(s) of SPECK deployed on a device is (are) manufacturer dependent.

Table 8 — IAM1 message format

Field	Payload						
	AuthMethod	Step	RFU	BlockSize	KeySize	KeyID	PS
Length (bits)	2	2	2	2	2	8	2
Value	01 ₂	00 ₂	00 ₂	00 ₂ : $b=64$ 01 ₂ : $b=96$ 10 ₂ : $b=128$ 11 ₂ : RFU	00 ₂ : $k=96$ 01 ₂ : $k=128$ 10 ₂ : $k=256$ 11 ₂ : RFU	variable	00 ₂

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.

If Interrogator authentication is not supported on the Tag, *i.e.* if "01₂" is not a valid value for AuthMethod, then the Tag shall return a "Not Supported" error condition.

The Tag shall check if the Step is "00₂". If the value of Step is different, the Tag shall return a "Not Supported" error.

The Tag shall check if the RFU is "00₂". If the value of RFU is different, the Tag shall return a "Not Supported" error.

The Tag shall check whether the values of BlockSize and KeySize are supported by the Tag. If at least one of these checks is failed, the Tag shall return a "Not Supported" error.

The Tag shall check whether the values of BlockSize and KeySize are supported by Key.KeyID and that Key.KeyID is authorized for use in Interrogator authentication. If at least one of these checks is failed, the Tag shall return a "Not Supported" error.

The Tag shall check whether the value of parameter set PS is supported by the Tag. If not, the Tag shall return a "Not Supported" error.

If the IAM1 message is successfully parsed by the Tag, the Tag shall calculate the IAM1 response.

9.4.4 IAM1 response

The Tag shall generate a random challenge TChallenge- b/k of length t bits, where t is determined by the parameter set, and shall send this to the Interrogator.

Table 9 — IAM1 response format

Field	Payload
	Challenge
Length (bits)	t
Value	TChallenge- b/k

9.4.5 Intermediate Interrogator processing

The Interrogator shall construct the IAM2 message.

9.4.6 IAM2 message

The Interrogator shall form a b -bit string IResponse such that:

$IResponse = SPECK-b/k-DEC (Key.KeyID, C_IAM-b/k \parallel IRnd-b/k \parallel TChallenge-b/k)$.

The Interrogator shall send IResponse to the Tag as part of the IAM2 message; see [Table 10](#).

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

Table 10 — IAM2 message format

Field	Payload			
	AuthMethod	Step	RFU	InterrogatorResponse
Length (bits)	2	2	4	b
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 "01₂" is not a valid value for AuthMethod, then the Tag shall return a "Not Supported" error condition.

The Tag shall check if the Step is "01₂". If the value of Step is different, the Tag shall return a "Not Supported" error.

The Tag shall check if the RFU is "0000₂". If the value of RFU is different, the Tag shall return a "Not Supported" error.

If the IAM2 Message is successfully parsed by the Tag, the Tag shall check the returned value of IResponse in the following manner.

The Tag shall use Key.KeyID to compute the b -bit string S where:

$$S = SPECK-b/k-ENC (Key.KeyID, IResponse).$$

1. The Tag shall check that $S[t-1:0] = TChallenge-b/k$.
2. The Tag may check that $S[b-1:b-c] = C_IAM-b/k$.

If the checks performed by the Tag are successful then the Tag may conclude 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 may conclude that the Interrogator is authentic and TStatus is set to 1₂. Otherwise TStatus is set to 0₂.

The Tag shall prepare IAM2 response.

9.4.8 IAM2 response

The Tag shall return the value of TStatus to the Interrogator. If TStatus = 1₂, then the cryptographic state machine moves to state **IA** (see [Clause 7](#)).

Table 11 — IAM2 response format

Field	Payload
	Status
Length (bits)	1
Value	TStatus

9.4.9 Final Interrogator processing

The Interrogator receives IAM2 Response.

If the value of TStatus is 1_2 then the Interrogator may assume 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.

9.5 Mutual authentication (AuthMethod “10”)

9.5.1 General

Mutual Interrogator-Tag authentication uses an interleaved challenge-response protocol.

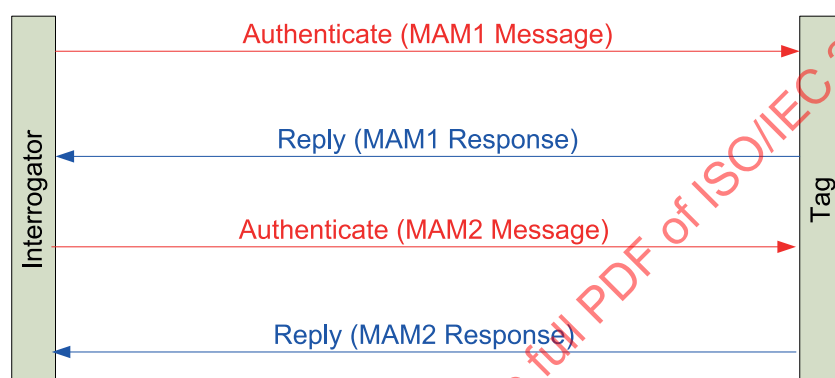


Figure 4 — Interrogator-Tag mutual authentication via an interleaved challenge-response scheme

The parameter set in [Table 12](#) gives the lengths of specific data fields for different choices of block and key size.

NOTE 1 The parameter set PS= 00_2 closely matches other parts of the ISO/IEC 29167 series, most notably 29167-10. This provides some drop-in compatibility between SPECK-128/128 and AES-128.

NOTE 2 The parameter set PS= 01_2 allows a more efficient mutual authentication protocol than PS= 00_2 . In particular, the length of the Tag and Interrogator challenges are chosen so that both can fit within a single b -bit block.

Table 12 — Parameter sets for mutual Tag-Interrogator authentication

Parameter set PS= 00_2					
b/k	64/96	64/128	96/96	128/128	128/256
$t = TChallenge-b/k $	42	42	56	80	80
$t = ICChallenge-b/k $	42	42	56	80	80
$c = C_MAM-b/k $	2	2	8	16	16
Parameter set PS= 01_2					
b/k	64/96	64/128	96/96	128/128	128/256
$t = TChallenge-b/k $	30	30	46	60	60
$t = ICChallenge-b/k $	30	30	46	60	60
$c = C_MAM-b/k $	4	4	4	8	8

9.5.2 MAM1 message

The Interrogator shall generate a random Interrogator challenge (IC_{Challenge-b/k}) that is carried in the MAM1 message. The length of IC_{Challenge-b/k} is denoted t and specified in Table 12.

The Interrogator shall also indicate the variant of SPECK to be used.

NOTE The variant(s) of SPECK deployed on a device is (are) manufacturer dependent.

Table 13 — MAM1 message format

Fields	Payload							
	AuthMethod	Step	RFU	BlockSize	KeySize	KeyID	PS	Challenge
Length (bits)	2	2	2	2	2	8	2	t
Value	10 ₂	00 ₂	00 ₂	00 ₂ : $b=64$ 01 ₂ : $b=96$ 10 ₂ : $b=128$ 11 ₂ : RFU	00 ₂ : $k=96$ 01 ₂ : $k=128$ 10 ₂ : $k=256$ 11 ₂ : RFU	variable	variable	IC _{Challenge-b/k}

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.

If Mutual authentication is not supported on the Tag, *i.e.* if "10₂" is not a valid value for AuthMethod, then the Tag shall return a "Not Supported" error condition.

The Tag shall check if the Step is "00₂". If the value of Step is different, the Tag shall return a "Not Supported" error.

The Tag shall check if the RFU is "00₂". If the value of RFU is different, the Tag shall return a "Not Supported" error.

The Tag shall check whether the values of BlockSize and KeySize are supported by the Tag. If at least one of these checks is failed, the Tag shall return a "Not Supported" error.

The Tag shall check whether the values of BlockSize and KeySize are supported by Key.KeyID and that Key.KeyID is authorized for use in Interrogator-Tag mutual authentication. If at least one of these checks is failed, the Tag shall return a "Not Supported" error.

The Tag shall check whether the value of parameter set PS is supported by the Tag. If not, the Tag shall return a "Not Supported" error.

Assuming the MAM1 message is successfully parsed by the Tag, the Tag shall calculate its response accordingly.

The Tag shall generate a random challenge T_{Challenge-b/k} of length t bits.

The Tag shall construct a b -bit string by concatenating C_{MAM-b/k} with the $(b-t-c)$ most significant bits of T_{Challenge-b/k} and the entirety of IC_{Challenge-b/k}.

The Tag shall use Key.KeyID to compute the b -bit string S where:

$$S = \text{SPECK-}b/k\text{-ENC} (\text{Key.KeyID}, \text{C_MAM-}b/k \parallel \text{TChallenge-}b/k [t-1:2t-b+c] \parallel \text{ICChallenge-}b/k).$$

TResponse is a string that consists of S concatenated with the remainder (if any) of T_{Challenge-b/k}

$$\text{TResponse} = \text{TChallenge-}b/k [2t-b+c-1:0] \parallel S.$$

NOTE 1 Only one input block of b bits is encrypted and so only one invocation of SPECK- b/k is required.

NOTE 2 Parameter set PS=01₂ is constructed so that TResponse is exactly b bits long; *i.e.* TResponse = S.

9.5.4 MAM1 response

The Tag returns the value of TResponse to the Interrogator.

Table 14 — MAM1 response format

Field	Payload
	Tag Response
Length (bits)	$2t + c$
Value	TResponse

9.5.5 Intermediate Interrogator processing

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

$$T = \text{SPECK-}b/k\text{-DEC} (\text{Key.KeyID}, \text{TResponse}[b-1:0]).$$

1. The Interrogator shall check that $T[t-1:0] = \text{IChallenge-}b/k$.
2. The Interrogator may check that $T[b-1:b-c] = \text{C_TAM-}b/k$.

If these verification steps are not successful, the Interrogator shall abandon the cryptographic protocol. Otherwise, the Interrogator may conclude 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 may conclude that the Tag is authentic.

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

9.5.6 MAM2 message

If the cryptographic protocol has not been abandoned, the Interrogator shall form a b -bit string IResponse depending on the parameter set PS as follows:

1. If PS = 00₂ then IResponse is equal to
 $\text{SPECK-}b/k\text{-DEC} (\text{Key.KeyID}, \text{C_MAM-}b/k \parallel T[b-t-c-1:0] \parallel T[b-c-1:t] \parallel \text{TResponse}[2t+c-1:b]).$
2. If PS = 01₂ then IResponse is equal to $T[b-c:t]$.

The Interrogator shall set SecureComm = 0001₂ if secure communications as described in [Clause 10](#) will be used after mutual authentication is completed. Otherwise the Interrogator shall set SecureComm = 0000₂.

The Interrogator shall send IResponse and the value of SecureComm to the Tag as part of the MAM2 message, see [Table 15](#).

Table 15 — MAM2 message format

Field	Payload				
	AuthMethod	Step	RFU	SecureComm	Interrogator Response
Length (bits)	2	2	4	4	variable
Value	10 ₂	01 ₂	0000 ₂	variable	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 Mutual authentication is not supported on the Tag, i.e. if "10₂" is not a valid value for AuthMethod, then the Tag shall return a "Not Supported" error condition.

The Tag shall check if the Step is "01₂". If the value of Step is different, the Tag shall return a "Not Supported" error.

The Tag shall check if the RFU is "0000₂". If the value of RFU is different, the Tag shall return a "Not Supported" error.

Assuming that the MAM2 Message is successfully parsed by the Tag, the Tag shall check the returned value of IResponse in the following manner.

1. If PS = 00₂ then the Tag computes the b -bit string $S = \text{SPECK-}b/k\text{-ENC}(\text{Key.KeyID}, \text{IResponse})$.

$\text{SPECK-}b/k\text{-DEC}(\text{Key.KeyID}, \text{C_MAM-}b/k \parallel \text{T}[b-t-c-1:0] \parallel \text{T}[b-c-1:t] \parallel \text{TResponse}[2t+c-b-1:0])$.

The Tag shall check if $S[t-1:0] = \text{TChallenge-}b/k$.

The Tag may check if $S[b-c-1:t] = \text{IChallenge-}b/k[b-t-c-1:0]$

The Tag may check if $S[b-1:b-c] = \text{C_MAM-}b/k$.

2. If PS = 01₂ then the Tag shall check whether $S = \text{TChallenge-}b/k$.

NOTE No encryption operation is required in the case of PS = 01₂.

If the checks performed by the Tag are successful then the Tag may conclude 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 may conclude that the Interrogator is authentic and TStatus is set to 1₂. Otherwise TStatus is set to 0₂.

If TStatus = 0₂, the Tag sets $N_T = \emptyset$.

If TStatus = 1₂ and SecureComm = 0₂, the Tag sets $N_T = \emptyset$.

If TStatus = 1₂ and SecureComm = 1_h, the Tag shall generate a random string N_T of the length shown in [Table 16](#). The Tag shall further indicate using KeyID₂ which key shall be used for the subsequent secure communication session.

Table 16 — Length of N_T for different values of b and k

b/k	64/96	64/128	96/96	128/128	128/256
$ N_T $ for PS = 00 ₂	6	6	24	32	32
$ N_T $ for PS = 01 ₂	18	18	34	52	52

9.5.8 MAM2 response

The Tag shall prepare and send a MAM2 response as specified in [Table 17](#).

If TStatus = 1₂ then the cryptographic state machine moves to state **IA** (see [Clause 7](#)).

Table 17 — MAM2 response format

Field	Payload		
	Status	KeyID	Pre-IV
Length (bits)	1	8	variable
Description	TStatus	KeyID ₂	N _T

9.5.9 Final Interrogator processing

The Interrogator receives MAM2 response.

If the value of TStatus is 1₂ then the Interrogator may assume that the Tag is in the state **1A** (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 0₂ then the Interrogator shall abandon the cryptographic protocol.

If the value of TStatus is 1₂, the Interrogator may subsequently invoke secure communication (see [Clause 10](#)) using the key identified by KeyID₂ and the Pre-IV N_T.

10 Communication

10.1 General

This document supports a method for secure communication between the tag and the reader. It permits the secure encapsulation of messages, as envisaged in ISO/IEC 18000-63, as a way of encrypting and authenticating communications between Interrogator and Tag.

In this document, one option for encapsulating communications is provided by the SILC mechanism for authenticated encryption. The full definition of the SILC mechanism, along with information on its design and analysis, is provided in Reference [4]. Implementation shall be as described in [Annex C](#). The SILC mechanism is invoked when the SecureComm field in MAM2 message takes the value 0001₂. Future versions of this document may support additional mechanisms.

SILC is built around the use of a block cipher and is notable for an accompanying proof-of-security. The fundamental approach to SILC was published in 2014 and, since then, the scheme has been both open to public cryptanalysis and widely promoted within the global cryptographic CAESAR initiative. The design of SILC is optimized for implementation efficiency in constrained hardware.

SILC consists of two transformations. The first, denoted SILC-E_k, is for encrypting and generating a message authentication code or authentication token T on an input P. The second, denoted SILC-D_k, is for decrypting a ciphertext C and verifying an accompanying message authentication code or authentication token T. SILC requires the use of a constant that provides separation between different block ciphers and different lengths to the authentication token T. The values of param are provided in Reference [4] and in [Annex C](#) and this document reflects the fact that there is a family of SILC transformations using the notation SILC_{param}.

With this slight change of notation, the SILC transformations[4] can be referred to as:

SILC_{param}-E_k (N, A, P) → (A, C, T): Input a nonce N, auxiliary data A that is to be authenticated but not encrypted, and plaintext P that it is to be authenticated and encrypted. Compute, using the key *k*, the ciphertext C and authentication token T and append these to the auxiliary data A.

SILC_{param}-D_k (N, A, C, T) → (A, P): Input a nonce N, auxiliary data A, ciphertext C, and authentication token T. Compute, using the key *k*, the message P provided the authentication token T is correct and append this to the auxiliary data A.

This subclause describes how these transformations are used in the context of this document.

NOTE 1 Throughout, the Tag only needs to support the encryption direction of the block cipher.

NOTE 2 Throughout, the Interrogator only needs to support the encryption direction of the block cipher.

NOTE 3 During secure communication, a value N is incremented by one using integer addition for each message exchanged. Any upper bound to the number of messages that can be exchanged, any policy on the potential roll-over of N , and the tag behaviour in such situations will be implementation dependent."

Secure communication can only be provided after successful mutual authentication. A Tag response to an encapsulated command may be encapsulated. If the response is encapsulated, then it shall be encapsulated as described in this crypto suite. A Tag shall only encapsulate the response to an encapsulated command.

10.2 Message and response formatting

The air interface specification provides definitions of the commands and responses that can be sent over the air. In the case of an *encapsulating* command, the air interface standard specifies the data fields and permissible payloads for the encapsulating command.

The following subclauses of this document describe how the payloads for the commands and responses should be processed using SILC to provide an authenticated encryption of that payload.

If a secure communication session is required, the session shall be launched by the Interrogator after a successful mutual authentication.

All secure sessions are initialized by a choice of key and a value to the nonce N .

1. The key to be used is chosen by the Interrogator and indicated using KeyID_2 . The variable KeyID_2 may be the same as KeyID used for the authentication session or it may be different.

NOTE Determining $\text{Key}.\text{KeyID}_2$ and managing the deployed choices for KeyID and KeyID_2 are matters of security architecture and key management that fall outside the scope of this document.

2. The initial value to the nonce N is derived from information exchanged during the mutual authentication process. After successful mutual authentication the value N is available to both the Interrogator and Tag.

10.3 Transforming a payload prior to encapsulation

10.3.1 General

This subclause specifies how to authenticate and/or encrypt a payload P that is to be encapsulated. This is achieved via a transformation SEC . The transformation SEC is best viewed as a wrapper around $\text{SILC}_{\text{param-}E_k}$ that provides the interface between the over-the-air command and $\text{SILC}_{\text{param-}E_k}$.

There are four inputs to SEC in addition to the payload P that is to be encapsulated.

1. A b -bit key $\text{Key}.\text{KeyID}_2$ that is passed on to the instantiation of $\text{SILC}_{\text{param-}E_k}$ where it is referred to as k .
2. A nonce N that is passed to the instantiation of $\text{SILC}_{\text{param-}E_k}$.
3. The 8-bit value of param which is used in the $\text{SILC}_{\text{param-}E_k}$ computation.

For each variant of SPECK- b/k and each permitted size of authentication tag T , an input param is defined to take a specific value. These values are specified in the description of SILC v3 and [Table 18](#).

Table 18 — Values of param for different variants of SPECK- b/k and different lengths of authentication tag T

b/k	64/96	64/128	96/96	128/128	128/256
$ T = 32$	$B0_h$	$B1_h$	$B2_h$	$B3_h$	$B4_h$
$ T = 48$	$B5_h$	$B6_h$	$B7_h$	$B8_h$	$B9_h$
$ T = 64$	BA_h	BB_h	BC_h	BD_h	BE_h

4. A one-bit parameter Enc that indicates whether the payload P is to be authenticated ($\text{Enc} = 0_2$) or encrypted and authenticated ($\text{Enc} = 1_2$).

SEC may be used by the Interrogator, *e.g.* to protect an over-the-air command. SEC may also be used by the Tag to protect a response.

SEC shall only be used after mutual authentication has been established and it guarantees the confidentiality and/or authenticity of the payload P being encapsulated.

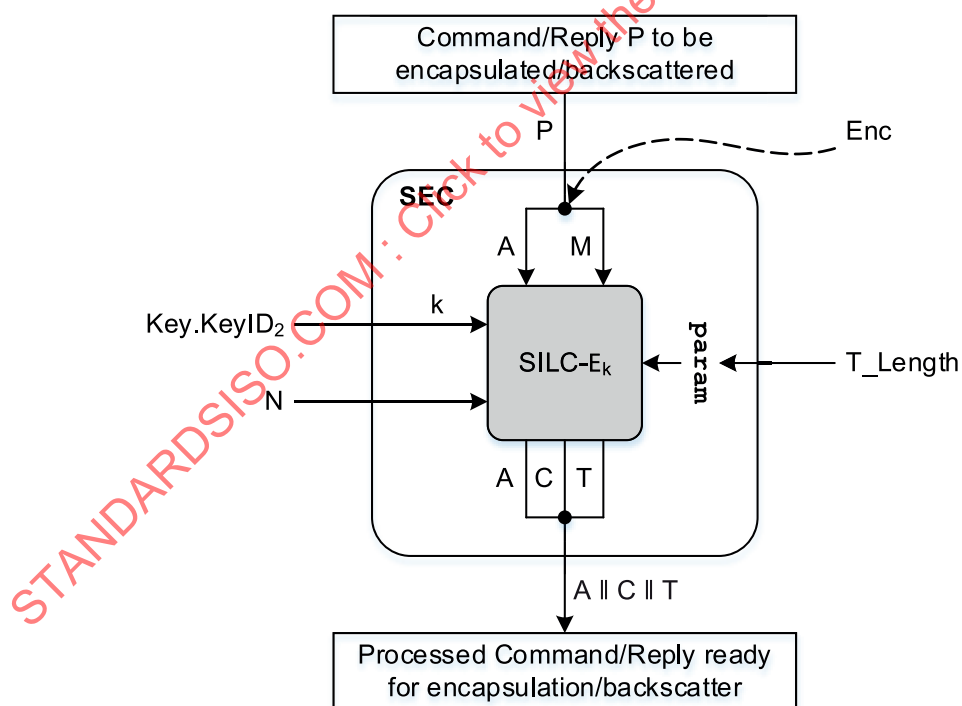
SEC takes Key.KeyID_2 , N , param , Enc , and P as five inputs and interfaces to the SILC transformations in the following way, where C is the encryption of P :

$$\text{SEC}(\text{Key.KeyID}_2, N, \text{param}, 0, P) = \text{SILC}_{\text{param-E}_{\text{Key.KeyID}_2}}(N, P, \emptyset) = P \parallel \emptyset \parallel T.$$

$$\text{SEC}(\text{Key.KeyID}_2, N, \text{param}, 1, P) = \text{SILC}_{\text{param-E}_{\text{Key.KeyID}_2}}(N, \emptyset, P) = \emptyset \parallel C \parallel T.$$

NOTE In the first case, the encapsulated command is authenticated. In the second case, the encapsulated command is encrypted and authenticated.

This process is illustrated in [Figure 5](#).

**Figure 5 — Interface between SEC and $\text{SILC}_{\text{param-E}_k}$**

10.3.2 Encapsulating an Interrogator command

Optionally, an Interrogator may wish to cryptographically protect an over-the-air command. To send a command as an encapsulated payload P , the Interrogator shall perform the following steps:

1. The Interrogator shall verify that T_{Status} is 12. If not, the Interrogator shall abandon the encapsulation process.

SEC shall only be used after mutual authentication has been established.

2. The Interrogator shall identify the key $Key.KeyID_2$ to be used.
3. The Interrogator shall construct the initial value of the $(b-16)$ -bit string N as $N = N_T \parallel T_{Challenge}$.

NOTE $T_{Challenge}$ is generated by the Tag and returned to the Interrogator in the MAM1 response while N_T is generated by the Tag and returned to the Interrogator in the MAM2 response.

4. The Interrogator shall choose the length T_Length for the authentication token T .
5. The Interrogator shall specify if the payload is authenticated or both encrypted and authenticated.

If authentication without encryption is required, then the Interrogator shall set $Enc = 0$. If both encryption and authentication are required, then the Interrogator shall set $Enc = 1$.

NOTE If the encapsulating over-the-air command only supports authentication then $Enc = 0$.

6. The Interrogator shall specify if the payload parameters should be protected.

If parameter protection is required then the Interrogator shall set $Protect = 1$. If parameter protection is not required then the Interrogator shall set $Protect = 0$.

7. The Interrogator shall specify whether the Tag response is to be in the clear, authenticated, or encrypted and authenticated. These different options are represented by the following values to the Response field in Table 19:

0 _x : Clear	1 _x : Authenticated only	2 _x : Encrypted and authenticated	3 _x – F _x : RFU
------------------------	-------------------------------------	--	---------------------------------------

8. The Interrogator shall construct a string X where $|X| = 0$ or $|X| = 8$.

If $Protect = 0$ then $X = \emptyset$. If $Protect = 1$ then $X = Response \parallel Enc \parallel Protect \parallel 00_2$.

9. The Interrogator shall compute SEC ($Key.KeyID_2$, N , $param$, Enc , $X \parallel P$). The output is denoted $Q \parallel T$.

NOTE Depending on the value of Enc , Q will either be equal to $X \parallel P$ or equal to the encryption of $X \parallel P$. In both cases, an authentication token T will be included.

10. The Interrogator shall encapsulate the payload defined in Table 19 and send this to the Tag using an encapsulating over-the-air command.

11. After a successful invocation of SEC, the value of N shall be incremented by 1 using integer addition.

Table 19 — Secured payload for transport by an encapsulating command

Field	Payload						Processed Command/ Reply
	KeyID	Param	Response	Enc	Protect	RFU	
Length (bits)	8	8	4	1	1	2	variable
Description	KeyID ₂	param	variable	variable	variable	00 ₂	$Q \parallel T$

10.3.3 Cryptographically protecting a Tag reply

Depending on the value of Response, a Tag may be required to cryptographically protect the reply to an encapsulated over-the-air command. To protect the reply to an encapsulated command, the Tag shall perform the following steps:

1. If the Tag is not in the state IA then the Tag shall return a “Cryptographic Suite Error” and abandon the communication session.

NOTE 1 A Tag can only react to an encapsulated command after mutual authentication has been established.

2. The Tag shall process the encapsulating command as described in 10.4.1. Unless an error has been encountered, this will reveal the encapsulated command and, optionally, the values of Response, Enc and Protect.

NOTE 2 The value of the Nonce N is always incremented after an invocation of SEC or CES.

NOTE 3 The values of Response, Enc and Protect are carried as headers to the encapsulated payload (see Table 19). When carried as headers, these values are not cryptographically protected. Optionally, Response, Enc and Protect can be included as part of the cryptographic computation invoked by SEC. In this latter case, the values of Response, Enc and Protect are cryptographically protected.

3. If Protect = 1₂ the Interrogator has used parameter protection. Provided the encapsulated command was processed without error (10.4.1), the values of Response, Enc and Protect in Table 19 shall be recovered from the output of CES in 10.4.1.

If Protect = 0₂ the Interrogator has not used parameter protection. The values of Response, Enc and Protect shall be recovered from the payload in Table 19.

4. If Response = 0000₂, the Tag shall respond without encapsulation.
5. Otherwise, the Tag shall execute the encapsulated command and construct the reply R.

NOTE 4 An encapsulated reply can only be formed by the tag on receipt of a correctly-received encapsulated command.

6. If Response = 1_x or 2_x, the Tag shall compute SEC (Key.KeyID₂, N, param, Enc, R) with the reply R. The value of Enc will be 0 if Response = 1_x and Enc = 1 if Response = 2_x. The output is denoted Q || T.

NOTE 5 Depending on the value of Enc, Q will either be equal to R or equal to the encryption of R. In both cases, an authentication tag T will be included.

7. The Tag shall return Q || T to the Interrogator instead of the unprotected reply R.
8. After a successful invocation of SEC, the value of N shall be incremented by 1 using integer addition.

10.4 Processing an encapsulated or cryptographically-protected reply

10.4.1 General

This subclause specifies how to process an encapsulated, *i.e.* cryptographically protected, payload P. This is achieved via a transformation CES. CES can be viewed as a wrapper around SILC_{param-D_k} that provides the interface between the over-the-air command and SILC_{param-D_k}.

NOTE With the same Key.KeyID₂, nonce N, param, and value to Enc the transformation CES can be viewed as providing the reverse functionality of SEC.

CES has four inputs in addition to Q || T which is either an encapsulated command or a protected reply.

1. A *b*-bit key Key.KeyID₂ that is passed on to the instantiation of SILC_{param-D_k} where it is referred to as *k*.

2. A nonce N that is passed on to the instantiation of $SILC_{param-D_k}$.
3. The 8-bit $param$ used in the $SILC_{param-D_k}$ computation. The values of $param$ are given in Table 18.
4. A one-bit parameter Enc indicating whether authentication or both encryption and authentication were used. In both cases, an authentication tag T is verified.

CES may be used by (a) the Tag for processing an encapsulated air interface command or (b) the Interrogator for processing a cryptographically-protected response. This process is illustrated in Figure 6.

CES takes $Key.KeyID_2$, N , $param$, Enc , and $Q \parallel T$ as five inputs and interfaces to $SILC_{param-D_k}$ in the following way, where P is the decryption of Q :

$$CES(Key.KeyID_2, N, param, 0, Q \parallel T) = SILC_{param-D_{Key.KeyID_2}}(N, Q, \emptyset, T) = Q \text{ or } AUTH_ERROR(\perp).$$

$$CES(Key.KeyID_2, N, param, 1, Q \parallel T) = SILC_{param-D_{Key.KeyID_2}}(N, \emptyset, Q, T) = P \text{ or } AUTH_ERROR(\perp).$$

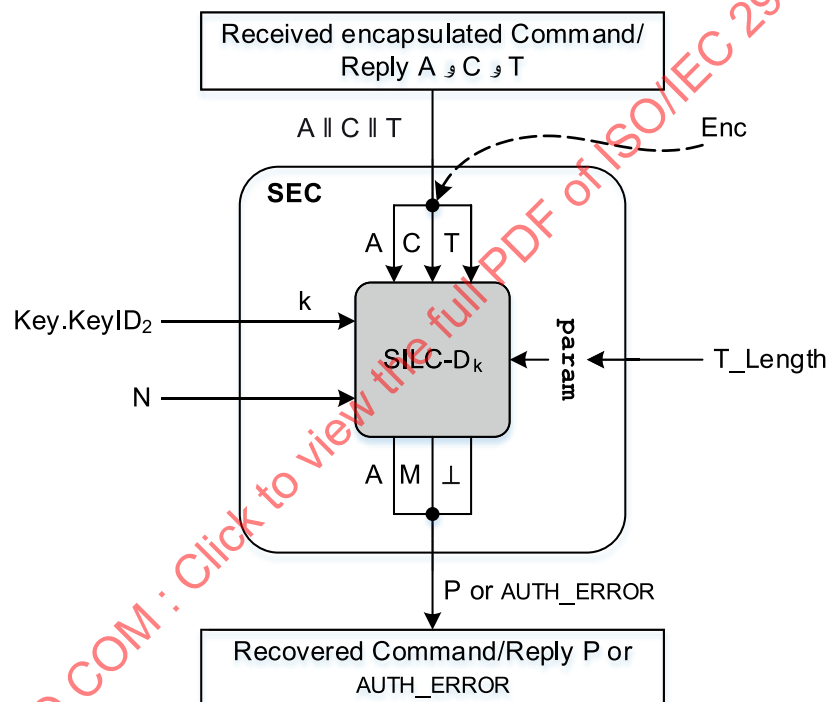


Figure 6 — Interface between CES and $SILC_{param-D_k}$ where an authentication error is denoted $\perp$

10.4.2 Recovering an encapsulated Interrogator command

On receiving an encapsulating command with payload $Q \parallel T$, the Tag shall perform the following steps to process the payload $Q \parallel T$.

1. If the Tag is not in the state IA then the encapsulating command shall be ignored and the Tag shall return a "Cryptographic Suite Error" and abandon the communication session.

CES shall only be used after mutual authentication has been established.

2. The Tag shall check whether $Key.KeyID_2$ is authorized for use with secure communication. If not, the Tag shall return a "Not Supported" error.
3. The Tag shall check if $param$ is supported by the Tag. If not, the Tag shall return a "Not Supported" error condition.

4. The Tag shall check if the value of Enc is supported by the Tag. If not, the Tag shall return a "Not Supported" error condition.
5. The Tag shall check if the RFU is "00₂". If the value of RFU is different, the Tag shall return a "Not Supported" error.
6. The Tag shall construct the initial value of N as $N = N_T \parallel T_{Challenge}$.
7. Assuming that the encapsulating command is successfully parsed by the Tag, the Tag shall recover $Q \parallel T$ from the encapsulating command and compute CES (Key.KeyID₂, N, param, Enc, $Q \parallel T$).
8. If the output from CES (Key.KeyID₂, N, T_Length, Enc, $Q \parallel T$) is AUTH_ERROR, the Tag shall return a "Cryptographic Suite Error" and abandon the communication session.
9. If the output from CES (Key.KeyID₂, N, T_Length, Enc, $Q \parallel T$) is not AUTH_ERROR, the Tag shall consider the output to be the intended encapsulated command.
10. After a successful invocation of CES, the value of N shall be incremented by 1 using integer addition.

10.4.3 Recovering a cryptographically-protected Tag response

To recover the original response from a cryptographically-protected Tag response, the Interrogator shall perform the following steps:

1. The Interrogator shall verify that the Tag response was received in response to an encapsulated command issued by the Interrogator. If not, the Interrogator shall abandon the secure communication process.

NOTE SEC can only be used by a Tag to cryptographically protect the response to an encapsulated Interrogator command.

2. The Interrogator shall recover the cryptographically-protected reply $Q \parallel T$ from the over-the-air Tag response.
3. The Interrogator shall compute CES (Key.KeyID₂, N, param, Enc, $Q \parallel T$).

If the output is AUTH_ERROR, the Interrogator shall abandon the communication session. Otherwise the output is the originally constructed Tag reply R.

4. After a successful invocation of CES, the value of N shall be incremented by 1 using integer addition.

11 Key table and key update

Since this cipher suite provides a method of secure encapsulation, key update can be naturally and transparently supported via command encapsulation.

Optionally, the Tag manufacturer may support a Key Table. If a Key Table is supported then it may store up to 256 keys (Key.0 through to Key.255) and it may take the form shown in [Table 20](#). Not all key values need to be specified. However Key.j shall not be specified when there remain unspecified Key.i with $i < j$.

The field RFU may be used to define properties that should be associated with a given Key.i. Any use of such values is manufacturer dependent.

Table 20 — Optional Key Table for 29167-21

Field	KeyID	Key	RFU
Length (bits)	8	<i>b</i>	8
	00000000 ₂	Key.0	00 _h
	00000001 ₂	Key.1	00 _h
	<i>etc.</i>	<i>etc.</i>	<i>etc.</i>

Optionally, the Key Table may be updated using a key update command. If so, the payload to the key update command may have the form given in [Table 21](#).

Table 21 — Optional payload format for a key update

Field	Payload		
	KeyID	Key	RFU
Length (bits)	8	<i>b</i>	8
Description	KeyID	Key.KeyID	00 _h

If a Tag is unable to act on the key update command, the Tag shall return a "Not Supported" error. The success or otherwise of a key update command may be indicated using a single bit in a reply while any response field to a key update command may be empty.

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.

This annex contains a listing of the errors that can result from the operation of this cryptographic suite. These errors shall be translated into an error code for the air interface in accordance with [Annex E](#).

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 (normative)

Description of SPECK and SILC v3

C.1 SPECK

SPECK- b/k is a block cipher that is parameterized to use a range of block and key sizes denoted by b and k . SPECK- b/k uses the repeated application of a round function and SPECK- b/k is defined by its round function and associated key schedule. With a block size b , the basic operational unit is an n -bit word where $b=2n$. The key length k will be an integral multiple of n with the multiple depending on the variant used.

For $k \in \text{GF}(2)^n$, the key-dependent SPECK- b/k round function is the map $R_k : \text{GF}(2)^n \times \text{GF}(2)^n \rightarrow \text{GF}(2)^n \times \text{GF}(2)^n$ defined by:

$$R_k(x, y) = (S^{-\alpha}x + y) \oplus k, S^{\beta}y \oplus (S^{-\alpha}x + y) \oplus k$$

where k is the round key. Here “+” denotes integer addition modulo 2^n , “ $\oplus$ ” denotes bitwise XOR, and S^tx denotes the leftward rotate of the n -bit word x by $(t \text{ modulo } n)$ bit positions. The SPECK- b/k key schedule takes a k -bit key and generates T key words $k_0 \dots k_{T-1}$, where T is the number of rounds. A pseudo-code description of encryption using SPECK- b/k is given below for the values of b/k supported by this document. More details can be found in Reference [3].

```

----- definitions -----
n = word size (32, 48, or 64)                                (Note: block size b = 2n)

m = number of key words = 3 or 4 if n = 32,
                        = 2 if n = 48,
                        = 2 or 4 if n = 64

T = number of rounds    = 26 or 27 if n = 32, m = 3 or 4
                        = 28 if n = 48, m = 2
                        = 32 or 34 if n = 64, m = 2 or 4

( $\alpha, \beta$ ) = (8, 3)
x, y = plaintext words                                       (Note: plaintext input = x || y)
 $\lambda[m-2] \dots \lambda[0], k[0]$  = key words
----- key expansion -----
for i = 0..T-2
     $\lambda[i+m-1] \leftarrow (k[i] + S^{-\alpha}\lambda[i]) \oplus i$ 
     $k[i+1] \leftarrow S^{\beta}(k[i]) \oplus \lambda[i+m-1]$ 
end for
----- encryption -----
for i = 0..T-1
     $x \leftarrow (S^{-\alpha}x + y) \oplus k[i]$ 
     $y \leftarrow S^{\beta}y \oplus x$ 
end for                                     (Note: ciphertext output = x || y)
----- end -----

```

C.2 SILC v3

Some additional parameters and definitions are required for a description of the authenticated-encryption mode SILC v3.

For each variant of SPECK-b/k, the size of the nonce N is fixed according to the block size b.

b/k	64/96	64/128	96/96	128/128	128/256
N	48	48	72	96	96

For each variant of SPECK-b/k and each permitted size of authentication tag T, an input **param** is defined to take a specific value. These values are specified in the description of SILC v3 and repeated here.

b/k	64/96	64/128	96/96	128/128	128/256
T = 32	B0 _h	B1 _h	B2 _h	B3 _h	B4 _h
T = 48	B5 _h	B6 _h	B7 _h	B8 _h	B9 _h
T = 64	BA _h	BB _h	BC _h	BD _h	BE _h

Some addition definitions for the following material in [Annex C](#) are provided here.

len_b(A) The b -bit binary representation of the integer length of the bit string A.

Example 1: **len**₈(01111₂) = 00000101₂.

Example 2: **len**₁₆(FFFF_h) = 00000000000010000₂.

Example 3: **len**₁₆(∅) = 00000000000000000₂.

zpp_b(A) Bit-wise padding to the left of the n -bit string A with 0₂ giving a b -bit result for $n < b$.

Example 1: **zpp**₈(1111₂) = 00001111₂.

Example 2: **zpp**₁₆(FFFF_h) = FFFF_h.

zap_b(A) Bit-wise padding to the right of the n -bit string A with 0₂ giving a b -bit result for $n < b$.

Example 1: **zap**₈(1111₂) = 11110000₂.

Example 2: **zap**₁₆(FFFF_h) = FFFF_h.

g(A) A byte-wise operation on the b -bit input A, represented as bytes A[1] || ... || A[b/8], and defined as follows.

For $b=64$: **g**(X) = **g**(A[1] || ... || A[7] || A[8]) = A[2] || ... || A[8] || (A[1]⊕A[2])

For $b=96$: **g**(A) = **g**(A[1] || ... || A[11] || A[12]) = A[2] || ... || A[12] || (A[1]⊕A[2])

For $b=128$: **g**(A) = **g**(A[1] || ... || A[15] || A[16]) = A[2] || ... || A[16] || (A[1]⊕A[2])

SILC-E _k (N, A, M)	SILC-D _k (N, A, C, T)
1. Set V = HASH _K (N,A) 2. Set C = ENC _K (V,M) 3. Set T = PRF _K (V,C) 4. return (C T)	1. Set V = HASH _K (N,A) 2. Set T* = PRF _K (V,C) 3. if T* ≠ T then return AUTH_ERROR 4. Set M = DEC _K (V,C) 5. return M

Algorithm HASH _K (N, A)
1. Set S _H [0] = SPECK- b/k -ENC (K, zpp _{b} (param N)) 2. if A = ∅ then Set V = g (S _H [0]) 3. else Write A as w b -bit blocks A[1] ... A[w] for $i = 1$ to $w - 1$ do Set S _H [i] = SPECK- b/k -ENC (K, S _H [$i-1$] ⊕ A[i]) Set S _H [w] = SPECK- b/k -ENC (K, S _H [$w-1$] ⊕ zap _{b} (A[w])) Set V = g (S _H [w] ⊕ len _{b} (A)) 4. return V

Algorithm ENC_K (V, M)

```

1. if M = ∅ then
    Set C = ∅
2. else
    Write M as d b-bit blocks M[1] || ... || M[d]
    Set SE[1] = SPECK-b/k-ENC (K, V)
    for i = 1 to d-1 do
        Set C[i] = SE[i] ⊕ M[i]
        Set SE[i+1] = SPECK-b/k-ENC (K, fix1(C[i]))
    Set C[d] = msb|M[d]| (SE[d]) ⊕ M[d]
    Set C = C[1] || ... || C[d]
3. return C

```

Algorithm DEC_K (V, C)

```

1. if C = ∅ then
    Set M = ∅
2. else
    Write C as d b-bit blocks C[1] || ... || C[d]
    Set SD[1] = SPECK-b/k-ENC (K, V)
    for i = 1 to d-1 do
        Set M[i] = SD[i] ⊕ C[i]
        Set SD[i+1] = SPECK-b/k-ENC (K, fix1(C[i]))
    Set M[d] = msb|C[d]| (SD[d]) ⊕ C[d]
    Set M = M[1] || ... || M[d]
3. return M

```

Algorithm PRF_K (V, C)

```

1. Set SP[0] = SPECK-b/k-ENC (K, g(V))
2. if |C| = 0 then
    Set U = g(SP[0])
    Set T = msbτ (SPECK-b/k-ENC (K, U))
3. else
    Write C as d b-bit blocks C[1] || ... || C[d]
    for i = 1 to d-1 do
        Set SP[i] = SPECK-b/k-ENC (K, SP[i-1] ⊕ C[i])
    Set SP[d] = SPECK-b/k-ENC (K, SP[d-1] ⊕ zapb (C[m]))
    Set U = g(SP[d] ⊕ lenb (C))
    Set T = msb|T| (SPECK-b/k-ENC (K, U))
4. return T

```

Annex D (informative)

Test vectors

[Table D.1](#) provides test vectors for encrypting plaintext messages using the SPECK block cipher with b -bit blocks and k -bit keys.

Table D.1 — Test vectors for SPECK-b/k

Variant b/k	Plaintext	Key	Ciphertext
64/96	6F722067 _h 6E696C63 _h	13121110 _h 0B0A0908 _h 03020100 _h	863376EF _h 7295059B _h
64/128	656B696C _h 20646E75 _h	1B1A1918 _h 13121110 _h 0B0A0908 _h 03020100 _h	DA0A71CB _h D5FAA975 _h
96/96	2072616C _h 6C697020 _h 65687420 _h	0D0C0B0A _h 09080504 _h 03020100 _h	4701A708 _h 73FA91E3 _h D885E712 _h
128/128	63736564 _h 20737265 _h 6C6C6576 _h 61727420 _h	0F0E0D0C _h 0B0A0908 _h 07060504 _h 03020100 _h	90AA5135 _h BC6624EB _h FE3CBBDf _h 66914001 _h
128/256	74206E69 _h 206D6F6F _h 6D697320 _h 61207369 _h	1F1E1D1C _h 1B1A1918 _h 17161514 _h 13121110 _h 0F0E0D0C _h 0B0A0908 _h 07060504 _h 03020100 _h	BBD10D45 _h D675C5F9 _h D0EC6494 _h 05B3AA29 _h

[Table D.2](#) provides test vectors for Tag authentication using SPECK with b -bit blocks and k -bit keys. The value of the key used for each example in [Table D.2](#) corresponds to the key used for each b/k value in [Table D.1](#). The value of KeyID is set to 0_h. See the text for the bit length of parameters; values are written here by filling 32-bit words starting from the right-most bit.

Table D.2 — Test vectors for Tag authentication using SPECK-b/k

Variant b/k	IChallenge-b/k	TAM1 Message	TRnd-b/k	TResponse
64/96	2F7 _h 220676E6 _h	000002F7 _h 220676E6 _h	ABCDE _h	EBAA6EF3 _h 3B790E37 _h
64/128	2F7 _h 220676E6 _h	002002F7 _h 220676E6 _h	ABCDE _h	D457AC8F _h B72682B4 _h
96/96	6F7220 _h 676E696C _h	010 _h 006F7220 _h 676E696C _h	321ABCDE _h	1262579b _h 203A135D _h CE0D62C2 _h
128/128	6F72 _h 20676E69 _h 6C636C6C _h	0 _h 24006F72 _h 20676E69 _h 6C636C6C _h	321ABCDE _h	4DE73016 _h 78A507E1 _h 7A372149 _h B3CA54B3 _h
128/256	6F72 _h 20676E69 _h 6C636C6C _h	0 _h 28006F72 _h 20676E69 _h 6C636C6C _h	321ABCDE _h	4A2FA6A7 _h DE46B48E _h 67090611 _h 1628C941 _h

[Table D.3](#) provides test vectors for Interrogator authentication using SPECK with b -bit blocks and k -bit keys. The value of the key used for each example in [Table D.3](#) corresponds to the key used for each b/k value in [Table D.1](#). The value of KeyID is set to 0_h. See the text for the bit length of parameters; values are written here by filling 32-bit words starting from the right-most bit.

Table D.3 — Test vectors for Interrogator authentication using SPECK-b/k

Variant <i>b/k</i>	IAM1 Message	TChallenge- <i>b/k</i>	IRnd- <i>b/k</i>	IAM2 Message
64/96	40000 _h	2F7 _h 220676E6 _h	ABCDE _h	50 _h 99B9D02C _h 060F6268 _h
64/128	40400 _h	2F7 _h 220676E6 _h	ABCDE _h	50 _h 875C87C4 _h BCFE0A84 _h
96/96	41000 _h	6F7220 _h 676E696C _h	321ABCDE _h	50 _h 47FA85B4 _h 7E838977 _h F67C3D04 _h
128/128	42000 _h	6F72 _h 20676E69 _h 6C636C6C _h	321ABCDE _h	50 _h 6A7B3E27 _h AE319BD6 _h 97A8C2DD _h AF002949 _h
128/256	42800 _h	6F72 _h 20676E69 _h 6C636C6C _h	321ABCDE _h	50 _h 3CE936AF _h 20F5DBC5 _h 72257DDE _h E0B049AE _h

Tables D.4 to D.13 provide test vectors for Mutual authentication using SPECK with *b*-bit blocks and *k*-bit keys. Tables D.14 to D.23 provide test vectors for authenticated command encapsulation using SPECK with *b*-bit blocks and *k*-bit keys. The value of the key used for each example corresponds to the key used for each *b/k* value in Table D.1. The value of KeyID is set to 0_h. See the text for the bit length of parameters; values are written here by filling 32-bit words starting from the right-most bit.

Table D.4 — Test vectors for Mutual authentication using SPECK-64/96 and PS= 002

SPECK-64/96	Bits	Value
Key	96	13121110 _h 0B0A0908 _h 03020100 _h
IChallenge-64/96	42	2F7 _h 220676E6 _h
MAM1 Message	64	200002F7 _h 220676E6 _h
TChallenge-64/96	42	2F7 _h 220676E6 _h
Input to ENC	64	6F7222F7 _h 220676E6 _h
S	64	50D07AF7 _h 535618D1 _h
TResponse	86	0676E6 _h 50D07AF7 _h 535618D1 _h
Input to DEC for MAM2	64	59DB9AF7 _h 220676E6 _h
IResponse	64	883D72B6 _h 7B67A756 _h
MAM2 Message (SecureComm = 0)	76	900 _h 883D72B6 _h 7B67A756 _h
MAM2 Message (SecureComm = 1)	76	901 _h 883D72B6 _h 7B67A756 _h

Table D.5 — Test vectors for Mutual authentication using SPECK-64/128 with PS=002

SPECK-64/128	Bits	Value
Key	128	1B1A1918 _h 13121110 _h 0B0A0908 _h 03020100 _h
IChallenge-64/96	42	2F7 _h 220676E6 _h
MAM1 Message	64	201002F7 _h 220676E6 _h
TChallenge-64/96	42	2F7 _h 220676E6 _h
Input to ENC	64	6F7222F7 _h 220676E6 _h
S	64	EA391A0A _h 23CFF898 _h
TResponse	86	0676E6 _h EA391A0A _h 23CFF898 _h
Input to DEC for MAM2	64	59DB9AF7 _h 220676E6 _h
IResponse	64	4E03BE13 _h D3D19E52 _h
MAM2 Message (SecureComm = 0)	76	900 _h 4E03BE13 _h D3D19E52 _h
MAM2 Message (SecureComm = 1)	76	901 _h 4E03BE13 _h D3D19E52 _h

Table D.6 — Test vectors for Mutual authentication using SPECK-96/96 with PS=00₂

SPECK-96/96	Bits	Value
Key	96	0D0C0B0A _h 09080504 _h 03020100 _h
ICChallenge-64/96	56	6F7220 _h 676E696C _h
MAM1 Message	76	810 _h 006F7220 _h 676E696C _h
TChallenge-64/96	56	6F7220 _h 676E696C _h
Input to ENC	96	FD676E69 _h 6C6F7220 _h 676E696C _h
S	96	E430A59E _h 64A2A4AB _h 5574F3CB _h
TResponse	120	6F7220 _h E430A59E _h 64A2A4AB _h 5574F3CB _h
Input to DEC for MAM2	96	FD676E69 _h 6C6F7220 _h 676E696C _h
IResponse	96	098601D6 _h 8602BCCC _h BE7EE9B1 _h
MAM2 Message (SecureComm = 0)	108	900 _h 098601D6 _h 8602BCCC _h BE7EE9B1 _h
MAM2 Message (SecureComm = 1)	108	901 _h 098601D6 _h 8602BCCC _h BE7EE9B1 _h

Table D.7 — Test vectors for Mutual authentication using SPECK-128/128 with PS=00₂

SPECK-128/128	Bits	Value
Key	128	0F0E0D0C _h 0B0A0908 _h 07060504 _h 03020100 _h
ICChallenge-128/128	80	6F72 _h 20676E69 _h 6C636C6C _h
MAM1 Message	100	8 _h 24006F72 _h 20676E69 _h 6C636C6C _h
TChallenge-128/128	80	6F72 _h 20676E69 _h 6C636C6C _h
Input to ENC	128	FFFD6F72 _h 20676F72 _h 20676E69 _h 6C636C6C _h
S	128	B77119B3 _h 621328E8 _h 616bA064 _h F01FE70C _h
TResponse	176	6E69 _h 6C636C6C _h B77119B3 _h 621328E8 _h 616bA064 _h F01FE70C _h
Input to DEC for MAM2	128	FFFD6F72 _h 20676F72 _h 20676E69 _h 6C636C6C _h
IResponse	128	BAFD52B9 _h 9E1334FF _h F26A7C2A _h CA1E08E7 _h
MAM2 Message (SecureComm = 0)	140	900 _h BAFD52B9 _h 9E1334FF _h F26A7C2A _h CA1E08E7 _h
MAM2 Message (SecureComm = 1)	140	901 _h BAFD52B9 _h 9E1334FF _h F26A7C2A _h CA1E08E7 _h

Table D.8 — Test vectors for Mutual authentication using SPECK-128/256 with PS=00₂

SPECK-128/256	Bits	Value
Key	256	1F1E1D1C _h 1B1A1918 _h 17161514 _h 13121110 _h 0F0E0D0C _h 0B0A0908 _h 07060504 _h 03020100 _h
ICChallenge-128/256	80	6F72 _h 20676E69 _h 6C636C6C _h
MAM1 Message	100	8 _h 28006F72 _h 20676E69 _h 6C636C6C _h
TChallenge-128/256	80	6F72 _h 20676E69 _h 6C636C6C _h
Input to ENC	128	FFFD6F72 _h 20676F72 _h 20676E69 _h 6C636C6C _h
S	128	B433F966 _h A69C8BE3 _h 64A4375A _h A74f4065 _h
TResponse	176	6E69 _h 6C636C6C _h B433F966 _h A69C8BE3 _h 64A4375A _h A74F4065 _h
Input to DEC for MAM2	128	FFFD6F72 _h 20676F72 _h 20676E69 _h 6C636C6C _h

Table D.8 (continued)

SPECK-128/256	Bits	Value
IResponse	128	DAB64659 _h C89CCDC6 _h 98021898 _h EC68D9E2 _h
MAM2 Message (SecureComm = 0)	140	900 _h DAB64659 _h C89CCDC6 _h 98021898 _h EC68D9E2 _h
MAM2 Message (SecureComm = 1)	140	901 _h DAB64659 _h C89CCDC6 _h 98021898 _h EC68D9E2 _h

Table D.9 — Test vectors for Mutual authentication using SPECK-64/96 with PS=01₂

SPECK-64/96	Bits	Value
Key	96	13121110 _h 0B0A0908 _h 03020100 _h
IChallenge-64/96	30	220676E6 _h
MAM1 Message	50	20000 _h 620676E6 _h
TChallenge-64/96	30	220676E6 _h
Input to ENC	64	18819DB9 _h A20676E6 _h
S	64	6019E12A _h 37B18C74 _h
TResponse	64	6019E12A _h 37B18C74 _h
Input to DEC for MAM2	—	—
IResponse	30	220676E6 _h
MAM2 Message (SecureComm = 0)	42	240 _h 220676E6 _h
MAM2 Message (SecureComm = 1)	42	240 _h 620676E6 _h

Table D.10 — Test vectors for Mutual authentication using SPECK-64/128 with PS=01₂

SPECK-64/128	Bits	Value
Key	128	1B1A1918 _h 13121110 _h 0B0A0908 _h 03020100 _h
IChallenge-64/128	30	220676E6 _h
MAM1 Message	50	20100 _h 620676E6 _h
TChallenge-64/128	30	220676E6 _h
Input to ENC	64	18819DB9 _h A20676E6 _h
S	64	B8FFDF48 _h 05A9F7F4 _h
TResponse	64	B8FFDF48 _h 05A9F7F4 _h
Input to DEC for MAM2	—	—
IResponse	30	220676E6 _h
MAM2 Message (SecureComm = 0)	42	240 _h 220676E6 _h
MAM2 Message (SecureComm = 1)	42	240 _h 620676E6 _h

Table D.11 — Test vectors for Mutual authentication using SPECK-96/96 with PS=01₂

SPECK-96/96	Bits	Value
Key	96	0D0C0B0A _h 09080504 _h 03020100 _h
IChallenge-96/96	46	3220 _h 676E696C _h
MAM1 Message	62	20400620 _h 676E696C _h
TChallenge-96/96	46	3220 _h 676E696C _h
Input to ENC	96	DC8819DB _h 9A5B3220 _h 676E696C _h
S	96	5ACE71E2 _h 5B151445 _h B1E5BA1B _h
TResponse	96	5ACE71E2 _h 5B151445 _h B1E5BA1B _h
Input to DEC for MAM2	—	—

Table D.11 (continued)

SPECK-96/96	Bits	Value
IResponse	46	3220 _h 676E696C _h
MAM2 Message (SecureComm = 0)	58	2403220 _h 676E696C _h
MAM2 Message (SecureComm = 1)	58	2407220 _h 676E696C _h

Table D.12 — Test vectors for Mutual authentication using SPECK-128/128 with PS=01₂

SPECK-128/128	Bits	Value
Key	128	0F0E0D0C _h 0B0A0908 _h 07060504 _h 03020100 _h
IChallenge-128/128	60	0676E69 _h 6C636C6C _h
MAM1 Message	80	8240 _h 10676E69 _h 6C636C6C _h
TChallenge-128/128	60	0676E69 _h 6C636C6C _h
Input to ENC	128	FD0676E6 _h 96C636C6 _h C0676E69 _h 6C636C6C _h
S	128	9F682F58 _h 42357D82 _h 4381FCE6 _h FABADB08 _h
TResponse	128	9F682F58 _h 42357D82 _h 4381FCE6 _h FABADB08 _h
Input to DEC for MAM2	—	—
IResponse	60	0676E69 _h 6C636C6C _h
MAM2 Message (SecureComm = 0)	72	90 _h 00676E69 _h 6C636C6C _h
MAM2 Message (SecureComm = 1)	72	90 _h 10676E69 _h 6C636C6C _h

Table D.13 — Test vectors for Mutual authentication using SPECK-128/256 with PS=01₂

SPECK-128/256	Bits	Value
Key	256	1F1E1D1C _h 1B1A1918 _h 17161514 _h 13121110 _h 0F0E0D0C _h 0B0A0908 _h 07060504 _h 03020100 _h
IChallenge-64/96	60	0676E69 _h 6C636C6C _h
MAM1 Message	80	8280 _h 10676E69 _h 6C636C6C _h
TChallenge-64/96	60	0676E69 _h 6C636C6C _h
Input to ENC	128	FD0676E6 _h 96C636C6 _h C0676E69 _h 6C636C6C _h
S	128	E98081D8 _h 32E85407 _h 921DBF44 _h 429960A6 _h
TResponse	128	E98081D8 _h 32E85407 _h 921DBF44 _h 429960A6 _h
Input to DEC for MAM2	—	—
IResponse	60	0676E69 _h 6C636C6C _h
MAM2 Message (SecureComm = 0)	72	90 _h 00676E69 _h 6C636C6C _h
MAM2 Message (SecureComm = 1)	72	90 _h 10676E69 _h 6C636C6C _h

Table D.14 — Test vectors for authenticated command encapsulation using SPECK-64/96 with PS=00₂

SPECK-64/96 (PS= 00 ₂)	Bits	Value
Key.00	96	13121110 _h 0B0A0908 _h 03020100 _h
TChallenge-64/96	42	2F7 _h 220676E6 _h
MAM2 Message (SecureComm = 1 _h)	76	901 _h 883D72B6 _h 7B67A756 _h
MAM2 Response (KeyID ₂ = 01 _h)	15	406D _h
Nonce N = N _T TChallenge	48	B4F7 _h 220676E6 _h
Key.01	96	03020100 _h 1B1A1918 _h 13121110 _h
READ first four rows of user memory	26	30B0004 _h

Table D.14 (continued)

SPECK-64/96 (PS= 00 ₂)	Bits	Value
SEC (Key.01, B0 _h , 0 _b , 30B0004 _h) <start>	Add 32-bit authentication tag	
Get V <start>		
S _H [0]	64	156253CA _h 551BA0A5 _h
S _H [1]	64	B32C498D _h 91016779 _h
V <end>	64	g(B32c498D _h 91016763 _h) = 2C498D91 _h 0167639F _h
Get C	-	-
Get T <start>		
S _P [0]	64	73AEB17f _h DE10A390 _h
S _P [1]	—	—
U	64	AEB17FDE _h 10A390DD _h
T <end>	32	F78F1D92 _h
SEC (Key.01, B0 _h , 0 _b , 30B0004 _h) <end>	58	30B0004 _h F78F1D92 _h
Secured payload	82	006C0 _h 030B0004 _h F78F1D92 _h

Table D.15 — Test vectors for authenticated and encrypted command encapsulation using SPECK-64/96 with PS=00₂

SPECK-64/96 (PS= 00 ₂)	Bits	Value
Key.00	96	13121110 _h 0B0A0908 _h 03020100 _h
TChallenge-64/96	42	2F7 _h 220676E6 _h
MAM2 Message (SecureComm = 1)	76	901 _h 883D72B6 _h 7B67A756 _h
MAM2 Response (KeyID ₂ = 01 _h)	15	406D _h
Nonce N = N _T Tchallenge	48	B4F7 _h 220676E6 _h
Key.01	96	03020100 _h 1B1A1918 _h 13121110 _h
READ first four rows of user memory	26	30B0004 _h
SEC (Key.01, B0 _h , 0 _b , 30B0004 _h) <start>	Encrypt and add 32-bit authentication tag	
Get V <start>		
S _H [0]	64	156253CA _h 551BA0A5 _h
S _H [1]	—	—
V <end>	64	g(156253CA _h 551BA0A5 _h) = 6253CA55 _h 1BA0A577 _h
Get C <start>		
S _E [1]	64	51C82AAE _h 66C0C68C _h
C <end>	26	24C20AE _h
Get T <start>		
S _P [0]		9780c47e _h 91724240 _h
S _P [1]		6930BEA5 _h 5C4D7DA2 _h
U	64	g(6930BEA5 _h 5C4D7DB8 _h) = 30BEA55C _h 4D7DB859 _h
T <end>	32	4B81178D _h
SEC (Key.01, B0 _h , 0 _b , 30B0004 _h) <end>	58	24C20AE _h 4B81178D _h
Secured payload	82	006C0 _h 224C20AE _h 4B81178D _h