INTERNATIONAL STANDARD

ISO/IEC 29167-17

First edition
2015-06-01

Information technology — Automatic identification and data capture techniques —

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

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

Partie 17: Services de sécurité par suite cryptographique cryptoGPS pour communications d'interface radio

Reference number
ISO/IEC 29167-17:2015(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*, 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 part is under preparation:

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

Introduction

cryptoGPS is a lightweight asymmetric identification scheme that is suitable for RFID Tag authentication. While there are many types of such scheme, the computational costs for the Tag when using cryptoGPS are relatively low. This is particularly the case since cryptoGPS is well-suited to an implementation strategy using what is referred to as "coupons". These are the results given by a modest off-line pre-computation, with coupons being used by the prover at each invocation of the cryptoGPS scheme. The resultant scheme offers very useful performance trade-offs.

This part of ISO/IEC 29167 specifies the security services of the cryptoGPS cryptographic suite that provides Tag authentication.

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 might 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 can be obtained from:

Orange
38-40, rue du General Leclerc
F-92794 Issy Les Moulineaux CEDEX 9

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 17: Crypto suite cryptoGPS security services for air interface communications

1 Scope

This part of ISO/IEC 29167 defines the cryptoGPS cryptographic suite 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 might be referred by ISO committees for air interface standards and application standards.

This part of ISO/IEC 29167 defines a lightweight mechanism using asymmetric techniques and providing a unilateral authentication mechanism whose security is related to the difficulty of taking discrete logarithms on elliptic curves.

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 message and response formatting defined in this part of ISO/IEC 29167 and conform to the relevant part of ISO/IEC 18000.

To conform to this part of ISO/IEC 29167, an Interrogator might

— implement any subset of the parameters for message and response formatting 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 message and response formatting defined in this part of ISO/IEC 29167 for the supported types and conform to the relevant part of ISO/IEC 18000.

To conform to this part of ISO/IEC 29167, a Tag might

— implement any subset of the parameters for message and response formatting 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 9798-5:2010, *Information technology — Security techniques — Entity authentication — Part 5: Mechanisms using zero-knowledge techniques*

ISO/IEC 15946-1, *Information technology — Security techniques — Cryptographic techniques based on elliptic curves — Part 1: General*

ISO/IEC 18000-3, *Information technology — Radio frequency identification for item management — Part 3: Parameters for air interface communications at 13,56 MHz*

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*

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

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
asymmetric cryptographic technique
cryptographic technique that uses two related operations: a public operation defined by a public data item, and a private operation defined by a private data item

Note 1 to entry: The two operations have the property that, given the public operation, it is computationally infeasible to derive the private operation.

[SOURCE: ISO/IEC 9798-5:2009, 2.3]

4.2
asymmetric pair
two related data items where the private data item defines a private operation and the public data item defines a public operation

[SOURCE: ISO/IEC 9798-5:2009, 2.5]

4.3
challenge
procedure parameter used in conjunction with secret parameters to produce a response

[SOURCE: ISO/IEC 9798-5:2009, 2.6]

4.4
claimant
entity whose identity can be authenticated, including the functions and the private data necessary to engage in authentication exchanges on behalf of a principal

[SOURCE: ISO/IEC 9798-5:2009, 2.7]

4.5
claimant parameter
public data item, number or bit string, specific to a given claimant within the domain

[SOURCE: ISO/IEC 9798-5:2009, 2.9]

4.6
commitment
public value used to engage a secret value without revealing it

Note 1 to entry: The commitment is used in a protocol so that a party cannot change a secret value after it has committed to it.

4.7
coupon
pre-computed number which shall be used only once

[SOURCE: ISO/IEC 9798-5:2009, 2.8, modified]

4.8
domain
collection of entities operating under a single security policy

Note 1 to entry: For instance, public key certificates created either by a single certification authority, or by a collection of certification authorities using the same security policy.

[SOURCE: ISO/IEC 9798-5:2009, 2.11]

4.9
domain parameter
public key, or function, agreed and used by all entities within the domain

[SOURCE: ISO/IEC 9798-5:2009, 2.12]

4.10
entity authentication
corroboration that an entity is the one claimed

[SOURCE: ISO/IEC 9798-1:2010, 3.14]

4.11
hash function
function which maps strings of bits to fixed-length strings of bits, satisfying the following two properties:

— it is computationally infeasible to find for a given output, an input which maps to this output;

— it is computationally infeasible to find two different input which map to the same output.

Note 1 to entry: Computational feasibility depends on the specific security requirements and environment.

[SOURCE: ISO/IEC 10118-1:2000, 3.5]

4.12
private key
private data item of an asymmetric pair, that shall be kept secret and should only be used by a claimant in accordance with an appropriate response formula, thereby establishing its identity

[SOURCE: ISO/IEC 9798-5:2009, 2.21]

4.13
procedure parameter
transient public data item used in an instance of an authentication mechanism, e.g. a commitment, challenge, or response

[SOURCE: ISO/IEC 9798-5:2009, 2.22]

4.14
public key
public data item of an asymmetric pair, that can be made public and shall be used by every verifier for establishing the claimant's identity

[SOURCE: ISO/IEC 9798-5:2009, 2.23]

4.15
random number
time variant parameter whose value is unpredictable

[SOURCE: ISO/IEC 9798-1:2010, 3.29]

4.16
response
procedure parameter produced by the claimant, and processed by the verifier for checking the identity of the claimant

[SOURCE: ISO/IEC 9798-5:2009, 2.25]

4.17
secret parameter
number or bit string that does not appear in the public domain and is only used by a claimant

Note 1 to entry: For instance, a private key.

[SOURCE: ISO/IEC 9798-5:2009, 2.26]

4.18
unilateral authentication
entity authentication which provides one entity with assurance of the other's identity but not vice versa

[SOURCE: ISO/IEC 9798-1:2010, 3.39]

4.19
verifier
entity including the functions necessary for engaging in authentication exchanges on behalf of an entity requiring an entity authentication or for engaging in verifying a signature of a given message and signer

[SOURCE: ISO/IEC 9798-5:2009, modified]

4.20
verify
verification process that takes a message, a signature and an identity of a signer to output accept meaning the given signature is generated by the signer with the corresponding signing key, or reject otherwise

4.21
witness
procedure parameter that provides evidence of the claimant's identity to the verifier

[SOURCE: ISO/IEC 9798-5:2009, 2.31]

5 Symbols and abbreviated terms

5.1 Symbols

For the purposes of this part of ISO/IEC 29167, the following symbols and abbreviated terms apply.

$\|A\|$	bit size of the number A if A is a non-negative integer (i.e. the unique integer i so that $2^{i-1} \leq A < 2^{i}$ if $A > 0$, or 0 if $A = 0$, e.g. $\|65\,537\| = \|2^{16}+1\| = 17$), or bit length of the bit string A if A is a bit string NOTE To represent a number A as a string of α bits with $\alpha > \|A\|$, $\alpha - \|A\|$ bits set to 0 are appended to the left of the $\|A\|$ bits.
$A[i]$	i^{th}-bit of the number A, where $A[0]$ is the right-most bit and $A[\|A\|-1]$ is the left-most bit
$A[i{:}j]$	bit string made of the bits from the i^{th}-bit to the j^{th}-bit of the number A, where $i > j$
$B \,\|\|\, C$	bit string resulting from the concatenation of data items B and C in the order specified. In cases where the result of concatenating two or more data items is input to a cryptographic algorithm as part of an authentication mechanism, this result shall be composed so that it can be uniquely resolved into its constituent data strings, i.e. so that there is no possibility of ambiguity in interpretation. This latter property could be achieved in a variety of different ways, depending on the application. For example, it could be guaranteed by (a) fixing the length of each of the substrings throughout the domain of use of the mechanism, or (b) encoding the sequence of concatenated strings using a method that guarantees unique decoding, e.g. using the distinguished encoding rules defined in ISO/IEC 8825-1 [10]
Y	response (procedure parameter)
C	challenge (procedure parameter)
Δ	byte length of fresh strings of random bits for representing challenges (domain parameter)
δ'	bit length of fresh strings of random bits for representing challenges (domain parameter)
S_c	set of challenges c
Z	derived challenge (procedure parameter)
Ω	byte length of the derived challenge z (procedure parameter)
ω'	bit length of the derived challenge z (procedure parameter)
S_z	set of derived challenges z
E	elliptic curve (domain parameter)
TRUNC	truncation function; $\text{TRUNC}_k(input)$ denotes the bitwise truncation of *input* to the k least significant (right-most) bytes
F	one-way function taking two inputs, a commitment X and a challenge c, and producing a derived challenge z
$\text{PRESENT}_K(B)$	encryption of the block B with the 128-bit key K, using the lightweight block cipher PRESENT
$\text{AES-}L_K(B)$	encryption of the block B with the L-bit key K, using the block cipher AES

P	base point over the elliptic curve E (domain parameter)
N	order of the base point P (domain parameter)
$[k]R$	multiplication operation that takes a positive integer k and a point R on the curve E as input and produces as output another point Q on the curve E, where $Q = [k]R = R + R + \ldots + R$ is the sum of k occurrences of R. The operation satisfies $[0]R = 0_E$ (the point at infinity), and $[-k]R = [k](-R)$
S	private key (secret parameter)
Σ	bit length of the secret key (domain parameter)
V	public key (public parameter)
V	byte length of the public key (domain parameter)
Q	field size (domain parameter)
$r,\{r_i\}$	fresh random number or fresh string of random bits, or indexed set thereof (secret parameter)
P	length of fresh strings of random bits for representing random numbers (domain parameter)
$X, \{X_i\}$	commitment, or indexed set thereof (procedure parameter)
X	security parameter, length of a commitment X (domain parameter)
Θ	security parameter (domain parameter)
Λ	bit length of the signature of the public key (public parameter)
$\{a, b, c, \ldots\}$	set containing the elements a, b, c, ...
0^k	string bit constructed with k zero bits
$CCCC_b$	binary notation
$CCCC_h$	hexadecimal notation

5.2 Abbreviated terms

CCR	Commitment Challenge Response
CS	Cryptographic Suite
CSI	Cryptographic Suite Identifier
ECDLP	Elliptic Curve Discrete Logarithm Problem
LHW	Low Hamming Weight
NTS	Non-transmissible Signature
RFU	Reserved for Future Use
TAM	Tag Authentication Method

6 Cipher introduction

This mechanism, cryptoGPS – called GPS in the earlier cryptographic literature – is due to Girault, Poupard, and Stern [3]. The name cryptoGPS is now used so as to avoid confusion with the physical location service. cryptoGPS is a zero-knowledge identification scheme that provides unilateral entity

authentication. Several variants of cryptoGPS are specified in ISO/IEC 9798-5 and ISO/IEC 29192-4 [15] with the elliptic curve variant [2], along with some optimizations, being presented below.

cryptoGPS is a *public key*, or *asymmetric,* cryptographic mechanism for Tag authentication that offers the potential for lightweight implementation on the Tag and security that is related to the difficulty of solving the *elliptic curve discrete logarithm problem* (ECDLP).

Two variants of the authentication are described:

— The Commitment-Challenge-Response (CCR) variant. CCR schemes, such as cryptoGPS, have previously been proposed as lightweight solutions to the problem of Tag authentication.

— The Non-Transmissible Signature (NTS) variant. This variant is based on the cryptoGPS signature scheme and reduces the number of exchanges between the reader and the Tag.

In addition cryptoGPS can be used with a variety of implementation optimizations. These include:

a) the use of what are termed *coupons*, essentially a pre-computation of the form *(r, X)* that can be stored by the claimant and used at the time of authentication, and

b) the use of a pseudo-random number generator that can be used to re-generate the first component *r* of the coupons in optimization 1, and

c) the use of a cryptographic hash function that can be used to reduce the size of the second component *X* of the coupons in optimization 1, and

d) the use of bitwise truncation that can be used to further reduce the size of the second component *X* of the coupons in optimization 3, and

e) the use of what are termed *low hamming weight (LHW) challenges*, available only to the CCR variant, that provides a carefully constructed challenge space offering some computational efficiencies to the claimant. A challenge is said to be LHW if there are at least $\sigma - 1$ zero bits between any two consecutive one bits in its binary representation (where σ is the bit length of the private key s).

All of these optimizations are optional and they can be used in combination.

Issues such as the key infrastructure required to support the techniques described in this Cryptographic Suite are outside the scope of the document. They remain, nevertheless, important considerations when assessing the suitability of any Cryptographic Suite for a given application.

7 Parameter definitions

cryptoGPS allows a verifier to check that a claimant knows the elliptic curve discrete logarithm of a claimed public point with respect to a base point. A general framework for cryptographic techniques based on elliptic curves is given in ISO/IEC 15946-1.

Within a given domain the following requirements shall be satisfied.

a) Domain parameters that govern the operation of the mechanism shall be selected. The selected parameters shall be made available in a reliable manner to all entities within the domain.

b) Every claimant shall be equipped with the same elliptic curve E and a set of parameters, namely the field size q, a base point P over E, and n the order of point P. The curve and the set of parameters are either domain parameters or claimant parameters.

c) Each point P used as the base for elliptic curve discrete logarithms shall be such that, for any arbitrary point J of the curve, finding an integer k in $[0, n - 1]$ (if one exists) such that $J = [k]P$ is computationally infeasible, where feasibility is defined by the context of use of the mechanism.

d) Every claimant shall be equipped with a private key.

e) Every verifier shall obtain an authentic copy of the public key corresponding to the claimant's private key.

NOTE 1 The exact means by which the verifier obtains a trusted copy of the public key of the claimant is beyond the scope of this part of ISO/IEC 29167. This may, for example, be achieved by the use of public-key certificates or by some other environment-dependent means.

f) Every verifier shall have the means to produce fresh strings of random bits. When coupons are not used, every claimant shall also have the means to produce fresh strings of random bits.

NOTE 2 The exact means by which the verifier and claimant obtain fresh strings of random bits is beyond the scope of this part of ISO/IEC 29167.

8 State diagram

This part of ISO/IEC 29167 shall implement an INITIAL state.

After power-up and after a reset of the Cryptographic Suite, the Cryptographic Suite transitions to its INITIAL state. After a successful Tag Authentication (TAM1 or TAM2) the Cryptographic Suite shall remain in its INITIAL state. If the Cryptographic Suite returns an error code, it goes to the INITIAL state.

The transition between states is specified in Figure 1 — State machine of cryptoGPS cryptographic engine (TAM1 and TAM2 mechanisms).

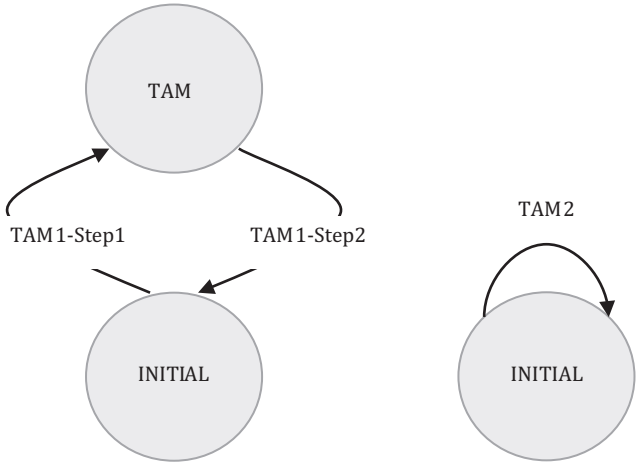

Figure 1 — State machine of cryptoGPS cryptographic engine (TAM1 and TAM2 mechanisms)

9 Initialization and resetting

Implementations of this suite shall assure that all memory used for intermediate results is cleared after each authentication operation and after reset.

For claimant A, a fresh string shall be uniformly selected at random from the set {2, 3, ..., n – 1}. The string represents the private key, denoted s.

The number $\sigma = |n|$ gives the number of bits to be used to represent private keys.

Denoted V, the public point for claimant A is set equal to the inverse of the multiplication of the base point P by the number s; namely

$$V = (x_V, y_V) = - [s]P$$

The challenges are selected from a set of integers S_c of cardinality Δ_c, where $2^{\delta'-1} < \Delta_c \leq 2^{\delta'}$. The length in bits of the greatest possible challenge is denoted by β. It is a domain parameter.

NOTE 1 The total number of possible challenges is application dependent. Values of δ' from 8 to 64 can be considered, but most applications would likely use δ' equal to 20 to 40.

NOTE 2 When the set of challenges is the interval $[0, \Delta_c - 1]$, then $\beta = \delta'$.

The derived challenges are selected from a set of integers S_z of cardinality Δ_z, where $2^{\omega'-1} < \Delta_z \leq 2^{\omega'}$.

NOTE 3 The total number of possible derived challenges is application dependent. Values of ω' from 8 to 64 bits can be considered, but most applications would likely use ω' equal to 20 to 40 bits.

NOTE 4 In the CCR variant, when the LHW optimization is not used, then $\delta' = \omega'$ and $c = z$.

In the NTS variant, the derived challenge z is computed as the result of the one-way function F with input a commitment X and a challenge c:

$$z = \mathrm{F}(X, c).$$

The F function is based either on the lightweight block cipher PRESENT, on the block cipher AES, or on the hash function SHA-256.

NOTE 5 PRESENT is a standardized lightweight block cipher and is described in ISO/IEC 29192-2 [14]. AES is the block cipher described in FIPS-197 [19]. SHA-256 is the fourth hash function specified in ISO/IEC 10118-3 [12].

10 Authentication

10.1 Introduction

This section outlines two variant mechanisms using cryptoGPS that provide Tag authentication.

The claimant shall store a private key s (as a string of σ bits).

Optionally, the claimant may store the corresponding public key V and a certificate C on that public key V issued by a trusted authority.

If the coupon optimization is not used, the claimant shall store the agreed base point P and the parameters for associated elliptic curve operations on the pre-selected elliptic curve E.

If the coupon optimization is used, the claimant shall store m coupons C_i where $0 \leq i \leq m-1$. The exact form of the coupons stored will depend on the additional optimizations that are used.

In the case of coupon use, in addition to:

— number δ and a private key s, in the CCR variant ;

— numbers δ and ω, and a private key s, in the NTS variant ;

the claimant shall only store a set of coupons. To be used only once, each coupon consists of a ρ-bit string (that need not be stored if it can be reproduced by a pseudo-random function, e.g. one of the functions specified in ISO/IEC 18031[16]) and a commitment.

The bracketed letters in Figures 2 and 3 correspond to the steps of the mechanism, including the exchanges of information, described in detail below. The claimant is denoted by A. The verifier is denoted by B.

10.2 Tag authentication: CCR variant (Method "00" = TAM1)

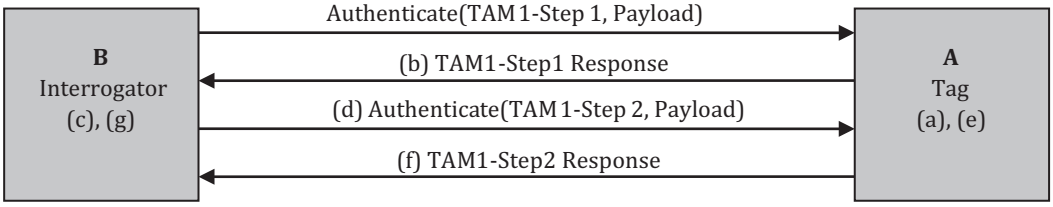

Figure 2 — Commitment-Challenge-Response variant

For each application of the mechanism the following procedure shall be performed. The verifier *B* (the Interrogator) shall only accept the claimant *A* (the Tag) as valid if the procedure completes successfully.

a) For each authentication,

1) either a new coupon (r_i, X_i) is used, or

2) a fresh string of ρ bits shall be uniformly selected at random. It shall be kept secret.

$$\rho = \sigma + \omega' + \theta$$

NOTE 1 A value of θ of 80 is appropriate for most applications.

NOTE 2 If the LHW optimization is not used, then $\omega' = \delta'$. Else, $\omega' = 256 \times \delta + (\delta - 1) \times (\sigma - 1)$.

NOTE 3 The ρ-bit value r_i shall not be equal to zero.

i) Denoted r_i, the number represented by the fresh string shall be converted into a commitment (used as witness) X_i as

$$X_i = \text{EC2OSP}_E([r_i]P, \text{fmt})$$

with EC2OSP_E the function to convert a point on elliptic curve E to octet strings defined in ISO/IEC 15946-1 and fmt a format specifier, which is one of the symbolic values compressed, uncompressed, or hybrid.

NOTE 4 Under certain circumstances some might prefer to use the witness formula $X_i = \text{EC2OSP}_E([r_i \bmod n]P, \text{fmt})$.

ii) Then, in addition and depending on Tag initialization, optionally set $X_i = \text{SHA-256}(X_i)$.

NOTE 5 SHA-256 is the fourth hash function specified in ISO/IEC 10118-3 [12].

iii) Then, in addition and depending on Tag initialization, optionally set $X_i = \text{TRUNC}_x(X_i)$.

NOTE 6 The size of the witness will depend on application security demands.

b) *A* sends commitment X_i to *B*.

c) On receipt of commitment X_i, the following steps are performed:

1) If the commitment length x, sent by *A*, is lower than application security policies allow, then the procedure fails.

2) If the desired challenge length δ, sent by *A*, does not satisfy application security policies then the procedure fails.

3) A fresh challenge string c shall be uniformly selected at random from S_c so that a derived challenge z lies in the set S_z and is not equal to zero.

d) B sends challenge c to A.

NOTE 7 If the LHW optimization is not used then $z = c$, and c shall not be equal to zero.

e) On receipt of the challenge, the following computational steps are performed.

1) If the challenge c is not an element of S_c, then the procedure fails, and the commitment X_i can be re-used.

2) A computes a derived challenge z as follow:

i) If the LHW optimization is not used, then $z = c$.

ii) If the LHW optimization is used, then the following computational steps are performed:

I) The δ-byte challenge c can be denoted as: $a_\delta \,||\, a_{\delta-1} \,||\, \ldots \,||\, a_2 \,||\, a_1$.

II) Starting from the least significant bit of z, there are a_1 zero bits before the first '1'.

III) There are then $(\sigma - 1) + a_i$ zeros between the $(i-1)^{th}$ and the i^{th} '1'.

3) If the derived challenge z is not an element of S_z, or is equal to zero, then the procedure fails, and the commitment X_i can be re-used.

4) A response y shall be computed (as an integer) using the number r_i and the private key s as

$$y = r_i + z \times s$$

f) A sends response y to B.

NOTE 8 A shall now delete coupon (r_i, X_i), or otherwise make it functionally inaccessible, in order to forbid the reuse of the same coupon for another authentication.

g) On receipt of response y, the following computational steps are performed.

1) If the response y is not a string of ρ bits and/or if the leftmost θ bits of y are all equal, then the procedure fails.

2) Denoted X^*, a candidate witness shall be computed as follows:

i) $X^* = \text{EC2OSP}_E([z]V + [y]P, \text{fmt})$

NOTE 9 Some might prefer to use the verification formula $X^* = \text{EC2OSP}_E([z]V + [y \bmod n]P, \text{fmt})$.

ii) Then, in addition and depending on Tag initialization, optionally set $X^* = \text{SHA-256}(X^*)$.

iii) Then, in addition and depending on Tag initialization, optionally set $X^* = \text{TRUNC}_x(X^*)$

3) If $X^* = X_i$ then verification is successful and the claimant is accepted as authentic. Otherwise the procedure fails.

10.3 Tag authentication: NTS variant (Method "01" = TAM2)

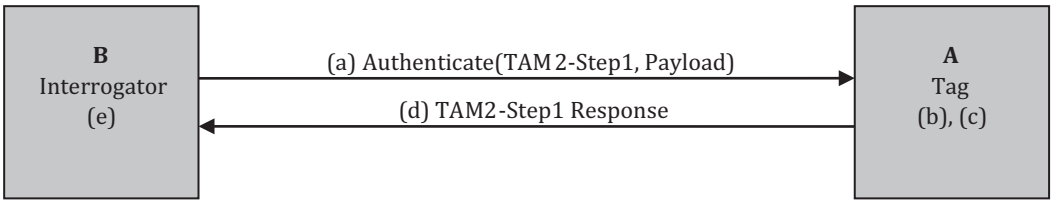

Figure 3 — Non-Transmissible Signature variant

For each application of the mechanism, the following procedure shall be performed. The verifier *B* (the Interrogator) shall only accept the claimant *A* (the Tag) as valid if the procedure completes successfully.

a) B sends to *A* a fresh string *c* of length δ bytes.

b) On receipt of the challenge *c*, the following steps are performed by *A*.

   1) If the length of *c* does not satisfy application security policies then the procedure fails.

c) Then, within *A*:

   1) either a new coupon (r_i, X_i) is used, or

   2) a fresh string of ρ bits shall be uniformly selected at random. It shall be kept secret.

$$\rho = \sigma + \omega' + \theta$$

NOTE 1 A value of θ of 80 is appropriate for most applications.

NOTE 2 The ρ-bit value r_i shall not be equal to zero.

i) Denoted r_i, the number represented by the fresh string shall be converted into a commitment (used as witness) X_i as

$$X_i = \text{EC2OSP}_E([r_i]P, \text{fmt})$$

with EC2OSP_E the function to convert a point on elliptic curve *E* to octet strings defined in ISO/IEC 15946-1 and fmt a format specifier, which is one of the symbolic values compressed, uncompressed, or hybrid.

NOTE 3 Under certain circumstances some might prefer to use the witness formula $X_i = \text{EC2OSP}_E([r_i \bmod n]P, \text{fmt})$.

ii) Then, in addition and depending on Tag initialization, optionally set $X_i = \text{SHA-256}(X_i)$.

iii) Then, in addition and depending on Tag initialization, optionally set $X_i = \text{TRUNC}_x(X_i)$.

NOTE 4 The size of the witness will depend on application security demands.

3) A derived challenge *z* shall be computed using the commitment X_i and the challenge *c* as

$$z = \text{F}(X_i, c)$$

The F function can be based either on the lightweight block cipher PRESENT, on the block cipher AES, or on the hash function SHA-256.To compute *z*, the following steps are performed:

i) The commitment X_i and the challenge *c* are concatenated to get $K = X_i \,||\, c$.

ii) If PRESENT is used (depending on Tag initialization), the following computational steps are performed:

I) If the length of K is greater than 128 bits, then the procedure is aborted and the commitment X_i can be re-used.

II) If the length of K is lower than 128 bits, then K is expanded with zero bits at the most significant (left-most) bit positions until its length is 128 bits: $K = 0...0 \,||\, K$.

III) With K as key, the lightweight block cipher PRESENT is used to encrypt the 64-bit zero string bit:

$$z = \text{PRESENT}_K(0^{64})$$

NOTE 5 PRESENT is a standardized lightweight block cipher and is described in ISO/IEC 29192-2.[14]

iii) Else if AES is used (depending on Tag initialization), let L be the key length used (L is equal to 128, 192 or 256 bits) and let AES-L be the corresponding version of AES (AES-L is either AES-128, AES-192 or AES-256). The following steps are performed:

I) If the length of K is greater than L bits, then the procedure is aborted and the commitment X_i can be re-used.

II) If the length of K is lower than L bits, then K is expanded with zero bits at the most significant (left-most) bit positions until its length is L bits: $K = 0...0 \,||\, K$.

III) With K as key, the block cipher AES-L is used to encrypt the 128-bit zero string bit:

$$z = \text{AES-}L_K(0^{128})$$

NOTE 3 AES is the block cipher described in FIPS-197.[19]

iv) Else if SHA-256 is used (depending on Tag initialization), then set $z = \text{SHA-256}(K)$.

v) Then, in addition and depending on Tag initialization, optionally set $z = \text{TRUNC}_\omega(z)$.

4) If z is equal to zero the authentication procedure is aborted and the commitment X_i can be re-used.

5) If the length of z does not satisfy application security policies then the procedure fails and the commitment X_i can be re-used.

6) Otherwise response y shall be computed (as an integer) using the number r_i and the private key s as

$$y = r_i + z \times s$$

d) A sends response y and derived challenge z to B.

NOTE 4 A shall now delete coupon (ri, Xi), or otherwise make it functionally inaccessible, in order to forbid the reuse of the same coupon for another authentication.

e) On receipt of response y and derived challenge z, the following computational steps are performed.

1) If the response y is not a string of ρ bits and/or if the leftmost θ bits of y are all equal, then the procedure fails.

2) If the derived challenge z is equal to zero, then the procedure fails.

3) If the length of z does not satisfy application security policies then the procedure fails.

4) If the witness length x does not satisfy application security policies then the procedure fails.

5) Denoted X^*, a candidate witness shall be computed as follows:

   i) $X^* = \text{EC2OSP}_E([z]V + [y]P, \text{fmt})$

   NOTE 5 Some might prefer to use the verification formula $X^* = \text{EC2OSP}_E([z]V + [y \bmod n]P, \text{fmt})$.

   ii) Then, in addition and depending on Tag initialization, optionally set $X^* = \text{SHA-256}(X^*)$.

   iii) Then, in addition and depending on Tag initialization, optionally set $X^* = \text{TRUNC}_x(X^*)$.

6) Denoted z^*, a candidate derived challenge shall be computed as follows:

   i) The candidate witness X^* and the challenge c are concatenated to get $K^* = X^* \,||\, c$.

   ii) If PRESENT is used (depending on Tag initialization), the following computational steps are performed:

      I) If the length of K^* is greater than 128 bits, then the procedure is aborted.

      II) If the length of K^* is lower than 128 bits, then K^* is expanded with zero bits at the most significant (left-most) bit positions until its length is 128 bits: $K^* = 0\ldots0 \,||\, K^*$.

      III) With K^* as key, the lightweight block cipher PRESENT is used to encrypt the 64-bit zero string bit:

$$z^* = \text{PRESENT}_{K^*}(0^{64})$$

   iii) Else if AES is used (depending on Tag initialization), let L be the key length used (L is equal to 128, 192 or 256 bits) and let AES-L be the corresponding version of AES (AES-L is either AES-128, AES-192 or AES-256). The following steps are performed:

      I) If the length of K* is greater than L bits, then the procedure is aborted.

      II) If the length of K* is lower than L bits, then K* is expanded with zero bits at the most significant (left-most) bit positions until its length is L bits: K* = 0…0 || K*.

      III) With K* as key, the block cipher AES-L is used to encrypt the 128-bit zero string bit:

$$z^* = \text{AES-}L_{K^*}(0^{128})$$

   iv) Else if SHA-256 is used (depending on Tag initialization), then set $z^* = \text{SHA-256}(K^*)$.

   v) Then, in addition and depending on Tag initialization, optionally set $z^* = \text{TRUNC}_\omega(z^*)$.

7) If $z^* = z$ then verification is successful and the claimant is accepted as authentic. Otherwise the procedure fails.

Message layout for authentication methods TAM1 and TAM2

For both Tag authentications variants described in 9.2 and 9.3, a sequence of one or more commands needs to be sent to the Tag in order to complete authentication with the authentication methods in this Cryptographic Suite. In order for an authentication to succeed, the entire sequence of commands needs to be executed successfully.

Message and Response are part of the security commands that are described in the air interface specification. The following sections of this document describe the formatting of Message and Response for the Tag authentication methods TAM1 and TAM2 and the payloads for each step of the authentication sequence will be detailed below.

State transition tables are provided in Annex A.

10.3.1 CCR variant (Method "00" = TAM1)

Tag authentication using a commitment-challenge-response protocol.

10.3.1.1 TAM1-Step1: Interrogator payload formatting

The Interrogator starts the authentication process and constructs a TAM1-Step1 payload defined below:

field	AuthMethod	Step	Flags
# bits	2	2	4

Figure 4 — TAM1-Step1 Interrogator message

The fields have the following interpretation:

— ***AuthMethod***: This field is set to 00.

— ***Step***: This field is set to 00.

— ***Flags***: This field is set to the following values:

  — *Flags*[0] = 0 if the Interrogator doesn't need to receive the Tag public key from the Tag.

  — *Flags*[0] = 1 if the Interrogator needs to receive the Tag public key from the Tag.

  — *Flags*[3:1] = 000.

  NOTE The value of Flags[3:1] can be viewed as RFU.

10.3.1.2 TAM1-Step1: Tag processing

The Tag shall only accept this payload in the INITIAL state. In any other state, receipt of this message generates an error.

The Tag checks if the following conditions are fulfilled:

1) The Tag checks that *AuthMethod* is equal to 00. If not, the Tag returns error code ERR_AUTHMETHOD and stays in INITIAL state.

2) The Tag checks that *Step* is equal to 00. If not, the Tag returns error code ERR_STEP and stays in INITIAL state.

3) The Tag checks that the received value of *Flags* is compatible with whether the Tag public key is stored on the Tag. If not, the Tag returns error code ERR_PUBKEY and stays in INITIAL state.

The Tag then performs the following steps:

1) It generates, or retrieves from memory, an unused commitment X_i of length x bytes. If the Tag is unable to produce a new commitment X_i, then it returns error code ERR_COMMITMENT and stays in INITIAL state.

2) It constructs a response message as described below.

3) It moves to state TAM.

10.3.1.3 TAM1-Step1: Tag response formatting

The full response payload is defined below:

							if requested		
field	**AuthMethod**	**Step**	**Flags**	**Length δ**	**Length x**	**Commitment X_i**	**Length v**	**Public Key V**	**Signature CA**
# bits	2	2	4	4	4	$8 \times x$	8	$8 \times v$	λ

Figure 5 — TAM1-Step1 Tag message

The fields have the following interpretation:

— ***AuthMethod***: This field is set to 00.

— ***Step***: This field is set to 00.

— ***Flags***: This field is set to the following values:

  — *Flags*[0] = 0 if the LHW challenge optimization is not used.

  — *Flags*[0] = 1 if the LHW challenge optimization is used.

  — *Flags*[1] = 0 if no hash function is used to construct the coupons.

  — *Flags*[1] = 1 if the hash function SHA-256 is used to construct the coupons.

  — *Flags*[2] = 0 if truncation is not used to construct the coupons.

  — *Flags*[2] = 1 if truncation is used to construct the coupons.

  — *Flags*[3] = 0.

  NOTE 1 The value of *Flags*[2:0] will likely be fixed at the time of Tag manufacture or personalization.

  NOTE 2 The value of *Flags*[3] can be viewed as RFU.

— ***Length δ***: The length in bytes of the challenge that is anticipated by the Tag.

  NOTE 3 The length δ will depend on application security considerations.

  NOTE 4 If the LHW challenge is used then δ also denotes the hamming weight of the derived challenge.

— ***Length x***: The length in bytes of the commitment X_i.

  NOTE 5 The length x will depend on application security considerations.

  NOTE 6 If truncation is used then x effectively communicates the extent of truncation to be used by the verifier.

— ***Commitment X_i***: The cryptoGPS Tag commitment.

Optionally included, depending on the value of *Flags* in the Interrogator authentication command:

— ***Length v***: The length in bytes of the representation of the Tag-specific public key V.

— ***Public key***: The Tag-specific public key V.

— ***Signature CA***: The signature (certificate) for the Tag-specific public key V. The choice of signature scheme will be system-wide and the size of the certification signature field will be fixed throughout.

10.3.1.4 TAM1-Step1: Interrogator processing

The Interrogator recovers the TAM1-Step1 Response and performs the following steps:

1) The Interrogator checks that *AuthMethod* is equal to *AuthMethod* from the TAM1-Step1 Interrogator message. If not, the Interrogator generates an error code and halts.

2) The Interrogator checks that *Step* is equal to *Step* from the TAM1-Step1 Interrogator message. If not, the Interrogator generates an error code and halts.

3) The Interrogator authenticates the Tag public key *V*. If the Tag public key cannot be authenticated, the Interrogator generates an error code and halts.

4) The Interrogator verifies that it supports the Tag type identified by the value of *Flags*. If not, the Interrogator generates an error code and halts.

5) The Interrogator "saves" the status of the Tag by setting three boolean flags as follows:

B_{LHW} = TRUE if *Flags*[0] = 1, FALSE otherwise

B_{HASH} = TRUE if *Flags*[1] = 1, FALSE otherwise

B_{TRUNC} = TRUE if *Flags*[2] = 1, FALSE otherwise

Then the Interrogator performs the following steps:

1) If the commitment length *x*, sent by the Tag, is lower than associated application security policies allow then the Interrogator generates an error code and halts.

2) The Interrogator checks that the requested length δ of the challenge satisfies associated application security policies. If it's not then the Interrogator generates an error code and halts.

3) The Interrogator stores the value of X_i in user-memory.

4) Depending on the value of *Flags*, the Interrogator constructs the message for TAM1-Step2 in one of two ways.

10.3.1.5 TAM1-Step2: Interrogator message formatting

The Interrogator shall compute a non-zero challenge *c*. This will be interpreted by the Tag as a derived challenge in one of two ways, depending on the value of *Flags* received in the response to TAM1-Step1.

The Interrogator shall construct a TAM1-Step2 payload.

The full message payload is defined below:

field	**AuthMethod**	**Step**	**Flags**	**Challenge *c***
# bits	2	2	4	$8 \times \delta$

Figure 6 — TAM1-Step2 Interrogator message

The fields have the following interpretation:

— ***AuthMethod***: This field is set to 00.

— ***Step***: This field is set to 01.

— ***Flags***: This field is set to 0001 if B_{LHW} = TRUE and to 0000 otherwise.

— ***Challenge***: This field contains a random challenge *c*.

  — Case B_{LHW} = FALSE: The challenge *c* and the derived challenge *z* will be the same non-zero random string of length δ bytes.

  — Case B_{LHW} = TRUE: The challenge *c* represents a δ-byte encoding of a non-zero random derived challenge *z* that has hamming weight δ.

10.3.1.6 TAM1-Step2: Tag processing

The Tag shall only accept this payload in the TAM state. In any other state, receipt of this message generates an error.

The Tag checks that the following conditions are fulfilled:

1) The Tag checks that *AuthMethod* is equal to 00. If not, the Tag returns error code ERR_AUTHMETHOD and returns in INITIAL state.

2) The Tag checks that *Step* is equal to 01. If not, the Tag returns error code ERR_STEP and returns in INITIAL state.

3) The length of *c* corresponds to the length (δ bytes) demanded. If not, the Tag returns error code ERR_CHALLENGE and returns to the INITIAL state.

4) The length of the derived challenge *z* satisfies application security policies. If not, the Tag returns error code ERR_CHALLENGE and returns to the INITIAL state.

5) The derived challenge *z* is non-zero. If not the Tag returns error code ERR_CHALLENGE and returns to the INITIAL state.

NOTE 1 When using the LHW variant of cryptoGPS, the encoding method necessarily implies that *z* is non-zero since it must have hamming weight δ.

If these conditions are fulfilled the Tag performs the following steps:

1) The Tag generates, or retrieves from memory, the pseudo-random number r_i of length ρ associated with coupon of index *i*.

2) The Tag uses *z*, r_i, and the Tag authentication key *s* to compute the cryptoGPS response *y*.

NOTE 2 When the LHW challenge is not used then $z = c$. When the LHW challenge is used, the δ-byte challenge *c* gives the derived value *z*.

3) The Tag constructs a response containing the value *y*, transmits the response and disables reuse of the selected coupon.

10.3.1.7 TAM1-Step2: Tag response formatting

The full response payload is defined below:

field	AuthMethod	Step	Response *y*
# bits	2	2	ρ

Figure 7 — TAM1-Step2 Tag message

The fields have the following interpretation:

— ***AuthMethod***: This field is set to 00.

— ***Step***: This field is set to 01.

— ***Response y***: The ρ-bit cryptoGPS Tag response.

10.3.1.8 TAM1-Step2: Interrogator response processing

The Interrogator recovers the TAM1-Step2 Response and performs the following steps:

1) The Interrogator checks that *AuthMethod* is equal to *AuthMethod* from the TAM1-Step2 Interrogator message. If not, the Interrogator generates an error code and halts.

2) The Interrogator checks that *Step* is equal to *Step* from the TAM1-Step2 Interrogator message. If not, the Interrogator generates an error code and halts.

3) The Interrogator uses the derived challenge z, the response y, the public key V, and the elliptic curve system parameters to compute $X^* = \text{EC2OSP}_E([z]V + [y]P, \text{fmt})$.

4) If B_{HASH} = TRUE, the Interrogator computes $X^* = \text{SHA-256}(X^*)$.

5) If B_{TRUNC} = TRUE, the Interrogator computes $X^* = \text{TRUNC}_x(X^*)$.

6) If $X^* = X_i$ then verification succeeds.

If verification succeeds the Tag is authenticated. In all other cases the Tag is not authenticated.

10.3.2 NTS variant (Method "01" = TAM2)

Tag authentication using a challenge-response protocol.

10.3.2.1 TAM2: Interrogator message formatting

The Interrogator starts the authentication process and constructs a TAM2 payload.

field	AuthMethod	Flags	Length δ	Challenge c
# bits	2	2	4	$8 \times \delta$

Figure 8 — TAM2 Interrogator message

The fields have the following interpretation:

— ***AuthMethod***: This field is set to 01.

— ***Flags***: This field is set to the following values:

  — *Flags*[0] = 0 if the Interrogator doesn't need to receive the Tag public key from the Tag.

  — *Flags*[0] = 1 if the Interrogator needs to receive the Tag public key from the Tag.

  — *Flags*[1] = 0.

  NOTE The value of Flags[1] can be viewed as RFU.

— ***Length δ***: The length in bytes of the challenge c.

— ***Challenge c***: The challenge.

10.3.2.2 TAM2: Tag processing

The Tag shall always accept this payload since the Tag is always in the INITIAL state.

The Tag checks if the following conditions are fulfilled:

1) The Tag checks that *AuthMethod* is equal to 01. If not, the Tag returns error code ERR_AUTHMETHOD, halts, and stays in INITIAL state.

2) The Tag checks that the length δ of the challenge c satisfies associated application security policies. If not, the Tag returns an error code ERR_CHALLENGE, halts, and stays in INITIAL state.

3) The Tag checks that the received value of *Flags* is compatible with whether the Tag public key is stored on the Tag. If not, the Tag returns error code ERR_PUBKEY, halts, and stays in INITIAL state.

The Tag then performs the following steps:

1) It generates, or retrieves from memory, an unused commitment X_i of length x bytes. If the Tag is unable to produce a new commitment X_i, then it returns error code ERR_COMMITMENT and stays in INITIAL state.

2) With commitment X_i and challenge c, it computes a derived challenge z of length that satisfies the application security policies. If the length of X_i and c does not allow to compute z, then the Tag returns an error code ERR_CHALLENGE, halts, and stays in INITIAL state.

   NOTE 1 In this situation the selected coupon has not been used in the computation of a Tag response and it can be reused.

3) If the derived challenge z is equal to zero, then the Tag returns an error code ERR_CHALLENGE, halts, and stays in INITIAL state.

   NOTE 2 In this situation the selected coupon has not been used in the computation of a Tag response and it can be reused.

4) If the derived challenge z is non-zero, the Tag uses z, r_i, and the Tag authentication key s to compute the cryptoGPS response y.

5) The Tag constructs a response containing the value y, transmits the response and disables reuse of the selected coupon.

10.3.2.3 TAM2: Tag response formatting

The full response payload is defined below:

							if requested		
field	**AuthMethod**	**Flags**	**Length ω**	**Derived challenge z**	**Length x**	**Response y**	**Length v**	**Public Key V**	**Signature CA**
# bits	2	6	4	$8 \times \omega$	4	ρ	8	$8 \times v$	λ

Figure 9 — TAM2 Tag message

The fields have the following interpretation:

— ***AuthMethod***: This field is set to 01.

— ***Flags***: This field is set to the following values:

  — *Flags*[2:0] = 000 if SHA-256 is used to compute the derived challenge z.

— *Flags*[2:0] = 001 if PRESENT is used to compute the derived challenge *z*.

— *Flags*[2:0] = 010 if AES-128 is used to compute the derived challenge *z*.

— *Flags*[2:0] = 011 if AES-192 is used to compute the derived challenge *z*.

— *Flags*[2:0] = 100 if AES-256 is used to compute the derived challenge *z*.

— *Flags*[3] = 0 if no hash function is used to construct the coupons.

— *Flags*[3] = 1 if the hash function SHA-256 is used to construct the coupons.

— *Flags*[4] = 0 if truncation is not used to construct the coupons.

— *Flags*[4] = 1 if truncation is used to construct the coupons.

— *Flags*[5] = 0 if truncation is not used to compute the derived challenge *z*.

— *Flags*[5] = 1 if truncation is used to compute the derived challenge *z*.

NOTE 1 The value of *Flags*[5:0] will likely be fixed at the time of Tag manufacture or personalization.

— ***Length ω***: The length in bytes of the derived challenge.

NOTE 2 The length ω will depend on application security considerations.

NOTE 3 If truncation is used then ω effectively communicates the extent of truncation to be used by the verifier.

— ***Derived challenge z***: The derived challenge computed by the Tag.

— ***Length x***: The length in bytes of the commitment X_i.

NOTE 4 The length *x* will depend on application security considerations.

NOTE 5 If truncation is used then *x* effectively communicates the extent of truncation to be used by the verifier.

— ***Response y***: The ρ-bit cryptoGPS Tag response.

Optionally included, depending on the value of *Flags* in the Interrogator authentication command:

— ***Length v***: The length in bytes of the representation of the Tag-specific public key *V*.

— ***Public key***: The Tag-specific public key *V*.

— ***Signature CA***: The signature (certificate) for the Tag-specific public key *V*. The choice of signature scheme will be system-wide and the size of the certification signature field will be fixed throughout.

10.3.2.4 TAM2: Interrogator response processing

The Interrogator recovers the TAM2 Response and performs the following steps:

1) The Interrogator checks that *AuthMethod* is equal to *AuthMethod* from the TAM2 Interrogator message. If not, the Interrogator generates an error code and halts.

2) If the response *y* is not a string of ρ bits and/or if the leftmost θ bits of *y* are all equal, then the procedure fails.

3) The Interrogator checks that the length *x* of the commitment *X* satisfies associated application security policies. If not then the Interrogator generates an error code and halts.

4) If the length ω of the derived challenge *z* does not satisfy application security policies then the Interrogator generates an error code and halts.

5) If the derived challenge *z* is equal to zero, then the Interrogator generates an error code and halts.

6) The Interrogator authenticates the Tag public key V. If the Tag public key cannot be authenticated, the Interrogator generates an error code and halts.

7) The Interrogator verifies that it supports the Tag type identified by the value of *Flags*. If not, the Interrogator generates an error code and halts.

8) The Interrogator "saves" the status of the Tag by setting three boolean flags and two additional values as follows:

F_{DERIV} = "SHA-256" if *Flags*[2:0] = 000

F_{DERIV} = "PRESENT" if *Flags*[2:0] = 001

F_{DERIV} = "AES" and L = 128 if *Flags*[2:0] = 010

F_{DERIV} = "AES" and L = 192 if *Flags*[2:0] = 011

F_{DERIV} = "AES" and L = 256 if *Flags*[2:0] = 100

B_{HASH} = TRUE if *Flags*[3] = 1, FALSE otherwise

B_{TRUNC1} = TRUE if *Flags*[4] = 1, FALSE otherwise

B_{TRUNC2} = TRUE if *Flags*[5] = 1, FALSE otherwise

Then the Interrogator performs the following steps:

1) The Interrogator uses the derived challenge z, the response y, the public key V, and the elliptic curve system parameters to compute $X^* = EC2OSP_E([z]V + [y]P, fmt)$.

2) If B_{HASH} = TRUE, the Interrogator computes $X^* = SHA\text{-}256(X^*)$.

3) If B_{TRUNC1} = TRUE, the Interrogator computes $X^* = TRUNC_x(X^*)$.

4) The Interrogator computes $z^* = F(X^*, c)$. In order to do so, the following computational steps are performed:

   a) The Interrogator sets $K^* = X^* \,||\, c$.

   b) If F_{DERIV} = "PRESENT", the Interrogator performs the following steps:

      i) If the length of K^* is greater than 128 bits, then the Interrogator generates an error code and halts.

      ii) If the length of K^* is lower than 128 bits, then K^* is expanded with zero bits at the most significant (left-most) bit positions until its length is 128 bits: $K^* = 0...0 \,||\, K^*$.

      iii) With K^* as key, the Interrogator uses the lightweight block cipher PRESENT to encrypt the 64-bit zero string bit: $z^* = \text{PRESENT}_{K^*}(0^{64})$.

   c) Else if F_{DERIV} = "AES", the Interrogator performs the following steps:

      i) If the length of K^* is greater than L bits, then the Interrogator generates an error code and halts.

      ii) If the length of K^* is lower than L bits, then K^* is expanded with zero bits at the most significant (left-most) bit positions until its length is L bits: $K^* = 0...0 \,||\, K^*$.

      iii) With K^* as key, the Interrogator uses the block cipher AES-L to encrypt the 128-bit zero string bit: $z^* = \text{AES-}L_{K^*}(0^{128})$.

   NOTE AES-L is either AES-128, AES-192, or AES-256 if L equals respectively 128, 192 or 256.

   d) Else if F_{DERIV} = "SHA-256", the Interrogator computes $z^* = SHA\text{-}256(K^*)$.

e) Then, if B_{TRUNC2} = TRUE, the Interrogator computes $z^* = TRUNC_{\omega}(z^*)$.

5) If $z^* = z$ then verification succeeds.

If verification succeeds the Tag is authenticated. In all other cases the Tag is not authenticated.

11 Communication

This cryptographic suite does not support secure communication.

12 Key table and key update

The Tag shall store in memory the following values:

— The private key value *s* which is used by the Tag for the computation of the response *y*. The private key value is σ bits long.

— If the coupon optimization is used: a set of *m* coupons.

— Optionally: the public key value *V* which is used by the Interrogator for the verification of the response *y*. The public key is *v* bytes long.

— Optionally: the certificate C on that public key *V*. The length of certificate is λ bits.

This cryptographic suite does not support other cryptographic functions such as updating the key.

Annex A
(normative)

State transition tables

Table A.1 — TAM1 Cryptographic state transitions

Payload	Start states	End state (success)	End state (failure)
TAM1-Step1	initial	tam	initial
TAM1-Step2	tam	initial	initial

Table A.2 — TAM1-Step1 Cryptographic state transitions

Start state	End state	Condition
initial	tam	Success
initial	initial	Error
tam	initial	Error

Table A.3 — TAM1-Step2 Cryptographic state transitions

Start state	End state	Condition
initial	initial	Error
tam	initial	Success
tam	initial	Error
tam	initial	Success

Table A.4 — TAM2 Cryptographic state transitions

Payload	Start states	End state (success)	End state (failure)
TAM2	initial	initial	initial

Table A.5 — TAM2 Cryptographic state transitions

Start state	End state	Condition
initial	initial	Success
initial	initial	Error

Annex B
(normative)

Error codes and error handling

Table B.1 — TAM Error codes

Error code	Description	Error condition
ERR_AUTHMETHOD	Incorrect *AuthMethod* value.	Message TAM1-Step1. The Tag checks that *AuthMethod* is equal to 00. If not, the Tag returns error code ERR_AUTHMETHOD and stays in INITIAL state. Message TAM1-Step2. The Tag checks that *AuthMethod* is equal to 00. If not, the Tag returns error code ERR_AUTHMETHOD and returns in INITIAL state. Message TAM2. The Tag checks that *AuthMethod* is equal to 01. If not, the Tag returns error code ERR_AUTHMETHOD, halts, and stays in INITIAL state.
ERR_STEP	Incorrect *Step* value.	Message TAM1-Step1. The Tag checks that *Step* is equal to 00. If not, the Tag returns error code ERR_STEP and stays in INITIAL state. Message TAM1-Step2. The Tag checks that *Step* is equal to 01. If not, the Tag returns error code ERR_STEP and returns in INITIAL state.
ERR_PUBKEY	Unavailable public key.	Message TAM1-Step1 and TAM2. The received value of *Flags* is not compatible with whether the Tag public key is stored on the Tag. The Tag returns error code ERR_PUBKEY and stays in INITIAL state.
ERR_COMMITMENT	Unavailable commitment.	Message TAM1-Step1. The Tag is unable to produce a new commitment. It returns error code ERR_COMMITMENT and stays in INITIAL state. Message TAM2. The Tag is unable to produce a new commitment. It returns error code ERR_COMMITMENT and stays in INITIAL state.

Table B.1 *(continued)*

Error code	Description	Error condition
ERR_CHALLENGE	Incorrect challenge.	Message TAM1-Step2. The length of c does not correspond to the length δ bytes demanded. The Tag returns error code ERR_CHALLENGE and returns to INITIAL state. Message TAM1-Step2. The derived challenge z computed by the Tag is equal to zero. The Tag returns error code ERR_CHALLENGE and returns to INITIAL state. Message TAM1-Step2. The length of the derived challenge z does not satisfy application security policies. The Tag returns error code ERR_CHALLENGE and returns to INITIAL state. Message TAM2. The length of the challenge c sent by the Interrogator does not satisfy associated application security policies. The Tag returns error code ERR_CHALLENGE and stays in INITIAL state. Message TAM2. With commitment X_i and challenge c, the Tag tries to computes a derived challenge z. The length of X_i or c does not allow to compute z. The Tag returns an error code ERR_CHALLENGE, halts, and stays in INITIAL state. Message TAM2. The derived challenge z computed by the Tag is equal to zero. The Tag generates an error code ERR_CHALLENGE, halts, and stays in INITIAL state.

Annex C
(normative)

Cipher description

The cryptoGPS authentication algorithm is described in References [3] and [1]. Implementation proposals (including a LHW version) and technical discussions can be found in References [2], [4], [5], and [6].

Annex D
(informative)

Test vectors

D.1 Key production

The elliptic E curve for this example is curve P-192 defined in FIPS PUB 186-3 [9].

$$E: Y^2 = X^3 - 3X + b \text{ over } F_q$$

q = FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE FFFFFFFF FFFFFFFF

b = 64210519 E59C80E7 0FA7E9AB 72243049 FEB8DEEC C146B9B1

Base point P over E.

P = (x_P, y_P)

= (188DA80E B03090F6 7CBF20EB 43A18800 F4FF0AFD 82FF1012,
07192B95 FFC8DA78 631011ED 6B24CDD5 73F977A1 1E794811)

n is the order of point P.

n = FFFFFFFF FFFFFFFF FFFFFFFF 99DEF836 146BC9B1 B4D22831

The bit length of the cryptoGPS private key is $\sigma = |n| = 192$ bits.

D.2 Authentication exchange: CCR variant

The lengths are the following:

— $\delta = \omega = 5$ bytes for the challenges c and z.

— $x = 49$ bytes for the commitment X.

Private key

s = 4F1DF03A A32DCA02 652E83E7 E5FF5259 D61F5563 B3A0FA10

Public point

$V = -[s]P$

$= (x_V, y_V)$

= (D753BF14 9529BC23 B1850A37 57C4D34A 0D686A95 C3B03855,
1656B8CB 2896BFD4 BC8F94A8 F3708741 B954CC44 4FC3951A)

Step a

r is a fresh string of random bits of length $\rho = \sigma + \omega' + \theta = 192 + 40 + 80 = 312$ bits.
X is the witness such that $X = \text{EC2OSP}_E([r]P, \texttt{uncompressed})$.

r = 05E8B1 E1121B08 FB9A0F58 FC1E932F 9CEFE94D 629BC223 40B5F04B
554DCD2B C812A76D 98F8BA3E

$[r]P$ = (DAD48D02 4B83E223 4C0F5FFF B51C15B7 1D52CF92 B35358CF,
FFE42756 843D0DF8 F3166971 E8AF6E22 6FD381B0 A816720F)

X = 04 DAD48D02 4B83E223 4C0F5FFF B51C15B7 1D52CF92 B35358CF
FFE42756 843D0DF8 F3166971 E8AF6E22 6FD381B0 A816720F

Step b

Commitment is equal to X.

X = 04 DAD48D02 4B83E223 4C0F5FFF B51C15B7 1D52CF92 B35358CF
FFE42756 843D0DF8 F3166971 E8AF6E22 6FD381B0 A816720F

Step c, d

Verifier's challenge:

c = 2D F0F5B4F2

Step e, f

z = 2D F0F5B4F2

y = 5E8B1 E1121B08 FB9A0F67 2ED9CE48 044BD618 3242087C ADDDA392
F2CA1F36 FDD94248 E8485D5E

Step g

Verification:

X^* = 04 DAD48D02 4B83E223 4C0F5FFF B51C15B7 1D52CF92 B35358CF
FFE42756 843D0DF8 F3166971 E8AF6E22 6FD381B0 A816720F

Authentication is valid.

D.3 Authentication exchange: NTS variant

The lengths are the following:

— $\delta = \omega = 8$ bytes for the challenges c and z.

— $x = 8$ bytes for the commitment X.

Private key

s = 4F1DF03A A32DCA02 652E83E7 E5FF5259 D61F5563 B3A0FA10

Public point

$V = -[s]P$

$= (x_V, y_V)$

= (D753BF14 9529BC23 B1850A37 57C4D34A 0D686A95 C3B03855,
1656B8CB 2896BFD4 BC8F94A8 F3708741 B954CC44 4FC3951A)

D.3.1 Challenge *z* derived with PRESENT

Step a, b

Verifier's challenge:

c = D2E49A1E 98917CA6

Step c

r is a fresh string of random bits of length $\rho = \sigma + \omega' + \theta = 192 + 64 + 80 = 336$ bits.
X is the witness such that $X = \mathrm{TRUNC}_8(\mathrm{SHA\text{-}256}(\mathrm{EC2OSP}_E([r]P, \mathtt{compressed})))$.

r = EA7E 7FD99858 4AB2612E 4D2BCA71 DBF57A64 28275FF6 7E1807D2
C82C2E28 9C9AE803 BCEAC8F0 51FE6A83

$[r]P$ = (8DC46B50 3906506F 9B348470 45186762 ACF8F187 22DCDFEB,
3C4BB35A 6A6EDE03 5D56E629 4BE81AB7 C495D387 EBA20DCC)

X = 4BAE0C3D F0A38D27

K = 4BAE0C3D F0A38D27 D2E49A1E 98917CA6

z = E5132316 5068D17C

y = EA7E 7FD99858 4AB2612E 93F77C67 218BF5D1 41D603CD 03C4FAB1
F7E1E66B 335E3784 32A77FCC 569E9A43

Step d

Response is equal to y.

y = EA7E 7FD99858 4AB2612E 93F77C67 218BF5D1 41D603CD 03C4FAB1
F7E1E66B 335E3784 32A77FCC 569E9A43

Derived challenge is equal to z.

z = E5132316 5068D17C

Step e

Verification:

X^* = 4BAE0C3D F0A38D27

K^* = 4BAE0C3D F0A38D27 D2E49A1E 98917CA6

z^* = E5132316 5068D17C

Authentication is valid.

D.3.2 Challenge *z* derived with AES-128

Step a, b

Verifier's challenge:

c = E223297E 5EC6F729

Step c

r is a fresh string of random bits of length $\rho = \sigma + \omega' + \theta = 192 + 64 + 80 = 336$ bits.
X is the witness such that $X = \mathrm{TRUNC}_8(\mathrm{SHA\text{-}256}(\mathrm{EC2OSP}_E([r]P, \texttt{compressed})))$.

r = D881 6DE2D0A9 37BCC0F0 E7A7FF7F AEF7502D 5B4A2B93 87C893A8
31031C61 4F1DD984 9EBD1B42 F86AE174

$[r]P$ = (0E580DBA B8BD37E0 096F158C D50743EE A5B42764 6F66ABF6
54BA4831 4DD0CA95 12BA072E E1E54BC7 4405C0E5 0B36EAFE)

X = 5DB43C92 01BB7C16

K = 5DB43C92 01BB7C16 E223297E 5EC6F729

z = C169886E 1610E61D

y = D881 6DE2D0A9 37BCC0F1 236E2F0D 5957EEC5 5F74D75A 1AE1A1B6
96C845E7 762FA92F 43405D5D F3519544

Step d

Response is equal to y.

y = D881 6DE2D0A9 37BCC0F1 236E2F0D 5957EEC5 5F74D75A 1AE1A1B6
96C845E7 762FA92F 43405D5D F3519544

Derived challenge is equal to z.

z = C169886E 1610E61D

Step e

Verification:

X^* = 5DB43C92 01BB7C16

K^* = 5DB43C92 01BB7C16 E223297E 5EC6F729

z^* = C169886E 1610E61D

Authentication is valid.

D.3.3 Challenge *z* derived with AES-192

Step a, b

Verifier's challenge:

c = D5BC55AD 9874221F

Step c

r is a fresh string of random bits of length $\rho = \sigma + \omega' + \theta = 192 + 64 + 80 = 336$ bits.
X is the witness such that $X = \text{TRUNC}_8(\text{SHA-256}(\text{EC2OSP}_E([r]P, \texttt{compressed})))$.

$r =$ 6619 F7652C72 67E81E79 F4013AD6 05A7B823 DB44A191 8B01E350
C7CA57DE 47FA9611 A2E8561D 8AC861A7

$[r]P =$ (C577AA8F E26E21DE 30101612 9B97F9D7 B6633039 227FFEFA
AB170D0B D808D97C A30508E3 7EEBF203 B9965C20 F1BB930F)

$X =$ 3EECAB5A 3BC7BB9D

$K =$ 00000000 00000000 3EECAB5A 3BC7BB9D D5BC55AD 9874221F

$z =$ 93DCD791 7D2762F7

$y =$ 6619 F7652C72 67E81E7A 21B3AC21 3F235930 BD7A2C46 59C59311
98BB3070 92604171 F0AAEEC3 6343C717

Step d

Response is equal to y.

$y =$ 6619 F7652C72 67E81E7A 21B3AC21 3F235930 BD7A2C46 59C59311
98BB3070 92604171 F0AAEEC3 6343C717

Derived challenge is equal to z.

$z =$ 93DCD791 7D2762F7

Step e

Verification:

$X^* =$ 3EECAB5A 3BC7BB9D

$K^* =$ 00000000 00000000 3EECAB5A 3BC7BB9D D5BC55AD 9874221F

$z^* =$ 93DCD791 7D2762F7

Authentication is valid.

D.3.4 Challenge z derived with AES-256

Step a, b

Verifier's challenge:

$c =$ E4741D5F 1A4DD9FB

Step c

r is a fresh string of random bits of length $\rho = \sigma + \omega' + \theta = 192 + 64 + 80 = 336$ bits.
X is the witness such that $X = \text{TRUNC}_8(\text{SHA-256}(\text{EC2OSP}_E([r]P, \texttt{compressed})))$.

$r =$ 483A D20CB5E2 8E6D3434 CBE5ABDB DC1A8128 20F7511E E52B3C40

019E2B24 A5C2707C A9CCF212 A62411F9

$[r]P$ = (2E442F49 6DA375D4 7AC84269 DFEF77C1 018A5468 8B5B1A22,
799E5382 48285D31 22ED82CF 91C1FE48 629E7524 18DEE10D)

X = 3EAB94C4 C73E8A9E

K = 00000000 00000000 00000000 00000000 3EAB94C4 C73E8A9E E4741D5F
1A4DD9FB

z = 916BD0B0 C7F02FC1

y = 483A D20CB5E2 8E6D3434 F8D6F2EF 7098F22D 3F623B41 6806D670
A15E22C6 C95F15B1 44BD1484 7F698809

Step d

Response is equal to y.

y = 483A D20CB5E2 8E6D3434 F8D6F2EF 7098F22D 3F623B41 6806D670
A15E22C6 C95F15B1 44BD1484 7F698809

Derived challenge is equal to z.

z = 916BD0B0 C7F02FC1

Step e

Verification:

X^* = 3EAB94C4 C73E8A9E

K^* = 00000000 00000000 00000000 00000000 3EAB94C4 C73E8A9E E4741D5F
1A4DD9FB

z^* = 916BD0B0 C7F02FC1

Authentication is valid.

D.3.5 Challenge z derived with SHA-256

Step a, b

Verifier's challenge:

c = 9BC9F1F7 B32739BA

Step c

r is a fresh string of random bits of length $\rho = \sigma + \omega' + \theta = 192 + 64 + 80 = 336$ bits.
X is the witness such that $X = \text{TRUNC}_8(\text{SHA-256}(\text{EC2OSP}_E([r]P, \texttt{compressed})))$.

r = 6409 8E79F049 4D17092D 8773EDDE B39F68E5 90A98014 95D0F204
9087F3B1 23756104 4F3A5320 A8A5943F

$[r]P$ = (72F286A8 2DEF31A7 D2291C5F F2F4BBF2 61FD7C35 8BF0FDA3,

A44782DD A1AACF13 A41145DA 0320DAD8 0A06C7E5 1D90BE58)

X = 03D7004B E8ED5513

K = 03D7004B E8ED5513 9BC9F1F7 B32739BA

z = 541F6897 7FD7AFC2

y = 6409 8E79F049 4D17092D A17375A5 0407393D EE55092B 08635CA9
B3008AB9 C8190379 0CAAE829 C704045F

Step d

Response is equal to y.

y = 6409 8E79F049 4D17092D A17375A5 0407393D EE55092B 08635CA9
B3008AB9 C8190379 0CAAE829 C704045F

Derived challenge is equal to z.

z = 541F6897 7FD7AFC2

Step e

Verification:

X^* = 03D7004B E8ED5513

K^* = 03D7004B E8ED5513 9BC9F1F7 B32739BA

z^* = 541F6897 7FD7AFC2

Authentication is valid.

Annex E
(normative)

Protocol specific

E.1 General

The cryptoGPS Crypto Suite provides security services for ISO/IEC 18000-3 mode 1, ISO/IEC 18000-3 mode 3 and ISO/IEC 18000-63 air interface protocols. Details of the specific implementation for these air interface protocols are described in sections below. ISO/IEC 29167-1 defines the Crypto Suite Identifier (CSI) for cryptoGPS to be 000111_2 and it is expanded to the 8-bit value 07_h for use by all air interface protocols in this Annex.

E.1.1 Supported Security Services

The security services that are supported by this cryptographic suite are specified in Table E.1 – Security Services.

Table E.1 — Security Services

Security Service	Method	Mandatory, optional, Prohibited, or not supported (NOTE 1)
Authentication		
Tag authentication (TA)		Mandatory
Interrogator authentication (IA)		Not supported
Mutual authentication (MA)		Not supported
Communication		
Authenticated Tag from TA	Authenticated communication (Tag => Interrogator)	Not supported
	Secure authenticated communication (Tag => Interrogator)	Not supported
Authenticated Interrogator from IA	Authenticated communication (Interrogator => Tag)	Not supported
	Secure authenticated communication (Interrogator => Tag)	Not supported
Authenticated Interrogator and Tag from MA	Authenticated communication (Interrogator <=> Tag)	Not supported
	Secure authenticated communication (Interrogator <=> Tag)	Not supported

E.2 Security Services for ISO/IEC 18000-3 mode 1

RFU – To be completed in the future when 18000-3 mode 1 information is available.

E.3 Security Services for ISO/IEC 18000-3 mode 3

RFU – To be completed in the future when 18000-3 mode 3 information is available.

E.4 Security Services for ISO/IEC 18000-63

Regarding the ISO/IEC 18000-63 Protocol Commands, the requirements that shall be fulfilled by this cryptographic suite (for both authentication methods TAM1 and TAM2) are the following:

— The tag shall not support the *Challenge* command in combination with Tag Authentication TAM1. The tag shall support the *Challenge* command in combination with Tag Authentication TAM2.

NOTE The NTS variant might be used with mechanisms for a broadcast challenge which makes this variant relevant if the *Challenge* command is available.

— The execution time for an authentication shall be below 200 ms.

— The Tag shall ignore commands from an Interrogator during execution of a cryptographic operation.

— The Tag shall not support sending the contents of the ResponseBuffer in the reply to an *ACK* command in combination with Tag Authentication TAM1. The tag shall support sending the contents of the ResponseBuffer in the reply to an *ACK* command in combination with TAM2 and *Challenge* command.

— The Tag shall support sending the contents of the ResponseBuffer in the reply to an *READ_BUFFER* command.

— The support of a security timeout following a cryptographic error by the Tag is optional.

— The *Authenticate* command shall be supported.

— 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 cryptographic commands with incorrect handle or CRC error.)

— For each Error Condition defined in the Cryptographic Suite, the Tag shall remain in its current state.

— The Tag shall remain in its current state after a Tag authentication.

— The following commands shall not be supported: *AuthComm*, *SecureComm*, *KeyUpdate*.

The messages for this crypto suite can be transmitted by the Interrogator to the Tag with the *Authenticate* or the *Challenge* command.

The *Authenticate* command is described in Figure E.1.

field	Command	RFU	SenRep	IncRepLen	CSI	Length	Message	RN	CRC
# bits	8	2	1	1	8	12	Variable	16	16

Figure E.1 — *Authenticate* command

The *Challenge* command is described in Figure E.2.

field	Command	RFU	IncRepLen	Immed	CSI	Length	Message	CRC
# bits	8	2	1	1	8	12	Variable	16

Figure E.2 — *Challenge* command

For the *Authenticate* and *Challenge* commands, the *Message* component shall be formatted according to the payloads sent by the Interrogator. These payloads are described in 9.4.1 and 9.4.2.

The messages for this crypto suite can be transmitted by the Tag to the Interrogator with the *Reply to Authenticate* and *Reply to Challenge* command.

The *Reply to Authenticate* and *Challenge* commands is described in Figure E.3.

may be omitted

field	Barker code	Done	Header	Length	Response	RN	CRC
# bits	7	1	1	16	Variable	16	16

Figure E.3 — *Reply to Authenticate* and *Reply to Challenge* commands

For the *Reply to Authenticate* and *Reply to Challenge* commands, the *Response* component shall be formatted according to the payloads sent by the Tag. These payloads are described in 9.4.1 and 9.4.2.

This crypto suite specifies error conditions when the authentication is not successful. The error codes of the crypto suite shall be returned to the Interrogator as error codes for the air interface.

In case of a functionality is not supported by the crypto suite, the Tag shall return the ISO/IEC 18000-63 error code equal to 00000001_b.

ISO/IEC 18000-63 requires the definition of key properties. Since this protocol does not provide Interrogator Authentication there are no keys for which key properties need to be provided. If an implementation however does provide key properties for a key belonging to this cryptographic suite (i.e. for the private key for the Tag authentication) it shall set the key properties to 0000_b and this value shall indicate the key can only be used for tag authentication.

Bibliography

[1] Girault M., & Lefranc D. *Public key authentication with one (online) single addition*, in CHES'04, pp 413-427, 2004

[2] Girault M., Juniot L., Robshaw M.J.B. *The feasibility of on-the-tag public key cryptography*, in RFIDSEC 2007, 11-13 july 2007

[3] Girault M., Poupard G., Stern J. On the fly authentication and signature schemes based on groups of unknown order. J. Cryptol. 2006, **19** (4) pp. 463–487

[4] McLoone M., & Robshaw M.J.B. *Public Key Cryptography and RFID*, in M. Abe, editor, Proceedings of CT-RSA 07, volume 4377 of LNCS, pp 372-384, Springer, 2007

[5] McLoone M., & Robshaw M.J.B. *New Architectures for Low-Cost Public Key Cryptography on RFID Tags*, in Proc. of IEEE International Conference on Security and Privacy of Emerging Areas in Communication Networks (SecureComm 2005), pp 1827-1830, IEEE, 2007

[6] Poschmann A., Robshaw M., Vater F., Paar C. *Lightweight Cryptography and RFID: Tackling the Hidden Overheads*, in D. Lee and S. Hong, editors, Proc. of ICISC-2009, volume 5984 of LNCS, pp 129-145, Springer, 2010

[7] Schnorr C.P. *Efficient identification and signatures for smart cards*, in Proceedings of CRYPTO '89, volume 435 of Lecture Notes in Computer Science, pages 239–252. Springer, 1990

[8] European Network of Excellence in Cryptology II. *ECRYPT II Yearly Report on Algorithms and Key Lengths (* 2010*)*, available on http://www.ecrypt.eu.org/documents/D.SPA.13.pdf

[9] National Institute of Standards and Technology. *Digital Signature Standard (DSS), FIPS Publication 186-3*, available on http://csrc.nist.gov/publications/PubsFIPS.html

[10] ISO/IEC 8825–1:2002, *Information technology — ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)*

[11] ISO/IEC 9796–3:2006, *Information technology — Security techniques — Digital signature schemes giving message recovery — Part 3: Discrete logarithm based mechanisms*

[12] ISO/IEC 10118–3:2004, *Information technology — Security techniques — Hash-functions — Part 3: Dedicated hash-functions*

[13] ISO/IEC 14888–3:2006, *Information technology — Security techniques — Digital signature with appendix — Part 3: Discrete logarithm based mechanisms*

[14] ISO/IEC 29192–2, *Information technology — Security techniques — Lightweight Cryptography — Part 2: Block ciphers*

[15] ISO/IEC 29192–4, *Information technology — Security techniques — Lightweight Cryptography — Part 4: Mechanisms using asymmetric techniques*

[16] ISO/IEC 18031:2011, *Information technology — Security techniques — Random bit generation*

[17] ISO/IEC 9798–1:2010, *Information technology — Security techniques — Entity authentication — Part 1: General*

[18] ISO/IEC 10118–1:2000, *Information technology — Security techniques — Hash-functions — Part 1: General*

[19] NIST FIPS Publication 197, *Advanced Encryption Standard (AES)*, November 26, 2001

[20] ISO/IEC 29192-1, *Information technology — Security techniques — Lightweight cryptography — Part 1: General*

ICS 35.040

Price based on 39 pages