INTERNATIONAL STANDARD

ISO/IEC 29167-12

First edition
2015-05-15

Information technology — Automatic identification and data capture techniques —

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

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

Partie 12: Services de sécurité par suite cryptographique ECC-DH pour communications par interface radio

Reference number
ISO/IEC 29167-12:2015(E)

Contents

Page

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

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

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

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

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

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

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 communication*

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

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

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

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

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

The following parts are under preparation:

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

Introduction

Elliptic curve cryptography (ECC) is an approach to public-key cryptography based on the algebraic structure of elliptic curves over finite fields. For elliptic-curve-based protocols, it is assumed that finding the discrete logarithm of a random elliptic curve element with respect to a publicly-known base point is computationally infeasible. The size of the elliptic curve determines the difficulty of the problem.

This part of ISO/IEC 29167 specifies the security services for an RFID Tag with an ECC-DH crypto suite based on the Diffie-Hellman key exchange algorithm. It specifies the details of a protocol and interface format for application with RFID Tags which provide unilateral authentication capability, based on the use of ECC. Although such Tags can operate in any frequency band legitimate for such applications, the main focus of this part of ISO/IEC 29167 is on externally-powered (also called “passive”) Tags designed for the HF/UHF frequency bands, where the demands on low silicon footprint and power consumption are most stringent.

This part of ISO/IEC 29167 defines only Tag authentication for the ECC-DH cipher.

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 may involve the use of patents concerning radio-frequency identification and cryptographic technologies 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 ensured the ISO and IEC that they are willing to negotiate licenses 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:

Impinj, Inc.
701 N 34th Street, Suite 300 Seattle, WA 98103 USA

The latest information on IP that may 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 12: Crypto suite ECC-DH security services for air interface communication

1 Scope

This part of ISO/IEC 29167 defines the crypto suite for ECC-DH for the ISO/IEC 18000 air interfaces standards for radio frequency identification (RFID) devices. Its purpose is to provide a common crypto suite with Diffie-Hellmann-based authentication using ECC (elliptic curve cryptography) over binary fields for security for RFID devices that may be referred by ISO committees for air interface standards and application standards.

This part of ISO/IEC 29167 specifies a crypto suite for ECC-DH for air interface for RFID systems. The crypto suite is defined in alignment with existing air interfaces.

This part of ISO/IEC 29167 defines various authentication methods and methods of use for the cipher. A Tag and an Interrogator may support one, a subset, or all of the specified options, clearly stating what is supported.

2 Conformance

2.1 Claiming conformance

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

2.2 Interrogator conformance and obligations

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

— implement the mandatory commands defined in this part of ISO/IEC 29167, and conform to the relevant part of ISO/IEC 18000.

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

— implement any subset of the optional commands defined in this part of ISO/IEC 29167.

To conform to this part of ISO/IEC 29167, the Interrogator shall not

— implement any command that conflicts with this part of ISO/IEC 29167, or

— require the use of an optional, proprietary, or custom command to meet the requirements of this part of ISO/IEC 29167.

2.3 Tag conformance and obligations

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

— implement the mandatory commands defined in this part of ISO/IEC 29167 for the supported types, and conform to the relevant part of ISO/IEC 18000.

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

— implement any subset of the optional commands defined in this part of ISO/IEC 29167.

To conform to this part of ISO/IEC 29167, a Tag shall not

— implement any command that conflicts with this part of ISO/IEC 29167, or

— require the use of an optional, proprietary, or custom command to meet the requirements of this part of ISO/IEC 29167.

3 Normative references

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

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

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

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

FIPS PUB 186-4, *Digital Signature Standard (DSS)*[1)]

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
Command (Message)
command that Interrogator sends to Tag with "Message" as parameter

4.2
Certificate
digitally signed statement binding a Public Key to an Identity

Note 1 to entry: The term "Certificate" is also known as "Public Key Certificate".

4.3
double-word
bit string comprised of 32 bits

4.4
entropy
randomness collected by an operating system or application for use in cryptography or other uses that require random data

1) http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf

4.5
isomorphism
one-to-one correspondence between the elements of two sets such that the result of an operation on elements of one set corresponds to the result of the analogous operation on their images in the other set

4.6
Message
part of the Command that is defined by the crypto suite

4.7
Reply (Response)
reply that Tag returns to the Interrogator with “Response” as parameter

4.8
weight
number of non-zero coefficients in the polynomial

4.9
Response
part of the Reply (stored or sent) that is defined by the crypto suite

4.10
X.509
ITU-T standard that defines what information should go into a certificate and describes the format

5 Symbols and abbreviated terms

5.1 Symbols

$xxxx_b$	binary notation of term “xxxx”, where “x” represents a binary digit
$xxxx_h$	hexadecimal notation of term “xxxx”, where “x” represents a hexadecimal digit In this part of ISO/IEC 29167 the bytes in the hexadecimal numbers are presented with the MSB at the left and the LSB at the right. The bit order per byte is also presented with the MSB at the left and the LSB at the right For example in “$ABCDEF_h$” the byte “AB” is the MSB and the byte “EF” is the LSB
||	Concatenation of syntax elements For example “123456_h” || “$ABCDEF_h$” results in “$123456ABCDEF_h$”, where the byte “12” is the MSB and the byte “EF” is the LSB.
$()_x$	x-coordinate of an elliptic curve point
$()^{-1}$	the modular inverse of the polynomial defined within the braces, where the modulus is as indicated in the expression context
b(t)	polynomial basis representation of the curve parameter b (FIPS186-4)
cert(Q)	certificate of the public key Q
c(t)	check polynomial used in the EPIF Format
s(t)	such that $s^2(t) \bmod p(t) = b(t)$ i.e. the square root of b(t) in the field $GF(2^{163})$
E	elliptic curve

Field[a:b]	Selection from a string of bits in Field. Selection ranges from bit a till and including bit b from the bits of the string in Field, whereby Field[0] represents the least significant bit. For example Field[2:0] represents the selection of the three least significant bits of Field
G	base point on the elliptic curve B-163 defined in FIPS 186-4
GF(2)[t]/p(t)	$GF(2^n)$ represented as the field of polynomials modulo a polynomial p(t) of degree n
m(t)	the defining polynomial of the ring GF(2)[t]/m(t) used by the EPIF Format
N	degree of the polynomial p(t)
ϕ	order of the base point on the chosen curve; the bit length of ϕ is considered to be the key size (FIPS186-4 Notation: n)
p(t)	the field polynomial (FIPS186-4)
p'(t)	the defining polynomial of the isomorphic field GF(2)[t]/p'(t) used by the EPIF Format
Polstr()	binary transmission of the polynomial defined within the braces, highest possible degree bit first i.e. including leading zeros; hence if the maximum possible degree of a polynomial is 170, then 171 bits are transmitted i.e. coefficients of terms of degree 170 down to degree 0
Q	private key value of the Tag, a scalar in the range 2..ϕ-2
Q	public key of the Tag is the elliptic curve point; Q = qG
R	random value chosen by the Interrogator in the range 2..ϕ-2 (FIPS186-4)
P	isomorphism from GF(2)[t]/p(t) to GF(2)[t]/p'(t)
Σ	mapping from GF(2)[t]/p'(t) to GF(2)[t]/m(t)
Trace()	function which is a mapping from $GF(2^n)$ to GF(2); the quadratic equation $y^2 + y + \alpha = 0$ has a solution in $GF(2^n)$ when $\mathrm{Trace}(\alpha) = 0$

5.2 Abbreviated terms

ECC	Elliptic Curve Cryptography
EPIF	Error-Protected Isomorphic Field
FIPS	Federal Information Processing Standard
GF(x)	Galois Field (with x elements)
HF	High Frequency (i.e. the frequency band 3MHz to 30 MHz)
NIST	(United States) National Institute of Standards and Technology
toEPIF	function which describes the transformation to the EPIF format

6 Introduction of the ECC-DH crypto suite

6.1 Core functionality

Elliptic curve cryptography has been the basis for many cryptographic protocols for authentication and key agreement. The oldest of these protocols is due to Diffie and Hellmann, and was originally described in Reference [1] as a method of key agreement between two parties performing computations in the multiplicative group of GF(p). This method and its analogous implementation using operations in a group of points on an elliptic curve defined over a finite field, are well known since the inception of public key cryptography, and are not described further here. Instead, attention is drawn to the specific idea of using this protocol for entity authentication, in which:

— One party (the proving entity or "prover") has a static public/private key-pair and a public key certificate which uses a digital signature to bind the public key with the name of the organization that produced the key-pair. In a real life application the certificate (with the digital signature) should be generated by a certification authority.

— The other party (the "verifier"), presents an ephemeral public key to the prover. The prover is required to perform an operation with the private key using this ephemeral public key as an input, and to return the result to the verifier.

— The verifier compares this result with that obtained from his own private key operation, using the ephemeral private key (effectively just a random number) and the prover's public key as an input.

The private key operation corresponds to multiplication of a point by the private key (a scalar). The public key corresponds to the multiplication of the private key (scalar) by a predetermined point on the curve, chosen as a domain parameter of the system. This protocol is illustrated by Figure 1 depicting an RFID system executing (an authentication protocol using) operations on a group of elliptic curve points.

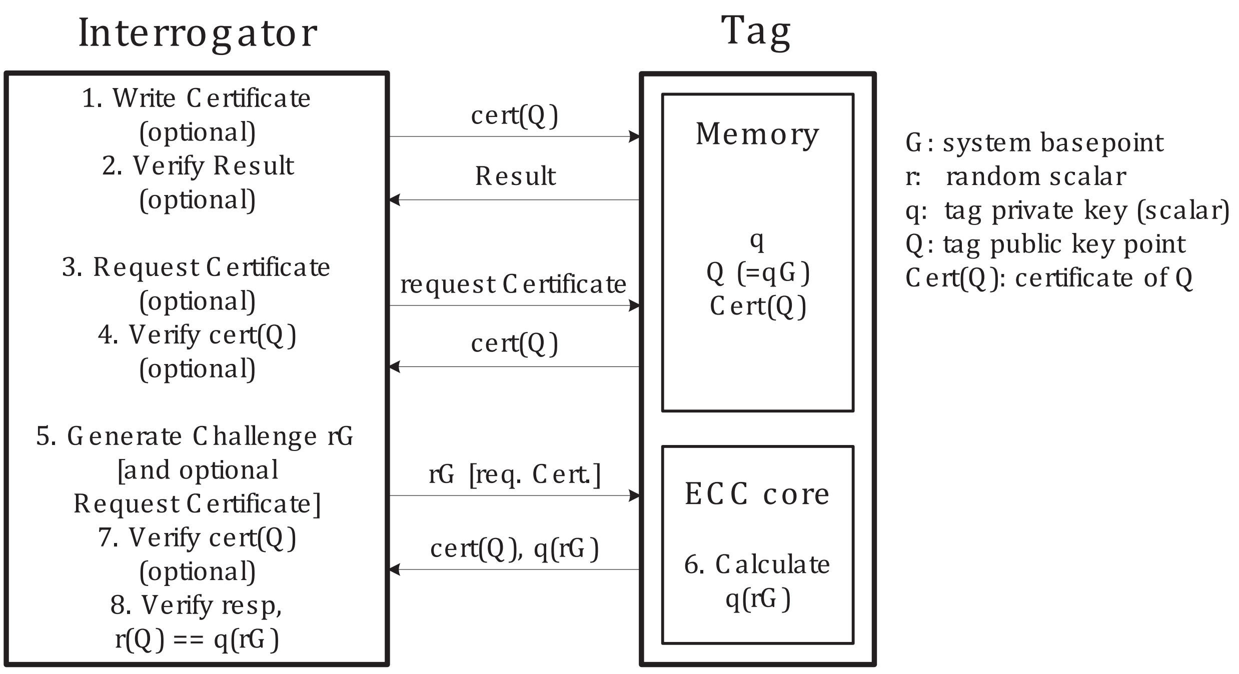

Interrogator
Tag
1. Write Certificate (optional)
2. Verify Result (optional)
3. Request Certificate (optional)
4. Verify cert(Q) (optional)
5. Generate Challenge rG [and optional Request Certificate]
7. Verify cert(Q) (optional)
8. Verify resp, r(Q) == q(rG)
cert(Q)
Result
request Certificate
cert(Q)
rG [req. Cert.]
cert(Q), q(rG)
Memory
q
Q (=qG)
Cert(Q)
ECC core
6. Calculate q(rG)
G: system basepoint
r: random scalar
q: tag private key (scalar)
Q: tag public key point
Cert(Q): certificate of Q

Figure 1 — Elliptic Curve static Diffie-Hellman authentication

In this protocol, the verifier (the Interrogator) first requests the public key certificate from the Tag and verifies if the certificate is valid. Then the Interrogator generates a ephemeral public key r and multiplies the system base point G by this number, and sends the resulting point rG to the prover (in this case the "Tag"). The Tag performs a multiplication of this point by its private key q and returns the result q(rG) to the Interrogator. The Interrogator then verifies that the private key q was really used by checking that r(qG) = = q(rG) where (qG) = Q, the public key point of the Tag. The Interrogator must also verify that this public key is that of a valid Tag, and accordingly the Tag is also required (somewhere within the overall protocol) to present a certificate cert(Q), which is signed by a trusted authority who ensures the authenticity of the public key).

The mathematics of this protocol permits the elliptic curve computations to be performed using only the x-coordinates of points on the chosen curve (i.e. omitting the computations which involve the y-coordinate); this results in a lower requirement for computation, and is a well-known property of Diffie-Hellman protocols, identified in the early days of elliptic curve cryptography.

6.2 Design principles of the crypto suite

The design of the crypto suite is based on the following principles:

— The data exchanges between Tag and Interrogator are designed to minimize the processing and computation requirements on the Tag (for example, by using formats which avoid the need to perform modular inversion on the Tag). The exchange of elliptic curve points between Tag and Interrogator use x-coordinates only to reduce communication overhead and ease computation.

— The data exchanges between the Tag and the Interrogator facilitate simplest possible checking on the Tag that the x-coordinate of the point supplied to the Tag lies on the intended curve (and not on its twist), by sending the pair of values (x, (√b)/x) from the Interrogator to the Tag instead of sending the x and y coordinates of rG; the required check shall then be performed using only two Trace() computations and a single modular multiplication.

— The data exchanges between the Tag and the Interrogator include an integral integrity mechanism intended to facilitate integrity checking of cryptographic computations by both parties. In particular, protection of the computation which is supported. The integrity mechanism is specified in 7.2.

7 Parameter definitions

7.1 Elliptic curve parameters

This part of ISO/IEC 29167 uses Elliptic Curve Cryptography, more particularly it uses elliptic curves E defined over a binary extension field GF(2^n) where E is given by:

$$E: y^2 + xy = x^3 + ax^2 + b$$

The variables x and y in the above equation are the coordinates of an elliptic curve point and the coefficients a and b are elliptic curve parameters. The set of points fulfilling the equation together with a neutral point – the point at infinity – form a group under elliptic curve point addition. Elliptic Curve Cryptography uses a subgroup of this group generated by a generator G of order ϕ.

NOTE In this part of ISO/IEC 29167 the variable “a” has the constant value 1.

The binary extension field is usually represented as GF(2)[t]/p(t) where p(t) is a primitive polynomial of degree n; i.e. the elements of the field are represented as polynomials of degree less than n and operations are performed modulo p(t). For ease of notation the suffix (t) should be dropped when it is clear from the context that an element belongs to GF(2^n).

The authentication protocol defined in this part of ISO/IEC 29167 shall use the Elliptic Curve NIST B-163 whose parameters shall be used as defined in NIST FIPS 186-4.

NOTE please note that NIST FIPS 186–4 uses the variable n to denote the order of the base point (ϕ) and 2^m to denote the size of the binary field (2^n).

7.2 Parameters of the EPIF Format

This part of ISO/IEC 29167 shall use the EPIF representation for points and parameters of the elliptic curve. The EPIF representation is designed on the principle that data exchanges between Tag and Interrogator should minimize processing time and computational requirements on the Tag; yet allow robust implementation.

The EPIF format represents the points and parameters of the elliptic curve as elements of the ring GF(2)[t]/m(t) where m(t) is defined as the product of two irreducible polynomials c(t) and p'(t). The polynomial m(t) is chosen such that it has minimal weight allowing operations modulo m(t) to be implemented very efficiently and the polynomial p'(t) is chosen such that GF(2)[t]/p'(t) is isomorphic to GF(2)[t]/p(t). The polynomials c(t) and p'(t) shall be as defined in normative Annex C.

The transformation to EPIF representation consists of a mapping from GF(2)[t]/p(t) to GF(2)[t]/m(t). This mapping consist of an isomorphism ρ from GF(2)[t]/p(t) to GF(2)[t]/p'(t) followed by a mapping σ from GF(2)[t]/p'(t) to GF(2)[t]/m(t) where the mapping σ is such that the following relations shall hold at all times:

$$\begin{aligned}
&\sigma : GF(2)[t]/p'(t) \text{ a } GF(2)[t]/m(t): \\
&\quad \forall e \in GF(2)[t]/p'(t): \\
&\qquad \sigma(e) \equiv e \quad \bmod p'(t) \\
&\qquad \sigma(e) \equiv 1 \quad \bmod c(t)
\end{aligned}$$

This property is used in the protocol to check the elements for errors.

Given the isomorphism ρ, defined as:

$$\rho : GF(2)[t]/p(t) \text{ a } GF(2)[t]/p'(t)$$

The transformation to EPIF format shall be defined as:

$$\begin{aligned}
&toEPIF : GF(2)[t]/p(t) \text{ a } GF(2)[t]/m(t): \\
&\qquad \forall v \in GF(2)[t]/p(t) : s = toEPIF(v): \\
&\qquad\quad s \in GF(2)[t]/m(t) \\
&\qquad\quad s \equiv \rho(v) \quad \bmod p'(t) \\
&\qquad\quad s \equiv 1 \qquad \bmod c(t)
\end{aligned}$$

Full details on how to compute the toEPIF() transformation are provided in normative Annex C.

7.3 Random number generation

This part of ISO/IEC 29167 requires the Interrogator to generate a random point. This point should contain sufficient entropy to meet the requirements of the application.

8 Crypto suite state diagram

The state diagram for this crypto suite consists of one state only; i.e. Initial State.

After power-up the crypto suite transitions to Initial State. The crypto suite shall remain in Initial State at all times.

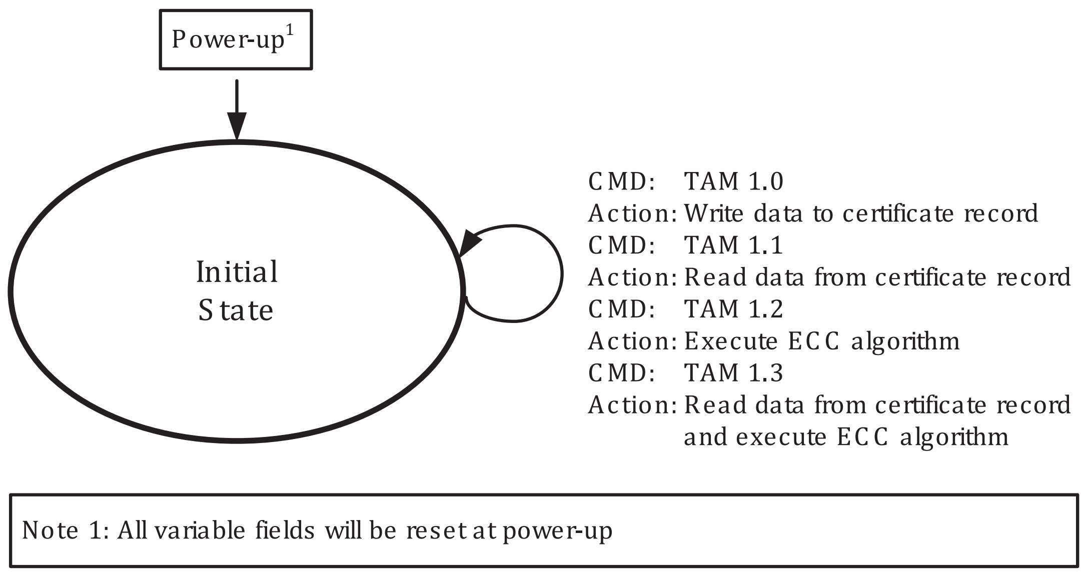

Power-up[1]
Initial State
CMD: TAM 1.0
Action: Write data to certificate record
CMD: TAM 1.1
Action: Read data from certificate record
CMD: TAM 1.2
Action: Execute ECC algorithm
CMD: TAM 1.3
Action: Read data from certificate record and execute ECC algorithm
Note 1: All variable fields will be reset at power-up

Figure 2 — Cryptographic suite state diagram

9 Initialization and resetting

This crypto suite only has one state and does not require initialization. The behaviour on reset is to stay within that state.

Implementations of this part of ISO/IEC 29167 shall ensure that all memory used for intermediate results is cleared after each operation (message-response pair) and after reset.

The crypto suite shall be reset at power-on.

10 Tag Authentication

10.1 Introduction

This part of ISO/IEC 29167 only supports Tag Authentication. In addition this part of ISO/IEC 29167 also supports writing data of a certificate record to the Tag and requesting data of a certificate record from the Tag as additional modes of operation. All functions are implemented using a message-response exchange. This section describes the details of the messages and responses that are exchanged between the Interrogator and Tag.

All message and response exchanges are listed in Table 1.

Table 1 — Tag authentication messages and responses

Command	Function
TAM1.0 message	Write certificate data to the Tag
TAM1.0 response	Return status of writing data
TAM1.1 message	Request certificate data from the Tag
TAM1.1 response	Receive certificate data
TAM1.2 message	Send Interrogator Challenge
TAM1.2 response	Receive cryptographic response
TAM1.3 message	Request certificate record and send Interrogator Challenge
TAM1.3 response	Receive certificate record and cryptographic response

10.2 Message and Response formatting

10.2.1 Concept

Message and Response are part of the security commands that are described in the air interface specification. The “air interface part” of the Tag passes the Message on to the “crypto suite part” of the Tag and returns the Response from the “crypto suite part” to the Interrogator.

10.2.2 Description of Message and Response concept

In TAM1.0 the Interrogator may write a public-key certificate in blocks to the Tag. In TAM1.1 the Interrogator may request a public key certificate in blocks from the Tag. In TAM1.2 the Interrogator sends a challenge to the Tag and the Tag replies with the response that the Interrogator shall verify using a public key. In TAM1.3 the Interrogator request the Tag to send a certificate record and sends a challenge to the Tag. The Tag replies with the content of the requested certificate record and the cryptographic response that the Interrogator shall verify using a public key.

NOTE All message and response pairs may be exchanged in random sequence.

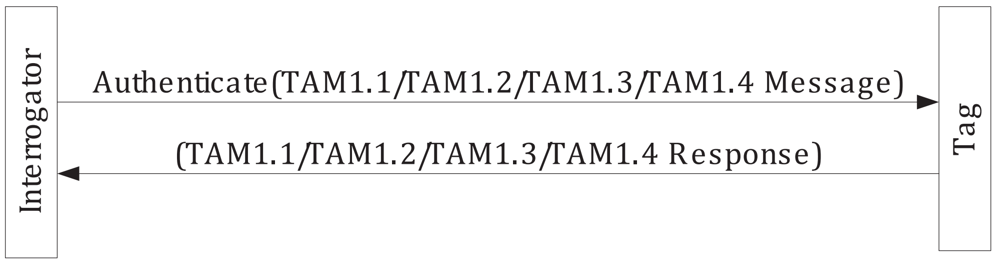

Interrogator
Authenticate(TAM1.1/TAM1.2/TAM1.3/TAM1.4 Message)
(TAM1.1/TAM1.2/TAM1.3/TAM1.4 Response)
Tag

Figure 3 — Message and Response exchange

10.2.3 Transmission order of the data

The transmission order of the data for all Messages and Responses shall be most significant bit first.

Within each Message or Response the most significant word shall be transmitted first.

Within each word the most significant bit shall be transmitted first.

10.2.4 Parsing the Message

For reasons of efficiency the coding for the authentication method and the mode of operation are combined into one field; AuthParam, as defined in Table 2.

Table 2 — Definition of AuthParam flags

Name	Value	Description
AuthParam[1:0]	00_b	Write data for Tag certificate
AuthParam[1:0]	01_b	Request data from Tag certificate
AuthParam[1:0]	10_b	Authenticate Tag
AuthParam[1:0]	11_b	Request certificate record from Tag and authenticate Tag——

The crypto suite shall parse the Messages and process the data based on the value of AuthParam, which is the first parameter of all Messages.

The following sections of this document describe the formatting of Message and Response for Tag Authentication. The Messages shall be distinguished by AuthParam.

If AuthParam = "00_b" the Tag shall parse Message as described in 10.3

If AuthParam = "01_b" the Tag shall parse Message as described in 10.4

If AuthParam = "10_b" then the Tag shall parse Message as described in 10.5.

If AuthParam = "11_b" then the Tag shall parse Message as described in 10.6.

If AuthParam[7:2] ≠ "000000_b" then the Tag shall return a "Not Supported" error condition and shall transition to the **Initial** state.

10.3 TAM1.0

10.3.1 TAM1.0 Message — write certificate data

An Interrogator may use the TAM1.0 Message to write certificate data in blocks (double words) to the Tag. The certificate data may be stored at any appropriate location in the Tag but it shall be accessible through the TAM1.1 Message. The layout of the certificate memory shall be as described in Clause 11.

An Interrogator may request the Tag to store certificate data at any time during the protocol.

To write a block of certificate data to the Tag the Interrogator shall format a TAM1.0 message according to the format described in Table 3.

The fields of the TAM1.0 message shall have the following meaning:

— **AuthParam**: The authentication parameter as described in 10.2.4. It shall be set to 00_h

— **CertNum**: The number of the certificate record.

NOTE The manufacturer may limit the size of each certificate and the number of supported certificates.

— **WordPtr**: A 16-bit pointer indicating the position of a double word within the certificate record.

— **Data**: A 32-bit value to be written to the certificate record.

Table 3 — TAM1.0 Message format

	AuthParam	CertNum	WordPtr	Data
# of bits	8	8	16	32
Description	00_h	number of certificate record	starting address pointer	data to be written

10.3.2 TAM1.0 Response – status of write operation

In case of a write error the Tag shall report a "Memory Write Error" error condition.

In case of other errors the Tag shall report an "Other Error" error condition.

A Tag shall respond to a TAM1.0 Message with a TAM1.0 Response. The TAM1.0 Response shall be empty (zero bits).

10.3.3 Protection of certificate record

A certificate record may be protected by using of the "properties" field of the certificate header, as described in 11.3. After a certificate header is written to the Tag's memory with the "properties" field set to "1_b", the contents of the certificate record shall not be changed.

NOTE The certificate header should be written as the last double word of the certificate record.

10.4 TAM1.1

10.4.1 TAM1.1 Message – request certificate data

An Interrogator may use the TAM1.1 Message to request the Tag to send certificate data in blocks. The certificate data may be stored at any appropriate location in the Tag but it shall be accessible through the TAM1.1 Message.

An Interrogator may request the Tag to send certificate data at any time during the protocol.

To request certificate data from the Tag the Interrogator shall format a TAM1.1 message according to the format described in Table 4.

The fields of the TAM1.1 message shall have the following meaning:

— **AuthParam**: The authentication parameter as described in 10.2.4. It shall be set to 01_h

— **CertNum**: The number of the certificate record.

— **WordPtr**: A 16-bit pointer indicating the position of a double-word within the certificate record.

— **WordCount**: A 16-bit value indicating the number of double words to be read from the certificate record starting from the position indicated by WordPtr. The value of 0000_h shall be used to indicate the full contents of the certificate record starting from the position indicated by WordPtr; i.e. memory shall be read from WordPtr until the end of the current certificate record.

Table 4 — TAM1.1 Message format

	AuthParam	**CertNum**	**WordPtr**	**WordCount**
# of bits	8	8	16	16
Description	01_h	number of certificate record	starting address pointer	number of double words to read

10.4.2 TAM1.1 Response – certificate data

If a WordCount value is provided that exceeds the number of available double words until the end of the current certificate record the Tag shall return a "Memory Overrun" error condition.

In case of a read error the Tag shall report a "Memory Read Error" error condition.

In case of another error the Tag shall report an "Other Error" error condition.

The Tag shall respond to a TAM1.1 Message with a TAM1.1 Response. The TAM1.1 Response shall be formatted according to Table 5.

The field of the TAM1.1 response shall have the following meaning:

— **Certificate data**: The requested part of the certificate memory.

Table 5 — TAM1.1 Response format

	Certificate data
# of bits	variable
Description	data

10.5 TAM1.2

10.5.1 TAM1.2: Message – send Interrogator challenge

An Interrogator shall use the TAM1.2 Message to request the Tag to perform an authentication.

An Interrogator may request the Tag to authenticate at any time during the protocol. To request an authentication the Interrogator shall format a TAM1.2 message according to the format described in Table 6.

The fields of the TAM1.2 message shall have the following meaning:

— **AuthParam**: The authentication parameter as described in 10.2.4. It shall be set to 02_h.

— **IChallenge**: The challenge from the Interrogator to the Tag. The challenge shall consist of two 172-bit values which shall be constructed as describe in Clause 12.

Table 6 — TAM1.2 Message format

	AuthParam	**IChallenge**
# of bits	8	2 * 172 = 344
description	02_h	Interrogator Challenge

10.5.2 TAM1.2 Response – authentication result

In case of another error the Tag shall report an “Other Error” error condition

A Tag shall respond to a TAM1.2 Message with a TAM1.2 Response. The TAM1.2 Response shall be formatted according to Table 7.

The field of the TAM1.2 response shall have the following meaning:

— **TResponse**: The Tag’s response to the Interrogator challenge. It shall consist of two 172-bit values which shall be constructed as described in Clause 12.

Table 7 — TAM1.2 Response format

	TResponse
# of bits	2 * 172 = 344
description	Tag Response

10.6 TAM1.3

10.6.1 TAM1.3: Message – request certificate data and send challenge

An Interrogator shall use the TAM1.3 Message to request the Tag to send an entire certificate record and to perform an authentication.

An Interrogator may request the Tag to send a certificate record and to authenticate at any time during the protocol. To request a certificate record and an authentication the Interrogator shall format a TAM1.3 message according to the format described in Table 8.

The fields of the TAM1.2 message shall have the following meaning:

— **AuthParam**: The authentication parameter as described in 10.2.4. It shall be set to 03_h.

— **CertNum**: Number of the certificate record that is requested by the Interrogator.

— **IChallenge**: The challenge from the Interrogator to the Tag. The challenge shall consist of two 172-bit values which shall be constructed as describe in Clause 12.

Table 8 — TAM1.3 Message format

	AuthParam	**CertNum**	**IChallenge**
# of bits	8	8	2 * 172 = 344
description	03_h	certificate number	Interrogator Challenge

10.6.2 TAM1.3 Response – certificate data and authentication result

In case of a read error the Tag shall report a “Memory Read Error” error condition.

In case of another error the Tag shall report an “Other Error” error condition

The Tag shall respond to a TAM1.3 Message with a TAM1.3 Response. The TAM1.3 Response shall be formatted according to Table 9.

The fields of the TAM1.3 response shall have the following meaning:

— **CertificateData**: The requested certificate data. It shall be formatted according to Table 10 as described in 11.3.

— **TResponse**: The Tags response to the Interrogator challenge. It shall consist of two 172-bit values which shall be constructed as described in Clause 12.

Table 9 — TAM1.3 Response format

	CertificateData	**TResponse**
# of bits	Variable	2 * 172 = 344
description	certificate data	Tag Response

11 Certificate memory

11.1 Concept

The authentication procedure described in this part of ISO/IEC 29167 relies on a certificate for storage and verification of the Tag’s public key. The Interrogator requires such a certificate during the verification procedure. The protocol provides a mechanism to request a certificate from the Tag which therefore should provide storage for it. This storage is called certificate memory and it is designed to hold zero or more certificate records.

Certificate memory may be implemented on the Tag by any appropriate means but if present the data of the certificate records shall be accessible through the TAM1.1 message as described in 10.4. Each certificate record in the memory shall get a certificate number (starting with 0 for the default certificate) and shall start at (virtual) address 0000_h corresponding to the value of WordPtr in the TAM1.1 message. The default certificate may optionally be requested through a TAM1.2 message as described in 10.5.

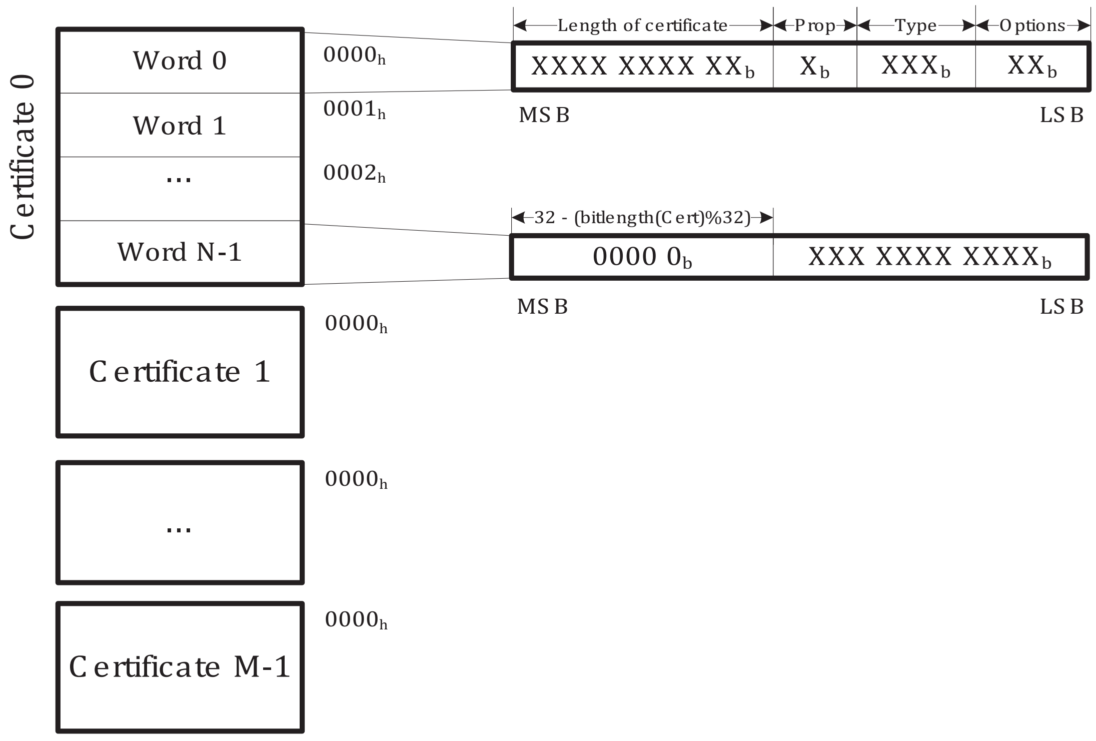
Note A manufacturer may limit the size of the certificates as well as the number of supported certificates.

The structure of the certificate memory shall be as described in 11.2

11.2 Certificate memory structure

If implemented, the certificate memory shall hold one or more certificate records, which have a certificate number starting from 0 (for the default certificate record) and increasing with 1 for every other certificate record. When the length of a certificate record is not a multiple of 32 bits it shall be padded to the MSB side with at most 31 zero bits until the length is a multiple of 32 bits.

The structure of the Certificate Memory is depicted in Figure 4.



NOTE Word 0 of all certificates contains the certificate header

Figure 4 — Certificate Memory Structure

The following structure defines how each certificate record should look like:

```
CertificateRecord = CertificateHeader || PaddedCertificate
PaddedCertificate = [ Padding ] || CertificateData
Padding = 0b || [ Padding ]
```

Note: [x] denotes x is optional

The Header and Value fields of the certificate shall be as described in 11.3.

11.3 Certificate record

The Certificate as defined in the previous section shall consist of a certificate header and the certificate value as defined in Table 10. The fields of the certificate record shall have the following meaning:

— **Header**: A 16-bit value indicating the length, properties, certificate type and options.

— **PaddedCertificate**: The value of the certificate padded with zeroes.

Table 10 — Certificate Record

	Header	PaddedCertificate
# of bits	16	Variable
description	certificate header	padded certificate data

The header shall be formatted as described in Table 11.

The fields of the certificate header shall have the following meaning:

— **Length**: This 10-bit value indicates the length of the certificate in double words (maximum length is $2^{10} \times 32 = 32$K bits).

— **Properties**: Indicates whether the certificate may be overwritten (Properties = 0) or is read-only (Properties = 1).

— **Type**: The type of the certificate. Type shall be set to one of the values listed in Table 12.

— **Options**: The options for the certificate type. They are described in 11.4.

Table 11 — Certificate Header

	Length	Properties	Type	Options
# of bits	10	1	3	2
description	length of the certificate	0 = not write protected 1 = w rite protected	type of the certificate	options of the certificate type

Table 12 — Certificate Types

Name	Value	Description
Type	000_b	Compressed X.509 certificate
Type	001_b	Full X.509 certificate
Type	010_b	Custom certificate
Type	011_b	Custom certificate
Type	100_b	Reserved for future use
Type	101_b	Reserved for future use
Type	110_b	Reserved for future use
Type	111_b	Reserved for future use

11.4 Compressed X.509 certificate

This part of ISO/IEC 29167 uses the X.509 standard.[10] The X.509 standard defines a large number of fields that may be included in a certificate record. However, only a relatively small number of fields are mandatory. Of these mandatory fields only a certain number should vary between certificates for the same application; all other fields shall be set to the fixed values defined for that application. For example, the authentication protocol defined in this part of ISO/IEC 29167 only uses the elliptic curve NIST B-163 so only this curve shall be used for the public key field of the certificate. In addition to the fields that are already fixed by the definition of the protocol it is possible to define fixed and default values for some other mandatory fields of the certificate. This results in only a small number of remaining fields for which the value actually has to be stored in the Tag.

NOTE The signature of the certificate is always calculated over the full X.509 certificate independent whether the stored certificate is compressed.

This section describes a compressed certificate format that stores only the certificate fields for which no default of fixed value can be defined. Normative Annex F describes the procedure to construct a valid X.509 certificate from the fields included in the compressed certificate.

According to Table 12 compressed X.509 certificates shall have certificate type set to 000_b. The content of the compressed certificate depends on the value of the Certificate header options field. When the Certificate header options field is set to 00_b the compressed certificate shall be formatted according to Table 13.

Table 13 — Compressed Certificate format for Options = 00_b

Name	# of bits	Description
SignatureR	236 [a]	Signature, value r
SignatureS	236 [a]	Signature, value s
PubKeyX	168 [b]	Public Key, affine X coordinate
PubKeyY	168 [b]	Public Key, affine Y coordinate
IssuerCN	64	Issuer Common Name
SubjectCN	64	Subject Common Name
CertSN	72	Tag Unique Identifier

[a] SignatureR and SignatureS each consist of 233 bits, but received additional padding to 236 bits to make the size of both combined a multiple of full bytes.

[b] PublicKeyX and PublicKeyY consist of 163 bits each, but need padding to 168 bits to make their size a multiple of full bytes.

When the Certificate header options field is set to 01_b the compressed certificate shall be formatted according to Table 14.

Table 14 — Compressed Certificate format for Options = 01_b

Name	# of bits	Description
SignatureR	236 [a]	Signature, value r
SignatureS	236 [a]	Signature, value s
PubKeyX	168 [b]	Public Key, affine X coordinate
PubKeyY	168 [b]	Public Key, affine Y coordinate
IssuerCN	64	Issuer Common Name
SubjectCN	64	Subject Common Name
SubjectUID	72	Subject Unique Identifier

[a] SignatureR and SignatureS each consist of 233 bits, but received additional padding to 236 bits to make the size of both combined a multiple of full bytes.

[b] PublicKeyX and PublicKeyY consist of 163 bits each, but need padding to 168 bits to make their size a multiple of full bytes.

Both certificate header options result in compressed certificates having a length of 126 bytes excluding the 2-byte header.

NOTE This part of ISO/IEC 29167 fixes the certification method (size of the ECC curve the certification authority needs to generate the signed certificate) for the Compressed X.509 certificate to a binary curve NIST B-233 by specifying lengths of the SignatureR and SignatureS fields. For other types of certificate (full X.509, custom etc.) a manufacturer or application may choose to use another certification method.

11.5 X.509 certificate

Certificate Type 001_b corresponds to a regular full X.509 certificate. The Certificate Type shall be set to 001_b, the Options shall be set to 00_b and the Certificate Value shall be a certificate formatted according to the X.509 standard.[10]

If the Default Certificate is of Certificate Type 001_b its public key shall be the public key of the Tag; i.e. it shall be a public key for NIST curve B-163.

11.6 Custom certificates

Certificate Types 010_b and 011_b correspond to custom certificates. The Options shall be set to 00_b and the Certificate Value shall be a certificate that shall have a custom format.

12 Tag authentication procedure

12.1 Processing steps

The Tag authentication procedure consists of a set of consecutive processing steps which result in the creation of a message, response, error condition or authentication status (result).

Table 15 summarizes the processing steps, where they are performed (Interrogator or Tag) and their possible outcomes.

Table 15 — Processing steps and their possible outcomes

Step	Device	Function	Result
1	Interrogator	IChallenge generation and formatting	Send IChallenge
2	Tag	Parse and verify IChallenge	OK, Error condition
3	Tag	Compute TResponse	Send TResponse
4	Interrogator	Parse and verify TResponse and Certificate	AUTHENTICATED, NON- AUTHENTICATED

12.2 IChallenge generation and formatting

The Interrogator shall generate and format a new random Interrogator challenge IChallenge for each Tag authentication.

To generate the challenge the Interrogator shall perform the following steps:

— The Interrogator shall generate a random value $\mathbf{r}$ in the range 2..$\boldsymbol{\phi}$-2, where $\boldsymbol{\phi}$ is the order of the elliptic curve

— The Interrogator shall retain this random value for use during the cryptographic examination of TResponse from the Tag

— The Interrogator shall perform the elliptic curve point multiplication $\mathbf{rG}$ where $\mathbf{G}$ is the base point of the elliptic curve. Denote the affine x-coordinate of the result with $\mathbf{(rG)_x}$

The Interrogator shall then format IChallenge by performing the following steps:

— Compute: $\mathbf{u = s^*((rG)_x)^{-1} \bmod p}$, where $\mathbf{s^2 = b \bmod p}$ and **b** is the corresponding parameter of the elliptic curve

— Convert $\mathbf{(rG)_x}$ and **u** to EPIF representation according to the procedure described in Annex C.

— Format: $\mathbf{IChallenge = 0_b \,||\, Polstr(toEPIF((rG)_x)) \,||\, 0_b \,||\, Polstr(toEPIF(u))}$

12.3 IChallenge examination

Upon reception of IChallenge the Tag shall examine it in order to establish that it is well-formed and cryptographically valid i.e. that the IChallenge is well-formed and represents an x-coordinate of a point lying in a group generated by the base point on the elliptic curve.

The Tag shall firstly verify that the length of the IChallenge is 344 bits; in the event of an incorrect IChallenge length, then the Tag shall respond with a “Not Supported” error condition.

When the Tag checked that the IChallenge is well-formed, it shall proceed to check the cryptographic validity of the IChallenge. It should do this by the method described in the following steps:

— The Tag shall parse the IChallenge and split the IChallenge into two received polynomials **f(t)** and **g(t)** of equal degree where these have maximum degree 170 and the IChallenge format was $\mathbf{0_b \,||\, Polstr(f(t)) \,||\, 0_b \,||\, Polstr(g(t))}$.

— CVTest1: the Tag should verify that **f(t) * g(t) mod m(t) = toEPIF(s(t))**.

— CVTest2: the Tag should verify that **Trace(f(t))**, computed in the field **GF(2)[t]/p’(t)** is equal to 1.

— CVTest3: the Tag should verify that **Trace(g(t))**, computed in the field **GF(2)[t]/p’(t)** is equal to 0.

— CVTest4: the Tag should verify either that **f(t) mod c(t) = 1** or that **g(t) mod c(t) = 1.**

The Tag should conclude that the IChallenge is cryptographically valid if all verifications of CVTest1, CVTest2, CVTest3 and CVTest4 are successful.

If by the method described above (or by another equivalent method), the Tag establishes that the IChallenge is cryptographically valid, then it shall proceed to calculate and transmit TResponse. Otherwise, it shall respond with a “Cryptographic Error” error condition.

12.4 TResponse generation and formatting

After successful examination of IChallenge the Tag shall generate and format the Tag response TResponse.

To generate the response the Tag shall perform the following steps:

— The Tag shall compute a representation of the x-coordinate $\mathbf{(rG)_x}$ included in the IChallenge by noting that the following relationship holds: $\mathbf{\rho((rG)_x) = f(t) \bmod p'(t)}$.

— The Tag shall compute a projective coordinate representation of a point **G** at this x-coordinate and multiply it by its private key **q**; denote the result with **q(rG)**. The Tag shall extract the x and z coordinates from **q(rG)**. Denote the projective x-coordinate of the result with $\mathbf{(q\ (rG))_x}$ and the projective z-coordinate of the result with $\mathbf{(q(rG))_z}$.

The Tag shall then format TResponse by performing the following steps:

— Convert $\mathbf{(q(rG))_x}$ and $\mathbf{(q(rG))_z}$ **to EPIF** representation

— Format: $\mathbf{TResponse = 0_b \,||\, Polstr(toEPIF((q(rG))_x)) \,||\, 0_b \,||\, Polstr(toEPIF((q(rG))_z))}$

12.5 TResponse examination

Upon reception of TResponse the Interrogator shall examine it in order to establish that it is error-free, well-formed and that it is consistent with the cryptographic response from an authentic Tag.

The output from this examination shall be either the crypto suite state (at the Interrogator) AUTHENTICATED or NON-AUTHETICATED.

Step 1: Error examination

If the Authenticate response from the Tag indicates an error then the output crypto suite state (at the Interrogator) from this examination shall be NON-AUTHENTICATED.

Step 2: Error examination of Tag certificate

If the Interrogator has obtained the public key certificate of the Tag by either method described in 10.4, 10.5 or any other method not described in this part of ISO/IEC 29167 it shall be verified. If the certificate is not valid then the output crypto suite state (at the Interrogator) from this examination shall be NON-AUTHENTICATED.

Step 3: Examination of well-formed TResponse

The Interrogator shall verify that the length of the TResponse is 344 bits. If the Interrogator receives a TResponse whose length is not 344 bits, then the output crypto suite state (at the Interrogator) from this examination shall be NON-AUTHENTICATED.

Step 4: Cryptographic examination

The Interrogator shall conduct the following steps to establish whether the TResponse is consistent with the cryptographic response from an authentic Tag:

— The Interrogator shall split the TResponse into two received polynomials **h(t)** and **k(t)** of equal maximum degree where these have maximum degree 170 and the TResponse format was $\mathbf{0_b \mid\mid Polstr(h(t)) \mid\mid 0_b \mid\mid Polstr(k(t))}$.

— The Interrogator shall compute a representation of the x-coordinate $\mathbf{(q(rG))_x}$ represented in TResponse by noting that the following relationship holds:

$$\boldsymbol{\sigma((q(rG))_x) = (h(t) * k(t)^{-1}) \bmod p'(t)}$$

— The Interrogator shall obtain by an appropriate means a representation of the x-coordinate $\mathbf{Q_x = (qG)_x}$ of the public key of the Tag obtained in step 2 of the verification procedure.

— The Interrogator shall multiply a point at this x-coordinate of the public key by the retained random value **r**, and thereby derive the result $\mathbf{(rQ)_x = (r(qG))_x}$.

— The Interrogator shall compare the x-coordinate $\mathbf{(q(rG))_x}$ represented in the TResponse with the result $\mathbf{(rQ)_x}$ derived from the earlier point multiplication.

If the result of this comparison is that $\mathbf{(q(rG))_x}$ and $\mathbf{(r(qG))_x}$ are equal, then the output crypto suite state (at the Interrogator) from this examination shall be AUTHENTICATED. Otherwise, the output crypto suite state (at the Interrogator) from this examination shall be NON-AUTHENTICATED.

13 Communication

This part of ISO/IEC 29167 does not provide secure communication.

14 Key table and key update

This part of ISO/IEC 29167supports only one public-private key pair to be used with the authentication method described in 10.5 and 10.6. It does not support any other cryptographic operations nor does it support updating the private key.

This part of ISO/IEC 29167 shall therefore not implement a key table.

Annex A
(normative)

Cryptographic suite State transition table

Table A.1 — Cryptographic suite State transition table

Start State	Transition	End State	Result
Initial State	TAM1.0	Initial State	No reply or Error status of Write Certificate is returned
Initial State	TAM1.1	Initial State	Certificate data or an error is returned
Initial State	TAM1.2	Initial State	TResponse or an error is returned
Initial State	TAM1.3	Initial State	Certificate record with TResponse or an error is returned

Annex B
(normative)

Error conditions and error handling

A Tag that encounters an error during the execution of a crypto suite operation may or may not 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 Error Conditions that can result from the operation of this crypto suite. The respective air interface standards need to translate this error condition into an error code for the air interface.

Table B.1 — Tag error conditions

Error-Condition	**Description**
Cryptographic Error	Cryptographic error detected. This triggers a reset.
Memory Overrun	The command attempted to access a non-existent memory location.
Memory Read Error	An error occurred during reading the Tag certificate.
Memory Write Error	An error occurred during writing a Tag certificate.
Not Supported	The requested functionality is not supported by this crypto suite.
Other error	Miscellaneous error.

Annex C
(normative)

Cipher description

C.1 Elliptic curve operations

The operations on the elliptic curve are described in normative reference ISO/IEC 15946-1.

C.2 Error-protected Isomorphic Field (EPIF) Representation

C.2.1 Starting points

The parameters for the EPIF representation have already been explained in Clause 7.

Let $\mathbb{F}$ be the field GF(2)[t]/p(t) and $\mathbb{F}'$ be the field GF(2)[t]/p'(t); ¡ is the ring GF(2)[t]/m(t).

C.2.2 Computing the isomorphism ρ

Let r(t) be one root of p(t) over $\mathbb{F}$; for simplicity take r(t) = t

Let r'(t) be one root of p(t) over $\mathbb{F}'$; the choice is arbitrary, and the definition is based on the first value returned by a search algorithm as follows:

r'(t) = $t^{154} + t^{152} + t^{151} + t^{150} + t^{146} + t^{145} + t^{141} + t^{140} + t^{137} + t^{136} + t^{133} + t^{129} + t^{126} + t^{124} + t^{123} + t^{121} + t^{119} + t^{118} + t^{116} + t^{111} + t^{110} + t^{108} + t^{107} + t^{105} + t^{101} + t^{99} + t^{98} + t^{96} + t^{94} + t^{93} + t^{92} + t^{90} + t^{89} + t^{87} + t^{85} + t^{84} + t^{83} + t^{82} + t^{81} + t^{80} + t^{78} + t^{77} + t^{76} + t^{75} + t^{73} + t^{72} + t^{71} + t^{69} + t^{68} + t^{67} + t^{66} + t^{59} + t^{58} + t^{56} + t^{54} + t^{52} + t^{51} + t^{49} + t^{47} + t^{45} + t^{44} + t^{41} + t^{38} + t^{37} + t^{35} + t^{34} + t^{33} + t^{32} + t^{29} + t^{28} + t^{27} + t^{24} + t^{22} + t^{16} + t^{15} + t^{14} + t^{11} + t^{10} + t + 1$

Galois theory indicates that the mapping r → r' gives us an isomorphism ρ() from $\mathbb{F}$ to $\mathbb{F}'$.

To compute the isomorphism in practice, note that $\rho\left(r^{\phi}\right)=\rho\left(r\right)^{\phi}=r'^{\phi}$.

Thus the following formula to be used to compute $\rho\left(f\right)$, where $f=\sum_{i=0}^{162}a_i t^i$, is:

$$\rho\left(f\right)=\rho\left(\sum_{i=0}^{162}a_i t^i\right)=\sum_{i=0}^{162}a_i\rho\left(t^i\right)=\sum_{i=0}^{162}a_i r'^i \bmod p'=\sum_{i=0}^{162}a_i\left(r'^i \bmod p'\right)$$

which shall be treated a matrix multiplication when the polynomials are regarded as column vectors in GF(2^{163}) e.g. f as $(a_0, a_1, \ldots, a_{162})^t$.. Denote this matrix

M = (1, r', r'^2, ..., r'^{162}), where the entries are column vectors.

C.2.3 Computing the mapping σ

To map values into EPIF (into the ring ¡) one shall regard $\rho(f)$ also as an element of the ring ¡ , and multiply by A(t) and add k(t), where

A(t) = 1 mod p'(t)	A(t) = 0 mod c(t)
k(t) = 0 mod p'(t)	k(t) = 1 mod c(t)

That way the result (in ¡) would stay the same modulo p', but would be 1 modulo c.

Treating the multiplication by A(t) also as a matrix multiplication mapping from GF(2^{163}) to GF(2^{171}) multiplying by a matrix A. The addition of k(t) can also be regarded as a vector addition with vector **k**.

C.2.4 Computing the mapping toEPIF

Accordingly, toEPIF (f) = A*M***v** + **k** = M'***v** + **k**, where **v** is the vector representing f and M' = A*M, which shall be pre-computed.

C.2.5 Values

Table C.1 — Parameter values for the toEPIF transformation

Parameter	Description
p(t); according to FIPS186–4	$t^{163} + t^{7} + t^{6} + t^{3} + 1$
m(t); the trinomial	$t^{171} + t^{70} + 1$
c(t); check polynomial	$t^{8} + t^{7} + t^{5} + t^{4} + t^{3} + t^{2} + 1$
p‘(t); polynomial that meets m(t) = p'(t) * c(t)	$t^{163} + t^{162} + t^{161} + t^{158} + t^{155} + t^{154} + t^{153} + t^{152} + t^{151} + t^{150} + t^{149} + t^{148} + t^{146} + t^{142} + t^{141} + t^{140} + t^{138} + t^{136} + t^{134} + t^{132} + t^{131} + t^{126} + t^{124} + t^{123} + t^{120} + t^{119} + t^{116} + t^{110} + t^{109} + t^{107} + t^{106} + t^{105} + t^{103} + t^{102} + t^{101} + t^{94} + t^{91} + t^{89} + t^{88} + t^{86} + t^{78} + t^{77} + t^{76} + t^{73} + t^{70} + t^{69} + t^{68} + t^{67} + t^{66} + t^{65} + t^{64} + t^{63} + t^{62} + t^{60} + t^{56} + t^{55} + t^{54} + t^{52} + t^{50} + t^{48} + t^{46} + t^{45} + t^{40} + t^{38} + t^{37} + t^{34} + t^{33} + t^{30} + t^{24} + t^{23} + t^{21} + t^{20} + t^{19} + t^{17} + t^{16} + t^{15} + t^{8} + t^{5} + t^{3} + t^{2} + 1$

Annex D
(informative)

Examples ECC cryptographic protocol

D.1 Example 1

Setup

```
NIST B-163

Base point     G = [ X:3f0eba16286a2d57ea0991168d4994637e8343e36
                     Y:d51fbc6c71a0094fa2cdd545b11c5c0c797324f1
                     ]
```

Tag setup

```
Private key    q = 20925cc492416e315bbe2ae590e627cbdac3e58db

Public key     Q = [ X:2368c1792808ce7e89821013e15947185485bfe46
                     Y:52047c7bfa0f24fd16e6cddd11eb650be1d51f84d
                     ]
```

Protocol

```
Random scalar  r = 6886a5d2978142da2dc5955739a3469005936793

Challenge      R = r*G = [ X:2f9b76d744d3d346e40097df63885d8b9495f7f2
                           Y:15d2cf11680baeb7808f6764a750e7df62c10a41f
                           ]
```

Transmit the following challenge

```
toEPIF(rG)_X   = 329596f6478361d46e273aabce0a18c3d5cdc5f4110

toEPIF(u)      = 61ede92b62887baa97c3903b5d3be67f1caccc08a83
```

Tag calculation

```
q*R            = [ X:ef67a9c726bb5649f4fd06ae0b7ff7930ccd0d86
                   Y:49dbccd3ef6576d58a988106fb07eceae3f6f1944
                   ]
```

Answer of the Tag

```
toEPIF(qR)_X   = 242413cf611533e1cd793ac8815933d8e85b65b0701

toEPIF(qR)_Z   = 07a4dd9c88db874ebef827f0b1feddf103df0428b55
```

Check value to compare with return result (use x coordinate)

```
r*Q         = [ X:ef67a9c726bb5649f4fd06ae0b7ff7930ccd0d86
                Y:49dbccd3ef6576d58a988106fb07eceae3f6f1944
                ]
```

D.2 Example 2

Setup

```
NIST B-163

Base point  G = [ X:3f0eba16286a2d57ea0991168d4994637e8343e36
                  Y:d51fbc6c71a0094fa2cdd545b11c5c0c797324f1
                  ]
```

Tag setup

```
Private key  q = e56a2d80da904fa46d658f2abcae1f110f8ac3f

Public key   Q = [ X:63c8ead8e9598c50acb2a7d493786d8dcea96bf7a
                   Y:1a573083a45e23037c0fee6a641e37ae3f3f547e9
                   ]
```

Protocol

```
Random scalar  r = 422dcd927c4a4f05b0353b21b77e39803f664606 (seed = 23432)

Challenge      R = r*G = [ X:1dff822e7c729994888112d599943f3b5e84123a1
                           Y:11cb1e3be94757d9acf8d06f3404388ab3d1983f1
                           ]
```

Transmit the following challenge

```
toEPIF(rG)_X   = 40f5f63fe0fe4694dea982ecda94de9ef797a7a8712

toEPIF(u)      = b4f145998dfa31ff7a9180a3e08396fe6f5df65983
```

Tag calculation

```
q*R            = [ X:37ab4900eb363a1f85c8c39accb623306337d7564
                   Y:40ce9e52f8721a083b7ce3945f76dc788d8bf7f39
                   ]
```

Answer of the Tag

```
toEPIF(qR)_X   = 5b7147d69fed36e272ec3401594223ff63b3d8ef4a

toEPIF(qR)_Z   = 1
```

Check value to compare with return result (use x coordinate)

```
r*Q            = [ X:37ab4900eb363a1f85c8c39accb623306337d7564
                   Y:40ce9e52f8721a083b7ce3945f76dc788d8bf7f39
                   ]
```

Annex E
(normative)

Air Interface Protocol specific information

E.1 General

E.1.1 Concept of exchanging Message and Response

For the implementation of this crypto suite an air interface protocol shall support security commands that allow the exchange of data between the Interrogator and the Tag that has this crypto suite implemented. The security command contains a message with parameters for the crypto suite. The reply of the Tag contains a response with the data that is returned by the crypto suite. An example of such data exchange for this crypto suite is depicted in Figure E.1.

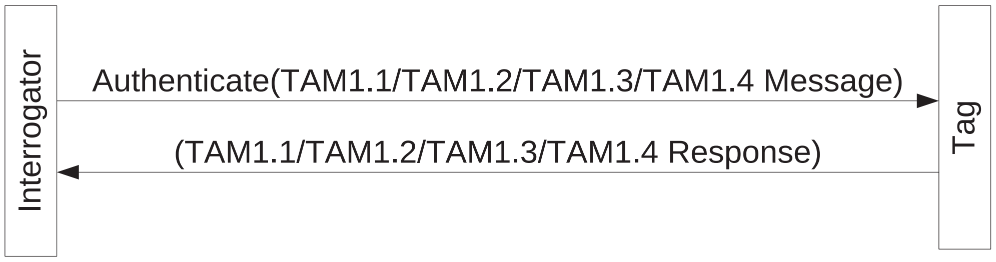

Interrogator
Authenticate(TAM1.1/TAM1.2/TAM1.3/TAM1.4 Message)
(TAM1.1/TAM1.2/TAM1.3/TAM1.4 Response)
Tag

Figure E.1 — Message and Response exchange

The crypto suites that are defined by ISO/IEC 29167 can be defined by their Crypto Suite Identifier (CSI). According to ISO/IEC 29167-1 the CSI for this crypto suite shall be defined as the 6-bit value “000010_2”

E.1.2 Supported Security Services

Table E.1 shows the security services that are supported by this part of ISO/IEC 29167.

Table E.1 — Security Services

Security Services	**Method**	**Mandatory, optional, Prohibited, or not supported**[a]
Authentication		Mandatory
Tag authentication (TA)		Mandatory
Interrogator authentication (IA)		Not supported
Mutual Authentication (MA)		Not supported
Communication		Not supported
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

[a] A crypto suite shall identify for each security service above and method if it is mandatory, optional, or prohibited

Table E.1 *(continued)*

Security Services	**Method**	**Mandatory, optional, Prohibited, or not supported**[a]
	Secure authenticated communication (Interrogator = > Tag)	Not supported
[a] A crypto suite shall identify for each security service above and method if it is mandatory, optional, or prohibited		

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

E.2.1 ISO/IEC 18000-63 Protocol Commands

A crypto suite supporting ISO/IEC 18000-63 shall fulfill the protocol security command requirements as defined in this section.

NOTE Optional choices shall be accepted for 1-to-1 communication. Reason: Since the Tag is singulated and the TID is known supported options can be derived from it.

a) The *Authenticate* command or *Challenge* command shall be supported for TAM1.2 and TAM1.3.

b) The maximum execution time for an *Authenticate* or *Challenge* Command containing a TAM1.0, TAM1.1, TAM1.2 or TAM1.3 payload shall be below 500 ms.

   NOTE If the executing time exceeds 20 ms, the Tag is expected to use the In-Process reply in accordance with the air interface standard to keep the Interrogator informed about the ongoing processing.

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

d) The Tag shall not support sending the contents of the ResponseBuffer in the reply to an ACK command.

e) The Tag shall support sending the contents of the ResponseBuffer in the reply to a *ReadBuffer* command

f) The Tag may support a security timeout following a crypto error. The length of the security timeout shall be < 200 ms.

g) The *Authenticate* command or the *Challenge* command or both commands shall be supported for all authentication methods.

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

i) For each Error Condition defined in the crypto suite:

   — The Tag shall transition to the **arbitrate** state

   — The Tag shall send an Error Code in case of a transition to the arbitrate state.

j) The Tag shall remain in its current state after a Tag Authentication.

k) This crypto suite does not support any encapsulation method.

E.2.2 Security commands in ISO/IEC 18000-63

In ISO/IEC 18000-63 the message to execute Write Certificate, Request Certificate or Tag authentication shall be transmitted to the Tag with the *Authenticate or Challenge* command. The air interface shall return the response, either it shall be backscattered immediately after the command or it shall be stored in the ResponseBuffer, from where it shall be returned to the Interrogator with the *ReadBuffer* command.

NOTE Information about the *Authenticate*, *Challenge* and *ReadBuffer* command and the ResponseBuffer can also be found in Reference [3].

ISO/IEC 18000-63 specifies an 8-bit CSI. For implementation of this part of ISO/IEC 29167 in ISO/IEC 18000-63 the CSI shall be expanded to the 8-bit value “02_h”.

E.2.3 Implementation of crypto suite error conditions in ISO/IEC 18000-63

This crypto suite specifies error conditions when the authentication is not successful. The error conditions of the crypto suite shall be returned to the Interrogator as error codes for the air interface. Table E.2 shows the conversion of Error Conditions in the crypto suite to ISO/IEC 18000-63 error codes.

Table E.2 — Implementation of crypto suite error conditions as Tag error codes

Error-Condition	**Description**	**18000–63 Error Code**	**Error-Code Name**
Other error	Miscellaneous error	00000101_b	Crypto Suite error
Memory Overrun	The command attempted to access a non-existent memory location	00000011_b	Memory overrun
Not Supported	The requested functionality is not supported by this crypto suite	00000001_b	Not supported
Cryptographic Error	Cryptographic error detected. This triggers a reset	00000101_b	Crypto Suite error
Memory Write Error	An error occurred during writing the Tag certificate	00000100_b	Memory locked

E.2.4 Key Properties

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 crypto 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 that the key shall only be used for Tag authentication.

E.2.5 Compressed certificate

Section 11.4 defines a compressed certificate format and Annex F defines a procedure to reconstruct an X.509 Certificate from this compressed certificate. This reconstruction procedure assigns default values to certain fields of the X.509 certificate depending on the fields provided in the Compressed Certificate. The default values are air interface specific and are defined in this section.

The default value of the Tag Unique Identifier (SubjectUIDdefault) shall be equal to the following 9 bytes of the TID which is defined in the Tag Data Standard.[4]

```
SubjectUIDefault = TAG MDID[11:0] (12 bits) || TAG MODEL NUMBER[11:0] (12 bits) || Serial
Number Segment[47:0] (48 bits)
```

Annex F
(normative)

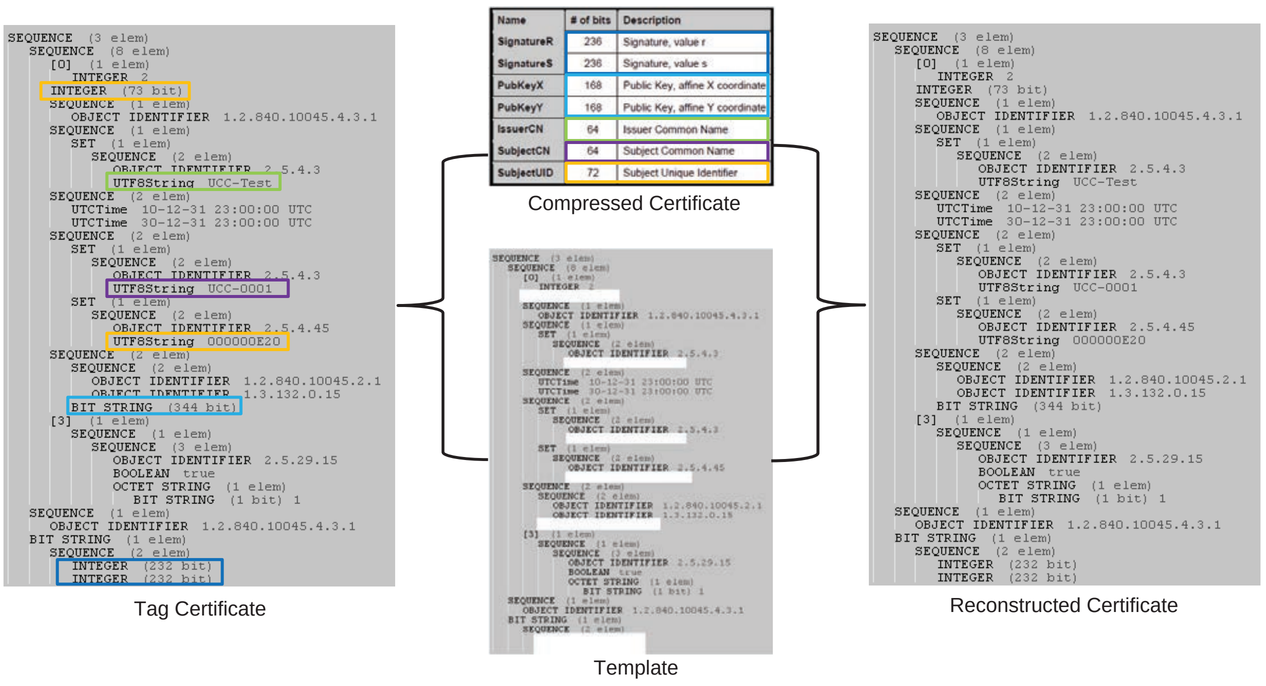
Reconstruction of X.509 Certificate

This appendix describes the procedure to reconstruct an X.509 certificate from the fields provided in the compressed certificate as defined in 11.4. It shall start by defining the structure of the X.509 certificate and then describe the contents of its fields.

F.1 Introduction to the reconstruction process

The public key certificate for a Tag is built during the personalization process and at that point separated into a “compressed certificate” part and a “template” part. The “compressed certificate” part contains the individual Tag information that may vary for every individual Tag and shall therefore be stored in a certificate record on the Tag. The content of the “template” part is identical for all Tags that belong to the same application and should be stored for the use in the reconstruction process later. The verification process requires the original full X.509 certificate that can be reconstructed by merging the Tag’s “compressed X.509 certificate” part with the “template” part.

The compressed X.509 certificate therefore only contains the variable fields of the Tag’s full X.509 certificate and is stored in then Tag’s certificate record. The full X.509 certificate can be reconstructed by combining the compressed X.509 certificate with the related X.509 template. Figure F.1 shows the reconstruction process:



```
SEQUENCE  (3 elem)
   SEQUENCE  (8 elem)
      [0]  (1 elem)
         INTEGER  2
      INTEGER  (73 bit)
      SEQUENCE  (1 elem)
         OBJECT IDENTIFIER  1.2.840.10045.4.3.1
      SEQUENCE  (1 elem)
         SET  (1 elem)
            SEQUENCE  (2 elem)
               OBJECT IDENTIFIER  2.5.4.3
               UTF8String  UCC-Test
      SEQUENCE  (2 elem)
         UTCTime  10-12-31 23:00:00 UTC
         UTCTime  30-12-31 23:00:00 UTC
      SEQUENCE  (2 elem)
         SET  (1 elem)
            SEQUENCE  (2 elem)
               OBJECT IDENTIFIER  2.5.4.3
               UTF8String  UCC-0001
         SET  (1 elem)
            SEQUENCE  (2 elem)
               OBJECT IDENTIFIER  2.5.4.45
               UTF8String  000000E20
      SEQUENCE  (2 elem)
         SEQUENCE  (2 elem)
            OBJECT IDENTIFIER  1.2.840.10045.2.1
            OBJECT IDENTIFIER  1.3.132.0.15
         BIT STRING  (344 bit)
      [3]  (1 elem)
         SEQUENCE  (1 elem)
            SEQUENCE  (3 elem)
               OBJECT IDENTIFIER  2.5.29.15
               BOOLEAN  true
               OCTET STRING  (1 elem)
                  BIT STRING  (1 bit) 1
   SEQUENCE  (1 elem)
      OBJECT IDENTIFIER  1.2.840.10045.4.3.1
   BIT STRING  (1 elem)
      SEQUENCE  (2 elem)
         INTEGER  (232 bit)
         INTEGER  (232 bit)
```

Tag Certificate

Name	# of bits	Description
SignatureR	236	Signature, value r
SignatureS	236	Signature, value s
PubKeyX	168	Public Key, affine X coordinate
PubKeyY	168	Public Key, affine Y coordinate
IssuerCN	64	Issuer Common Name
SubjectCN	64	Subject Common Name
SubjectUID	72	Subject Unique Identifier

Compressed Certificate

```
SEQUENCE  (3 elem)
   SEQUENCE  (8 elem)
      [0]  (1 elem)
         INTEGER  2

      SEQUENCE  (1 elem)
         OBJECT IDENTIFIER  1.2.840.10045.4.3.1
      SEQUENCE  (1 elem)
         SET  (1 elem)
            SEQUENCE  (2 elem)
               OBJECT IDENTIFIER  2.5.4.3

      SEQUENCE  (2 elem)
         UTCTime  10-12-31 23:00:00 UTC
         UTCTime  30-12-31 23:00:00 UTC
      SEQUENCE  (2 elem)
         SET  (1 elem)
            SEQUENCE  (2 elem)
               OBJECT IDENTIFIER  2.5.4.3

         SET  (1 elem)
            SEQUENCE  (2 elem)
               OBJECT IDENTIFIER  2.5.4.45

      SEQUENCE  (2 elem)
         SEQUENCE  (2 elem)
            OBJECT IDENTIFIER  1.2.840.10045.2.1
            OBJECT IDENTIFIER  1.3.132.0.15

      [3]  (1 elem)
         SEQUENCE  (1 elem)
            SEQUENCE  (3 elem)
               OBJECT IDENTIFIER  2.5.29.15
               BOOLEAN  true
               OCTET STRING  (1 elem)
                  BIT STRING  (1 bit) 1
   SEQUENCE  (1 elem)
      OBJECT IDENTIFIER  1.2.840.10045.4.3.1
   BIT STRING  (1 elem)
      SEQUENCE  (2 elem)
```

Template

```
SEQUENCE  (3 elem)
   SEQUENCE  (8 elem)
      [0]  (1 elem)
         INTEGER  2
      INTEGER  (73 bit)
      SEQUENCE  (1 elem)
         OBJECT IDENTIFIER  1.2.840.10045.4.3.1
      SEQUENCE  (1 elem)
         SET  (1 elem)
            SEQUENCE  (2 elem)
               OBJECT IDENTIFIER  2.5.4.3
               UTF8String  UCC-Test
      SEQUENCE  (2 elem)
         UTCTime  10-12-31 23:00:00 UTC
         UTCTime  30-12-31 23:00:00 UTC
      SEQUENCE  (2 elem)
         SET  (1 elem)
            SEQUENCE  (2 elem)
               OBJECT IDENTIFIER  2.5.4.3
               UTF8String  UCC-0001
         SET  (1 elem)
            SEQUENCE  (2 elem)
               OBJECT IDENTIFIER  2.5.4.45
               UTF8String  000000E20
      SEQUENCE  (2 elem)
         SEQUENCE  (2 elem)
            OBJECT IDENTIFIER  1.2.840.10045.2.1
            OBJECT IDENTIFIER  1.3.132.0.15
         BIT STRING  (344 bit)
      [3]  (1 elem)
         SEQUENCE  (1 elem)
            SEQUENCE  (3 elem)
               OBJECT IDENTIFIER  2.5.29.15
               BOOLEAN  true
               OCTET STRING  (1 elem)
                  BIT STRING  (1 bit) 1
   SEQUENCE  (1 elem)
      OBJECT IDENTIFIER  1.2.840.10045.4.3.1
   BIT STRING  (1 elem)
      SEQUENCE  (2 elem)
         INTEGER  (232 bit)
         INTEGER  (232 bit)
```

Reconstructed Certificate

Figure F.1 — Certificate Compression and Reconstruction

F.2 X.509 certificate structure

F.2.1 Certificate construction

An X.509 device certificate according to ITU-T X.509[10] and IETF RFC 5280[12] shall be constructed. It shall be composed of three required fields: tbsCertificate, signatureAlgorithm and signatureValue, described in ASN.1 notation as defined by ISO/IEC 8824.[13] The following ASN.1 syntax describes the structure of the certificate whereby all possible fields are listed. F.3 defines which fields have a fixed or default value and which fields are reconstructed from the compressed certificate.

```
Certificate     ::=  SEQUENCE {
            tbsCertificate       TBSCertificate,
            signatureAlgorithm   AlgorithmIdentifier,
            signatureValue       BIT STRING }

TBSCertificate  ::= SEQUENCE {
            version              [0]  EXPLICIT Version,
            serialNumber              CertificateSerialNumber,
            signature                 AlgorithmIdentifier,
            issuer                    Name,
            validity                  Validity,
            subject                   Name,
            subjectPublicKeyInfo      SubjectPublicKeyInfo,
            issuerUniqueID       [1]  IMPLICIT UniqueIdentifier OPTIONAL,
            subjectUniqueID      [2]  IMPLICIT UniqueIdentifier OPTIONAL,
            extensions           [3]  EXPLICIT Extensions OPTIONAL
 }

Version     ::=  INTEGER  {  v1(0), v2(1), v3(2)  }

CertificateSerialNumber    ::=  INTEGER

Validity    ::=  SEQUENCE {
            notBefore       Time,
            notAfter        Time }

Time        ::=  CHOICE {
            utcTime         UTCTime,
            generalTime     GeneralizedTime }

SubjectPublicKeyInfo  ::=  SEQUENCE  {
            algorith             AlgorithmIdentifier,
            subjectPublicKey     BIT STRING  }

Extensions      ::=  SEQUENCE SIZE (1..MAX) OF Extension

Extension       ::=  SEQUENCE  {
            extnID          OBJECT IDENTIFIER,
            critical        BOOLEAN DEFAULT FALSE,
            extnValue       OCTET STRING
                            -- contains the DER encoding of an ASN.1 value
                            -- corresponding to the extension type identified
                            -- by extnID }
```

Figure F.2 — X.509 Certificate structure

F.2.2 Extension fields

This section lists possible extension fields for the certificate. F.3 defines which fields are present in the certificate, which fields have a fixed or default value and which fields are reconstructed from the compressed certificate.

F.2.2.1 Authority key identifier

The authority key identifier extension provides a means of identifying the public key corresponding to the private key used to sign a certificate. This extension is used where an issuer has multiple signing keys (either due to multiple concurrent key pairs or due to changeover). The identification may be based on either the key identifier (the subject key identifier in the issuer's certificate) or the issuer name and serial number.

```
AuthorityKeyIdentifier     ::=  SEQUENCE {
          keyIdentifier             [0]  KeyIdentifie             OPTIONAL,
          authorityCertIssuer       [1]  GeneralNames             OPTIONAL,
          authorityCertSerialNumber [2]  CertificateSerialNumber  OPTIONAL  }

KeyIdentifier   ::=  OCTET STRING
```

Figure F.3 — Authority key identifier extension field

F.2.2.2 Key usage

The key usage extension defines the purpose (e.g. encipherment, signature, certificate signing) of the key contained in the certificate. The usage restriction should be employed when the usage of a key that could be used for more than one operation is to be restricted.

```
AuthorityKeyIdentifier     ::=  SEQUENCE {
          keyIdentifier             [0]  KeyIdentifie             OPTIONAL,
          authorityCertIssuer       [1]  GeneralNames             OPTIONAL,
          authorityCertSerialNumber [2]  CertificateSerialNumber  OPTIONAL  }

KeyIdentifier   ::=  OCTET STRING
```

Figure F.4 — Key usage extension field

The key usage extension field shall be fixed to the value digitalSignature(0) meaning that the private key must not be used for different purposes than device authentication.

F.3 Certificate fields

F.3.1 tbsCertificate

F.3.1.1 Basic content of tbsCertificate

The `tbsCertificate` field contains the names of the subject and issuer, a public key associated with the subject, a validity period, and other associated information. All fields listed in Figure F.2 are allowed but some of them are not mandatory. The supported extension fields are described later in more detail.

F.3.1.2 Version (FIXED value)

This field describes the version of the encoded certificate. It shall be set to 02_h (version 3), which allows the inclusion of extensions and/or unique identifier fields.

```
Version::= INTEGER { v3(2) }
```

The DER encoding of this field is constant:

```
A0h 03h 02h 01h 02h
```

F.3.1.3 Serial number (DEFAULT value)

The serial number shall be a positive integer assigned by the CA to each certificate. It shall be unique for each certificate issued by a given CA (i.e. the issuer name and serial number identify a unique certificate).

This field is defined to be of type INTEGER.

```
CertificateSerialNumber::= INTEGER
```

The serial number field shall contain an 10 byte value uniquely identifying the certificate as depicted in Table F.1 — Certificate serial number. It shall be composed of 1 byte certificate count number (CertCount) and a 9 byte serialization (CertSN). When multiple certificates are issued for the same Tag (or public key) they should have the same value of CertSN, but in that case they shall have different values for CertCount. The first certificate issued for a Tag shall have CertCount set to 01_h and it shall be incremented for each subsequent certificate that is issued for that Tag. CertCount shall be at most $7F_h$.

Table F.1 — Certificate serial number

	CertCount	**CertSN**
# of bits	8	72
description	Certificate counter	Unique serialization

For compressed certificates CertCount shall be set to 01_h.

When the options field in the compressed certificate header is set to 00_b the value of CertSN shall be taken from the compressed certificate. Otherwise CertSN shall be set to SubjectUID as defined in F.3.1.7.

Example: when the CertSN in the compressed certificate is set to XX XX XX XX XX XX XX XX XX_h the Certificate Serial Number shall be 01 XX XX XX XX XX XX XX XX XX_h and its DER encoding:

```
02h 0Ah
     01h XXh XXh XXh XXh XXh XXh XXh XXh XXh
```

F.3.1.4 Signature algorithm (FIXED value)

This field contains the algorithm identifier for the algorithm used by the CA to sign the certificate. This field shall contain the same algorithm identifier as the signatureAlgorithm field in F.3.2.

This field is defined to be of type `AlgorithmIdentifier`:

```
AlgorithmIdentifier::= SEQUENCE {
           algorithm            OBJECT IDENTIFIER,
           parameters           ANY DEFINED BY algorithm OPTIONAL}
```

The algorithm field shall be set to ecdsa-with-SHA224 (OID 1.2.840.10045.4.3.1) and the parameter field shall be absent.

The DER encoding of this field is

```
30h 0Ah
      06h 08h
           2Ah 86h 48h CEh 3Dh 04h 03h 01h
```

F.3.1.5 Issuer

The issuer field identifies the entity that has signed and issued the certificate. The issuer field shall contain a non-empty distinguished name (DN). It shall be restricted to just the Common Name, with fixed width of 8 bytes.

This field is defined to be of X.501 type Name:

```
Name::= CHOICE { – only one possibility for now –
           rdnSequence RDNSequence }
    RDNSequence:: = SEQUENCE OF RelativeDistinguishedName
    RelativeDistinguishedName:: =
         SET SIZE (1..MAX) OF AttributeTypeAndValue
    AttributeTypeAndValue:: = SEQUENCE {
         type AttributeType,
         value AttributeValue }
    AttributeType:: = OBJECT IDENTIFIER
         AttributeValue:: = ANY – DEFINED BY AttributeType
```

This field shall contain only one relative distinguished name, having only a Common Name (OID 2.5.4.3) attribute. The common name shall be taken from the IssuerCN field in the compressed certificate.

Example: When the IssuerCN in the compressed certificate is set to XX_h XX_h XX_h XX_h XX_h XX_h XX_h XX_h the DER encoding of the Issuer field is:

```
30h 13h
      31h 11h
      30h 0Fh
            06h 03h 55h 04h 03h
            0Ch 08h XXh XXh XXh XXh XXh XXh XXh XXh
```

F.3.1.6 Validity (DEFAULT value)

The validity field holds the validity period of the certificate. It shall be used by the Interrogator to determine if the Tag certificate is still valid provided it has access to a trusted time source.

The used date format is the standard ASN.1 type GeneralizedTime, expressed in Greenwich Mean Time including zero-valued seconds.

The field type is defined as:

```
Validity ::= SEQUENCE {
          notBefore GeneralizedTime,
          notAfter GeneralizedTime }
```

The compressed certificate does not include validity information so a default values are defined for the “Not Before” and “Not After” dates of the validity of a compressed certificate.

The validity of the certificate shall be set to the following values:

— “Not Before” Date: 01/01/2011 – 00:00:00

— “Not After” Date: 01/01/2031 – 00:00:00

The DER encoding of the default validity is:

```
30h 1Eh
      17h 0Dh
            31h 30h 31h 32h 33h 31h 32h 33h 30h 30h 30h 30h 5Ah
```

```
        17h 0Dh
                33h 30h 31h 32h 33h 31h 32h 33h 30h 30h 30h 30h 5Ah
```

F.3.1.7 Subject (DEFAULT value)

The Subject shall be identified by a sequence containing a Common Name and a Unique Number.

The Common Name (OID 2.5.4.3) is defined as type DirectoryString, with the following ASN.1 definition:

```
DirectoryString { INTEGER : maxSize }  ::= CHOICE {
      teletexString           TeletexString (SIZE (1..maxSize)),
      printableString         PrintableString (SIZE (1..maxSize)),
      bmpString               BMPString (SIZE (1..maxSize)),
      universalString         UniversalString (SIZE (1..maxSize)),
      uTF8String              UTF8String (SIZE (1..maxSize)) }
```

For compressed certificates, the Common Name in the subject shall be restricted to only the UTF8String choice, with a fixed encoded size of 8 octets. It has no default value and its value shall be taken from the compressed certificate field SubjectCN.

The Unique Identifier (OID 2.5.4.45) is defined as type UniqueIdentifier, with the following ASN.1 definition:

```
UniqueIdentifier:: = BIT STRING
```

The unique identifier field shall contain a 10-byte value uniquely identifying the subject as depicted in Table F.2. Its first byte shall be reserved for future use and shall be set to 01$_h$. The remaining 9 bytes shall be set to SubjectUID.

Table F.2 — Subject Unique Identifier

	RFU	**SubjectUID**
# of bits	8	72
description	Reserved for Future Use (01$_h$)	Subject Unique Identifier

When the options field in the compressed certificate header is set to 01$_b$ the value of SubjectUID shall be taken from the compressed certificate. Otherwise SubjectUID shall be set to SubjectUIDdefault as be defined in Annex E.

Example: When the SubjectCN in the compressed certificate is set to XX XX XX XX XX XX XX XX$_h$ and the SubjectUID, coming either from the compressed certificate or the default value, is set to YY YY YY YY YY YY YY YY YY$_h$ the DER encoding of the Issuer field is:

```
30h 29h
        31h 11h
                30h 0Fh
                        06h 03h
                                55h 04h 03h
                        0Ch 08h
                                XXh XXh XXh XXh XXh XXh XXh XXh
        31h 14h
                30h 12h
                        06h 03h
                                55h 04h 2Dh
                        0Ch 0Bh
                                00h 01h YYh YYh YYh YYh YYh YYh YYh YYh YYhh
```

F.3.1.8 Subject public key info

F.3.1.8.1 Algorithm identifier and public key

The subject public key Info consists of the Algorithm Identifier (used for the public key) which is fixed and the actual value of the public key which is variable for every Tag.

The field is defined to be:

```
SubjectPublicKeyInfo       ::=  SEQUENCE {
           algorithm            AlgorithmIdentifier,
           subjectPublicKey     BIT STRING }
```

F.3.1.8.2 Algorithm identifier (FIXED)

This field contains the algorithm identifier for the algorithm used by the authentication protocol.

The field is defined to be of type AlgorithmIdentifier.

```
AlgorithmIdentifier  :: = SEQUENCE {
           algorithm               OBJECT IDENTIFIER,
           parameters              ANY DEFINED BY algorithm OPTIONAL}
```

The Public Key algorithm must be set to `id-ecPublicKey` (OID 1.2.840.10045.2.1) and uses the Named Curve Parameter field `sect163r2` (OID 1.3.132.0.15).

The DER encoding of this field is fixed to:

```
30h 10h
      06h 07h
            2Ah 86h 48h CEh 3Dh 02h 01h
      06h 05h
            2Bh 81h 04h 00h 0Fh
```

F.3.1.8.3 Public key

The public key of the Tag is an elliptic curve point that is represented in uncompressed form:

```
ECPoint:: = OCTET STRING
```

The value shall be constructed from the PublicKeyX and PublicKeyY values provided in the compressed certificate as follows: 00_h 04_h || PublicKeyX || PublicKeyY, where the first byte denotes the number of unused bits in the octet string and 04_h denotes the key is provided in uncompressed format.

Example: When PublicKeyX in the compressed certificate is set to XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX_h and PublicKeyY to YY YY YY YY YY YY YY YY YY YY YY YY YY YY YY YY YY YY YY YY YY YY YY YY YY YY YY YY_h the DER encoding of the Issuer field is:

```
03h 2Ch
     00h 04h XXh XXh XXh XXh XXh XXh XXh XXh XXh XXh XXh XXh XXh XXh
     XXh XXh XXh XXh XXh XXh XXh YYh YYh YYh YYh YYh YYh YYh YYh YYh
     YYh YYh YYh YYh YYh YYh YYh YYh YYh YYh YYh YYh
```

F.3.1.9 X.509 v3 extensions

F.3.1.9.1 DER encoding for fixed Key Usage field

The extension is a context-specific sequence starting with Tag A3 followed by the length. If only the fixed Key Usage field is available it has the following DER encoding:

```
A3h 12h
     30h 10h
           30h 0Eh
                 06h 03h
                       55h 1Dh 0Fh
                 01h 01h
                       FFh
                 04h 04h
                       03h 02h 07h 80h
```

F.3.1.9.2 Key usage (FIXED)

The key usage extension shall be included in the certificate but it is fixed to the value digitalSignature(0) meaning that the public key shall only be used for verification during the authentication procedure. The Key Usage extension has (OID 2.5.29.15). The DER encoding of this extension field is:

```
30h 0Eh
      06h 03h
            55h 1Dh 0Fh
      01h 01h
            FFh
      04h 04h
            03h 02h 07h 80h
```

F.3.1.9.3 Authority key identifier

The Authority key identifier is used to identify the issuing key of the CA (e.g. if the CA has multiple keys). This field shall not be used in the X.509 certificate corresponding to a compressed certificate.

F.3.2 signatureAlgorithm (FIXED)

The signatureAlgorithm field contains the identifier for the cryptographic algorithm used by the CA to sign this certificate. It is defined to be of type AlgorithmIdentifier.

```
AlgorithmIdentifier:: = SEQUENCE {
           algorithm OBJECT IDENTIFIER,
           parameters ANY DEFINED BY algorithm OPTIONAL }
```

The value of this field corresponds to the value of the Signature field (refer to F.3.1.4) in the sequence tbsCertificate and shall be set to `ecdsa-with-SHA224` (OID 1.2.840.10045.4.3.1) and the parameter field shall be absent.

The DER encoding of this field is

```
30h 0Ah
      06h 08h
            2Ah 86h 48h CEh 3Dh 04h 03h 01h
```

F.3.3 signatureValue

The signatureValue field contains a digital signature computed over the ASN.1 DER encoded tbsCertificate. By generating this signature, a CA certifies the validity of the information in the tbsCertificate field. In particular, the CA certifies the binding between the device's public key and the subject of the certificate, i.e. the transponder IC unique identifier.

The signature value is defined by the following structure:

```
signatureValue  :: = BIT STRING
```

The value shall be constructed from the SignatureR and SignatureS values provided in the compressed certificate according to the following ASN.1 structure:[11]

```
ECDSA-Signature:: = CHOICE {
     two-ints-plus ECDSA-Sig-Value,
     point-int [0] ECDSA-Full-R      }

ECDSA-Sig-Value:: = SEQUENCE {
     r INTEGER,
     s INTEGER,
     a INTEGER OPTIONAL,
     y CHOICE { b BOOLEAN, f FieldElement } OPTIONAL
```

In the previous structure the option `ECDSA-Sig-Value` is chosen and the optional fields within it are left out.

Example: When SignatureR in the compressed certificate is set to XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX$_h$ and SignatureS to YY YY YY YY YY YY YY YY YY YY YY YY YY YY YY YY YY YY YY YY YY YY YY YY YY YY YY YY YY YY$_h$ the DER encoding of the signatureValue field is (assuming the ASN.1 padding scheme does not change their lengths):

```
03h 42h
      00h
      30h 40h
            02h 1Eh
                 XXh XXh XXh XXh XXh XXh XXh XXh XXh XXh XXh XXh XXh XXh XXh XXh
                 XXh XXh XXh XXh XXh XXh XXh XXh XXh XXh XXh XXh XXh XXh
            02h 1Eh
                 YYh YYh YYh YYh YYh YYh YYh YYh YYh YYh YYh YYh YYh YYh YYh YYh
                 YYh YYh YYh YYh YYh YYh YYh YYh YYh YYh YYh YYh YYh YYh
```

Bibliography

[1] Certicom Research. *Certicom ECC Challenge November 10,* 2009 http://www.certicom.com/images/pdfs/challenge-2009.pdf

[2] Diffie W., & Hellman M. New Directions in Cryptography. IEEE Trans. Inf. Theory. 1976 Nov., **22** (6) pp. 644–654

[3] EPC™ Radio-Frequency Identity Protocols, UHF RFID Generation-2 Version 2.0, *Specification for RFID Air Interface Protocol for Communications at 860 MHz – 960 MHz; GS1 EPCglobal™ Inc*

[4] GS1 EPCglobal™ *Tag Data Standards version 1.5 and above;* GS1 EPCglobal™ Inc

[5] Miller V. *Use of Elliptic Curves in Cryptography, Crypto '85*

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

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

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

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

[10] ITU-TRecommendationX.509,*Informationtechnology–Opensystemsinterconnection–TheDirectory: Public-key and attribute certificate frameworks* http://www.itu.int/itu-t/recommendations/rec.aspx?rec=X.509

[11] Standards for Efficient Cryptography 1 (SEC 1): *Elliptic Curve Cryptography,* Certicom Research, May 21, 2009, Version 2.0

[12] RFC 5280, *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile,* The Internet Engineering Task Force (IETF), May 2008

[13] ISO/IEC 8824, *Information technology — Open Systems Interconnection — Specification of Abstract Syntax Notation One (ASN.1)*

ICS 35.040

Price based on 39 pages