INTERNATIONAL STANDARD

ISO/IEC 15946-4

First edition
2004-10-01

Information technology — Security techniques — Cryptographic techniques based on elliptic curves —

Part 4: **Digital signatures giving message recovery**

Technologies de l'information — Techniques de sécurité — Techniques cryptographiques basées sur les courbes elliptiques —

Partie 4: Signatures digitales offrant un message de recouvrement

Reference number
ISO/IEC 15946-4:2004(E)

Contents

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.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

ISO/IEC 15946-4 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 27, *IT Security techniques*.

ISO/IEC 15946 consists of the following parts, under the general title *Information technology — Security techniques — Cryptographic techniques based on elliptic curves*:

— *Part 1: General*

— *Part 2: Digital signatures*

— *Part 3: Key establishment*

— *Part 4: Digital signatures giving message recovery*

Introduction

A potentially useful class of public-key cryptosystems consists of those schemes based on elliptic curves defined over finite fields. Elliptic curve based public-key cryptosystems make use of the following two observations:

— Every elliptic curve is endowed with a binary operation "+" under which it forms a finite abelian group.

— The group law on elliptic curves extends in a natural way to a "discrete exponentiation" on the point group of the elliptic curve.

Based on the discrete exponentiation on an elliptic curve one can easily derive elliptic curve analogues of the well-known public-key schemes of Diffie-Hellman and ElGamal type.

The security of such a public-key system depends on the difficulty of determining discrete logarithms in the group of points of an elliptic curve. For similar parameter sizes, this problem is - with current knowledge - much harder than the factorization of integers or the computation of discrete logarithms in a finite field. Indeed, since V. Miller and N. Koblitz in 1985 independently suggested the use of elliptic curves for public-key cryptographic systems, no substantial progress in tackling the elliptic curve discrete logarithm problem has been reported. In general, only algorithms that take exponential time are known to determine elliptic curve discrete logarithms. Thus, it is possible for elliptic curve based public-key systems to use much shorter parameters than the RSA system or the classical discrete logarithm based systems that make use of the multiplicative group of some finite field. This yields significantly shorter digital signatures and system parameters and allows for computations using smaller integers.

In order to meet the increasing interest in elliptic curve based public key technology, this part of ISO/IEC 15946 defines methods for implementing elliptic curve digital signature techniques that give message recovery.

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

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

The holder of this patent right has assured the ISO and IEC that he is willing to negotiate licences under reasonable and non-discriminatory terms and conditions with applicants throughout the world. In this respect, the statement of the holder of this patent right is registered with the ISO and IEC. Information may be obtained from:

ISO/IEC JTC 1/SC 27 Standing Document 8 (SD8) "Patent Information"

Standing Document 8 (SD8) is publicly available at: http://www.ni.din.de/sc27

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

Information technology — Security techniques — Cryptographic techniques based on elliptic curves —

Part 4: Digital signatures giving message recovery

1 Scope

ISO/IEC 15946 specifies public-key cryptographic techniques based on elliptic curves. These techniques include methods for the establishment of keys for symmetric cryptographic techniques, and digital signature mechanisms.

The scope of this part of ISO/IEC 15946 is restricted to cryptographic techniques based on elliptic curves defined over finite fields (including the special cases of prime order and characteristic two). The representation of elements of the underlying finite field (i.e. which basis is used) is outside the scope of this part of ISO/IEC 15946.

This part of ISO/IEC 15946 specifies five different mechanisms for digital signatures giving message recovery. The mathematical background and general techniques necessary for implementing the mechanisms are described in ISO/IEC 15946-1.

Digital signature mechanisms can be divided into the following two categories.

- When the whole message has to be stored and/or transmitted with the signature, the mechanism is named a ‘signature mechanism with appendix’.
- When the whole message, or part of it, can be recovered from the signature, the mechanism is named a ‘signature mechanism giving message recovery’. The mechanisms specified in this part of ISO/IEC 15946 fall into the second category, i.e. they give either total or partial message recovery. [For elliptic curve based digital signature schemes with appendix, see ISO/IEC 15946-2.]

NOTE In applications where a combination of algorithms is used to provide security services or when an algorithm is parameterised by the choice of a combination of other algorithms such a combination may be specified as a sequence of object identifiers assigned to these algorithms or by including the object identifiers of lower layer algorithms in the parameters field of the algorithm identifier structure specifying higher layer algorithms (for example by specifying the object identifier of a hash function as a parameter in the algorithm identifier structure of a signature scheme). The algorithm identifier structure is defined in ISO/IEC 9594-8.

NOTE The encoding of object identifiers is application dependent.

2 Normative references

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

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

ISO/IEC 10118 (all parts), *Information technology — Security techniques — Hash-functions*

ISO/IEC 14888-1, *Information technology — Security techniques — Digital signatures with appendix — Part 1: General*

ISO/IEC 15946-1:2002, *Information technology — Security techniques — Cryptographic techniques based on elliptic curves — Part 1: General*

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 15946-1 and the following apply.

3.1 collision-resistant hash-function
[ISO/IEC 10118-1] A hash-function satisfying the following property:

— it is computationally infeasible to find any two distinct inputs which map to the same output.

NOTE Computational feasibility depends on the specific security requirements and environment.

3.2 data input
A data item which depends on the entire message or a portion of the message and which forms a part of the input to the signature generation process.

3.3 domain parameter
[ISO/IEC14888-1] A data item which is common to and known by or accessible to all entities within the domain.

NOTE The set of domain parameters may contain data items such as hash-function identifier, length of the hash-token, length of the recoverable part of the message, finite field parameters, elliptic curve parameters, or other parameters specifying the security policy in the domain.

3.4 hash-code
[ISO/IEC 10118-1] The string of bits which is the output of a hash-function.

3.5 hash-function
[ISO/IEC 10118-1] A function which maps strings of bits to fixed-length strings of bits, satisfying the following two properties:

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

— for a given input, it is computationally infeasible to find a second input which maps to the same output.

3.6 hash-token
[ISO/IEC 14888-1] A concatenation of a hash-code and an optional control field which can be used to identify the hash-function and the padding method.

NOTE The control field with the hash-function identifier is mandatory unless the hash-function is uniquely determined by the signature mechanism or by the domain parameters.

3.7 message
A string of bits of any length.

3.8 pre-signature
[ISO/IEC 14888-1] A value computed in the signature generation process which is a function of the randomizer but which is independent of the message.

3.9 randomized

[ISO/IEC 14888-1] Dependent on a randomizer.

3.10 randomizer

[ISO/IEC 14888-1] A secret data item produced by the signing entity in the pre-signature production process, and not predictable by other entities.

3.11 signature

[ISO/IEC14888-1] The string of bits resulting from the signature generation process.

3.12 private signature key

[ISO/IEC 14888-1] A secret data item specific to an entity and usable only by this entity in the signature generation process.

3.13 signature generation process

[ISO/IEC 14888-1] A process which takes as inputs the message, the signature key and the domain parameters, and which gives as output the signature.

3.14 signed message

[ISO/IEC 14888-1] A set of data items consisting of the signature, the part of the message which cannot be recovered from the signature, and an optional text field.

3.15 public verification key

[ISO/IEC 14888-1] A data item which is mathematically related to an entity's signature key and which is used by the verifier in the signature verification process.

3.16 signature verification process

[ISO/IEC 14888-1] A process, which takes as its input the signed message, the verification key and the domain parameters, and which gives as its output the recovered message if valid.

4 Symbols and abbreviated terms

4.1 Symbols and notation

For the purposes of this document, the symbols and notation defined in ISO/IEC 15946-1 and the following apply.

d, d'	data input, recovered data input, respectively
h, h', h''	hash-token, recovered (truncated) hash-token, recomputed (truncated) hash-token, respectively
k	randomizer
len_h	length in bits of (truncated) hash-token
L_F	size in octets of the field F
len_rec, len_clr, len_n	length in bits of M_{rec}, M_{clr} and n, respectively
L_p	length in octets of p
len_1, len_2	length in bits of short and long redundancy, respectively
M, M_{clr}, M_{rec}	message, non-recoverable part of M, and recoverable part of M, respectively

M', M'_{rec}	recovered message, recovered part of message, respectively
m	positive integer
n, p	prime numbers
$\boldsymbol{F}(q)$	finite field with q elements, where q is a prime power.
$\boldsymbol{E}(\boldsymbol{F}(q))$	group with $\boldsymbol{F}(q)$-rational points on an elliptic curve
Π	pre-signature
r, r'	first part of the signature, first part of the received signature, respectively
s, s'	second part of the signature, second part of the received signature, respectively
x_A	the private signature key of entity A
Y_A	the public verification key of entity A
P, Q	points dependent on the chosen key generation scheme, that is $P=G$ and $Q=Y_A$ for Key Generation Scheme I resp. $P=Y_A$ and $Q=G$ for Key Generation Scheme II (See Clause 6.2)
π	conversion of a point on an elliptic curve to an integer
$\times$	Cartesian product
XOR	bit-wise exclusive or operation
SYM	the symmetric cipher to be used, whose key is generated by key derivation function KDF.
KDF	the symmetric key derivation function, whose input is an elliptic curve point and output is a value suitable for use as a symmetric key in symmetric cipher SYM.

4.2 Coding convention, length and field size

All integers are written with the most significant digit (or bit, or octet) in the leftmost position.

If an integer U is in the range $2^{i-1} \leq U < 2^i$, it is said that the length of U in bits is equal to i, and the notation $i = len\_U$ is used.

If an integer U is in the range $256^{m-1} \leq U < 256^m$, it is said that the length of U in octets equals m, and the notation $m = L_U$ is used. Hence, L_U is the least integer with the property $8 \cdot L_U \geq len\_U$.

If F is a finite prime field $\boldsymbol{F}(p)$, we set $L_F = L_p$. If F is an extension field $\boldsymbol{F}(2^m)$, we set L_F to be the least integer with the property $8 \cdot L_F \geq m$. If F is an extension field $\boldsymbol{F}(p^m)$, we set L_F to be the least integer with the property $L_F \geq log_{256}\, p^m$.

4.3 Legend for figures

The following legend is used for the figures in clause 6 depicting the signature generation and verification process for digital signatures giving message recovery.

$\boxed{\text{STEP OF THE PROCESS}}$	step of the process
$\longrightarrow$	mandatory data flow
$\dashrightarrow$	optional data flow

5 Processes

This part of ISO/IEC 15946 describes signature schemes based on the one way property of the scalar multiplication on elliptic curves defined over some finite prime field $\boldsymbol{F}(p)$, some finite field $\boldsymbol{F}(2^m)$ or some finite extension field of $\boldsymbol{F}(p)$.

A digital signature scheme is defined by the specification of the following processes:

— *Parameter generation process*;

— *Signature generation process; and*

— *Signature verification process.*

5.1 Parameter Generation Process

The parameters can be divided into domain parameters and user parameters.

5.1.1 Domain Parameters

The domain parameters consist of parameters to define a finite field, parameters to define an elliptic curve over the finite field, and other public information which is common to and known by or accessible to all entities within the domain. As well as the domain parameters for a general cryptographic scheme based on elliptic curves which are specified in Part 1 of this standard, the following parameters must be specified:

— *An identifier for the digital signature scheme used;*

— *The hash function* Hash*; and*

— *The user parameter generation procedures.*

5.1.2 User Parameters

Each entity has its own public and private parameters. The user parameters of entity *A* consist of the following:

— *The private signature key* x_A*;*

— *The public verification key* Y_A*; and*

— *(Optional) Other information, which is specific to the entity A, for the use in the signature generation and/or verification process.*

NOTE User parameters are valid only within the context of a specified set of domain parameters.

5.1.3 Validity of Parameters

The signature verifier may require assurance that the domain parameters and public verification key are valid, otherwise there is no assurance of meeting the intended security even if the signature verifies. The signer may also require assurance that the domain parameters and public verification key are valid, otherwise an adversary may be able to generate signatures that verify.

Assurance of validity of domain parameters can be provided by one of the following:

— *selection of valid domain parameters from a trusted published source, such as a standard;*

— *generation of valid domain parameters by a trusted third party, such as a CA;*

— *validation of candidate domain parameters by a trusted third party, such as a CA;*

— *for the signer, generation of valid domain parameters by the signer using a trusted system; and*

— *validation of candidate domain parameters by the user (i.e., the signer or verifier).*

Assurance of validity of a public verification key can be provided by one of the following:

— *for the signer, generation of the public verification/private signature key pair using a trusted system;*

— *for the signer or verifier, validation of the public verification key by a trusted third party, such as a CA; and*

— *validation of the public verification key by the user (i.e., the signer or verifier).*

5.2 Signature Generation Process

The following data items are required for the signature generation process:

— *the domain parameters;*

— *the signer A's user parameters including the private signature key* x_A*; and*

— *the message* M.

For all the schemes the signature generation process consists of the following procedures:

— *splitting message;*

— *computation of redundancy, or computation of the message digest (option);*

— *elliptic curve computations;*

— *computations modulo the group order of the base point* G*; and*

— *formatting the signed message.*

The output of the signature generation process is a pair of integers (r, s) that constitutes A's digital signature of the message M.

5.3 Signature Verification Process

The following data items are required for the signature verification process:

— *the domain parameters;*

— *elements of the signer A's user parameters including the public verification key* Y_A *(but not the private signature key* x_A*);*

— *the non-recoverable message* M'_{clr} *(if any); and*

— *the received signature for* M*, represented as the two integers,* r' *and* s'*.*

For all the schemes the signature verification process consists of some or all of the following procedures:

— *signature size verification;*

— *recovering the pre-signature and the data input;*

— *recovering the message;*

— *verification of redundancy, or computation of the message digest (optional);*

— *computations modulo the group order of the base point* G*;*

— *elliptic curve computations; and*

— *signature checking.*

If all procedures are passed successfully, the signature is accepted by the verifier, otherwise it is rejected.

6 General Model for Digital Signatures giving message recovery

This clause contains a general model for the five signature schemes specified in this part of ISO/IEC 15946.

6.1 Requirements

6.1.1 Domain parameters

Users who wish to employ one of the digital signature mechanisms specified in this part of ISO/IEC 15946 shall select the domain parameters of the digital signature scheme:

— *: a finite field* **F**(q)*;*

— *an elliptic curve* **E** *over* **F**(q) *which has a unique cyclic subgroup of prime order* n*; and*

— *a point* G *on* **E** *of prime order* n*.*

Agreement on these choices amongst the users is essential for the purpose of the operation of the digital signature mechanism giving message recovery.

NOTE The sizes of both q and n correspond to the security parameters, and shall be chosen to meet the defined security objectives.

6.1.2 Type of redundancy

Users shall select the type of redundancy.

— *natural redundancy;*

— *added redundancy; or*

— *both.*

NOTE A message with natural redundancy means that the message includes redundancy naturally or that redundancy of the message is verifiable implicitly in some applications. A message with added redundancy may be constructed by the hash token of the message or the recoverable message. The natural or added redundancy may be anything agreed upon and able to be checked by the communicating parties. Total redundancy, which consists of natural redundancy and added redundancy, shall be greater than some minimum value specified by the application.

If users use added redundancy, the types of the redundancy shall be fixed to be either:

— *short redundancy; or*

— *long redundancy.*

Short redundancy shall be used if the entire message is recoverable from the signature.

Long redundancy shall be used if only part of the message, not the entire message, is recoverable from the signature.

The length of the short redundancy and the long redundancy, *len*_1 and *len*_2 shall be fixed, respectively:

— *If the bit-length of the message is at most* len_n – len_*1 –1, then the entire message is recoverable from the signature and short redundancy is used; and*

— *If the bit-length of the message is greater than* len_n – len_*1 –1, then the recoverable part of the message is at most* len_n – len_*2 –1 bits and long redundancy is used.*

Agreement on these choices amongst the users is essential for the purpose of the operation of the digital signature mechanism giving message recovery.

NOTE The values of the parameters *n*, *len*_1 and *len*_2 also affect the security level of the signatures giving message recovery. Typical values of *len*_1 are 64 or 80. Typical values of *len*_2 vary from 136 to 168. It is also possible to set *len*_1 = *len*_2.

6.2 Summary of Functions and Procedures

The signature schemes specified in this part of ISO/IEC 15946 give message recovery. More precisely, some of the data which is input to the signature generation function is recovered from the signature as part of the signature verification procedure.

The signature scheme consists of the following functions and procedures:

— *producing domain parameters;*

— *producing signature and verification key;*

— *producing randomizer and pre-signature;*

— *computing the first part of the signature;*

— *computing the second part of the signature;*

— *recovering the pre-signature; and*

— *recovering the data input.*

6.2.1 Signature and verification key

One of the following two methods shall be used to compute the key pair consisting of the public and the private signature key. The signing entity shall keep the private signature key secret.

(1) Key generation I

Given a valid set of elliptic curve domain parameters a private signature key and corresponding public verification key may be generated as follows.

1 Select a random or pseudorandom integer x_A in the set [2, n-2]. The integer x_A must be protected from unauthorised disclosure and be unpredictable.

2 Compute the point $Y_A = x_A G$.

3 The key pair is (Y_A, x_A), where Y_A will be used as public verification key, and x_A is the private signature key.

To allow an unified representation of the algorithms, put $P{:=}G$ and $Q{:=}Y_A$.

(2) Key generation II

Given a valid set of elliptic curve domain parameters a private signature key and corresponding public verification key may be generated as follows.

1. Select a random or pseudorandom integer e in the set [2, n-2] and compute an integer x_A in the interval [2, n-2] with the property $x_A e$=1 *mod n.* The integers x_A and e must be protected from unauthorised disclosure and be unpredictable.

2. Compute the point $Y_A = eG$.

3. The key pair is (Y_A, x_A), where Y_A will be used as public verification key, and x_A is the private signature key.

To allow an unified representation of the algorithms, put $P{:=}Y_A$ and $Q{:=}G$.

Prior to use of the public verification key the verifier shall have assurance about its validity and ownership. This validation may be obtained by various means, see Clause 5.1.3.

6.2.2 Randomizer and pre-signature

Prior to each signature computation the signing entity must have a fresh, secret randomizer value available. The randomizer is an integer k such that $1 < k < n-1$. The implementation of the signature scheme must ensure that the following two requirements are satisfied:

- *Used randomizers shall never be disclosed. Once used, they shall be destroyed.*
- *Randomizers shall be generated in such a way tat the probability that the same randomizer is used to produce signatures for two different messages shall be negligible.*

The pre-signature is computed as a function of the randomizer.

NOTE Disclosure of a randomizer after use may jeopardise the secrecy of the private signature key. Used randomizers are never required again by the signer or verifier, and can thus be erased immediately after signature computation. If the same value of the randomizer is used to produce signatures for two different messages, then it might be possible to construct a valid signature for a message that the signer does not wish to be signed from the pair of signatures, or even to deduce the private signature key.

6.2.3 Computing the first part of the signature

The first part of the signature is computed as a function of the pre-signature Π and the data input *d,* which is an integer with $0 \leq d < n$ that depends upon the message. The first part of the signature is an integer *r* such that $0 < r < n$.

6.2.4 Computing the second part of the signature

This computation is determined by the private signature key x_A. Given the first part of the signature *r* and the randomizer *k* the second part of the signature is an integer *s* such that $0 \leq s < \text{n}$.

6.2.5 Recovering the pre-signature

This computation is determined by the public verification key Y_A. Given the signature (*r, s*), the pre-signature Π is recovered.

6.2.6 Recovering the data input

Given the first part *r* of the signature and the recovered pre-signature Π' the data input *d* is recovered.

6.3 Signature generation process

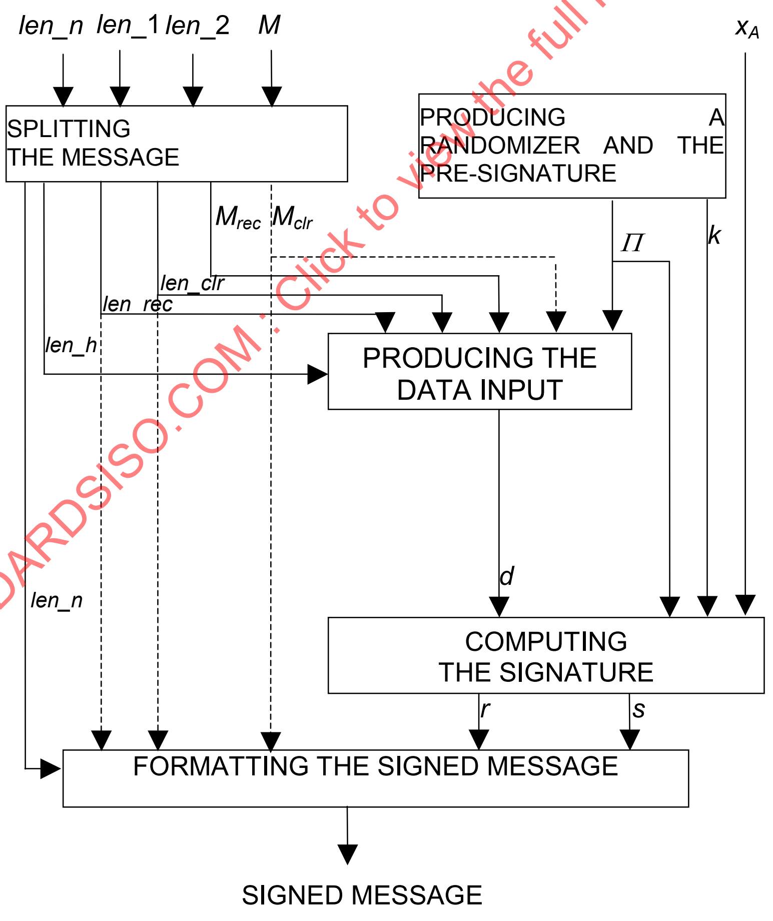

Figure 1 — The signature generation process

Figure 1 shows the the signature generation process, which consists of the following steps:

— *producing a randomizer and the pre-signature;*

— *splitting the message;*

— *producing the data input;*

— *computing the signature; and*

— *formatting the signed message.*

6.3.1 Producing a randomizer and the pre-signature

The pre-signature is an intermediate data item that is produced during the signature generation process in any randomized signature mechanism. First a randomizer k, which is an integer, is produced. The pre-signature is a public data item, while the value of the randomizer shall be available only to the signature generation process.

NOTE Randomizers can be produced and corresponding pre-signatures are computed off-line and stored securely for future use by the signature generation process.

6.3.2 Splitting the message

The message M is split into the recoverable part M_{rec} and the non-recoverable part M_{clr} of the message, and *len_rec* and *len_clr* are defined to be the length of the recoverable part M_{rec} and the non-recoverable part M_{clr} respectively. In the case of added redundancy, the message M may be split as follows:

— *If the bit length* len_M *of the message* M *satisfies* $\text{len\_M} \leq \text{len\_n} - \text{len\_}1 - 1$*, then the entire message* M *is recoverable from the signature. In this case the equations:* $\text{M}_{\text{rec}} = \text{M}$ *and* len_rec= len_M*. Further,* len_clr = *0 and* len_h = len_*1 hold;*

— *If the bit length* len_M *of the message* M *satisfies* $\text{len\_M} > \text{len\_n} - \text{len\_}1 - 1$*, then* len_rec *is defined to be an integer such that* $\text{len\_rec} \leq \text{len\_n} - \text{len\_}2 - 1$*. The len_rec leftmost bits of* M *form the recoverable part* M_{rec} *of the message. The* len_M –len_rec *rightmost bits form the non-recoverable part* M_{clr} *of the message* M*, and the equations* len_rec = len_M – len_clr *and* len_h = len_*2 hold.*

6.3.3 Producing the data input

The inputs to the data input function are *len_rec*, *len_clr*, the pre-signature Π, the non-recoverable part M_{clr} (optional) and the hash-token, the recoverable message M_{rec} with added redundancy, or the recoverable message M_{rec} with natural redundancy. The hash-token is formed by the hash-code itself, or with the hash-function identifier concatenated to the right of the hash-code, where the hash-code is computed by hashing the message. The choice of whether or not the hash-token includes the hash-function identifier shall be controlled by the domain parameters. The output of the data input function is d, which is in the range $0 \leq d < n$, after conversion to an integer.

NOTE The choice of data input may be determined by each application or signature scheme.

6.3.4 Computing the signature

The signatures produced by this scheme have two parts r and s. The first part r is computed as a function of the data input d and the pre-signature Π, see 6.2.4. The second part s is computed as a function of the first part r, the randomizer k, and the private signature key x_A, see 6.2.5.

6.3.5 Formatting the signed message

Knowledge of the lengths of the recoverable and non-recoverable parts of the message is necessary for the successful opening and verification of the signed message. Unless given by the domain parameters, this information must be included in the signed message.

The signed message consists of the following data items:

— *the non-recoverable part of the message and the message length;*

— *the first part* r *of the signature; and*

— *the second part* s *of the signature.*

6.4 Signature verification process

Figure 2 shows the signature verification process, which consists of the following steps:

— *opening the signed message;*

— *signature size verification;*

— *recovering the pre-signature or the data input;*

— *recovering the data input or the message;*

— *recomputing the hash-token(optional); and*

— *checking the signature.*

Checking the signature consists of :

- comparing the recovered and recomputed (truncated) hash-tokens; or
- verifying the redundancy.

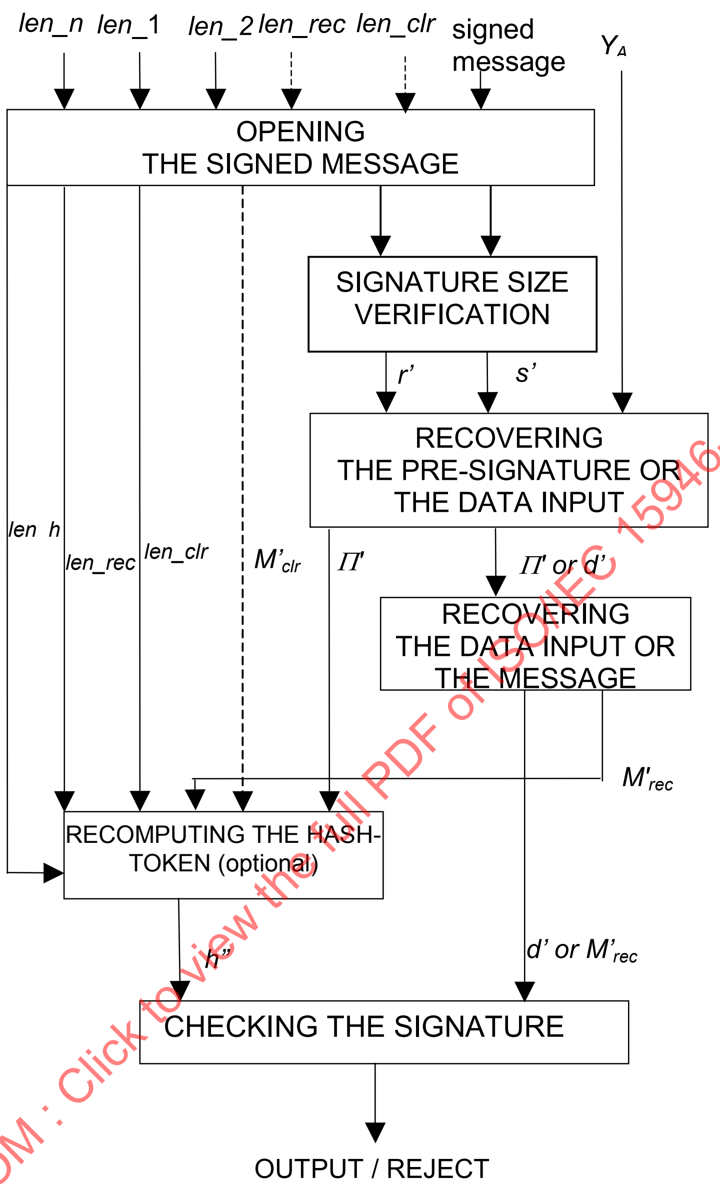
len_n
len_1
len_2
len_rec
len_clr
signed message
Y_A
OPENING THE SIGNED MESSAGE
SIGNATURE SIZE VERIFICATION
r'
s'
RECOVERING THE PRE-SIGNATURE OR THE DATA INPUT
len h
len_rec
len_clr
M'_{clr}
Π'
Π' or d'
RECOVERING THE DATA INPUT OR THE MESSAGE
M'_{rec}
RECOMPUTING THE HASH-TOKEN (optional)
h"
d' or M'_{rec}
CHECKING THE SIGNATURE
OUTPUT / REJECT

Figure 2 — Signature verification process

6.4.1 Opening the signed message

When starting this step, the verifier must have the following information available:

— *the lengths of the different message parts included in the signed message; and*

— *the values of the parameters* len_n, len_*1 and* len_*2.*

The verifier extracts the different parts of the signed message in the following order:

— *the non-recoverable part of the message;*

— *the first part* r' *of the signature; and*

— *the second part* s' *of the signature.*

6.4.2 Signature size verification

The verifier shall verify the size of the parts of a signature.

6.4.3 Recovering the pre-signature and the data input

At the beginning of this step the verifier must have the following information available:

— *the public parameters which specify the signature scheme in use; and specifically,*

— *the public verification key* Y_A *of the signing entity.*

The computations at this step are specific to the signature scheme in use. The pre-signature and the data input are recovered from the signature as specified in 6.2.6 and 6.2.7, respectively. The recovered pre-signature is Π' and the recovered data input is *d'*.

6.4.4 Recovering the data input or the message

The recovered data input *d'* is converted to a string of bits. The data input *d' is*

— *the hash token;*

— *the recoverable message with added redundancy; or*

— *the recoverable message with natural redundancy.*

6.4.5 Recomputing the hash-token

First, the hash-function used by the signing entity in 6.3 is identified, possibly by retrieving the hash-function identifier from the recovered hash-token. Then the hash-code is recomputed by hashing the message.

The recomputed hash-code is used to obtain the recomputed hash-token by optionally concatenating the hash-function identifier.

6.4.6 Checking the signature

Checking the signature consists of:

— *Comparing the recovered and the recomputed (truncated) hash-tokens; or*

It is to compare the recomputed (truncated) hash-token h" with the recovered (truncated) hash-token h'. The signature shall be rejected if these two values are not equal.

— *Verifying the redundancy.*

It is to check added, and/or natural redundancy of the recovered message. The signature shall be rejected if the redundancy is not confirmed.

7 ECNR (Elliptic Curve Nyberg-Rueppel message recovery signature)

ECNR is based on a signature scheme defined in [4].

7.1 Domain and User Parameters

The keys of ECNR are produced as follows:

— A*'s private signature key* x_A*:* x_A *is a random integer in the interval [2,* n-2*]; and*

— A*'s public verification key* Y_A *computed as Clause 6.2.2.*

7.2 Signature Generation Process

The input to the signature generation process consists of:

— *the domain parameters;*

— *the private signature key* x_A*; and*

— *the data* d*, an element of* **F**(n)*.*

The data d is produced from a message, see Clause 6.3.3. The output of the signature generation process is a pair $(r,s) \in \boldsymbol{F}(n) \times \boldsymbol{F}(n)$ that constitutes A 's digital signature to the data.

To sign a data d, A executes the following steps:

7.2.1 Producing a randomizer and the pre-signature(Elliptic curve computations)

1. Select a random integer k in the interval $[2, n\text{-}2]$.

2. Compute the elliptic curve point $(x_1, y_1)=kP$.

7.2.2 Computations modulo the group order of *G*(Arithmetic operations in *F*(*n*))

1. Set $\Pi \equiv \pi(kP) \pmod n$.

2. Compute $r \equiv d + \Pi \pmod n$.

3. Compute $s \equiv k - x_A r \pmod n$.

4. Erase k.

If the signature generation process yields either s=0 or r=0, then the process of signature generation must be repeated with a new random value k.

7.2.3 Formatting the signed message

The pair $(r, s) \in \boldsymbol{F}(n) \times \boldsymbol{F}(n)$ constitutes A's signature to the data d.

7.3 Signature Verification Process

The signature verification process consists of three steps: calculation of the message digest, elliptic curve computations, and signature checking.

The input to the signature verification process consists of:

— *the domain parameters;*

— A*'s public verification key* Y_A*;*

— *the received signature for* d*, represented as the two integers,* r' *and* s'*; and*

— *the non-recoverable message* M'_{clr} *(if any).*

To verify A's signature for d, B executes the following steps:

7.3.1 Signature size verification

Verify that $0 < r' < n$ and $0 < s' < n$; if not, then reject the signature.

7.3.2 Recovering the pre-signature and the data input (Elliptic curve computations)

1. Compute $R_1' = s' P + r' Q$

2. Compute $\Pi' \equiv \pi(R_1')$ (mod n).

7.3.3 Recovering the data input or the message

Compute $d' \equiv r' - \Pi'$ (mod n).

7.3.4 Checking the signature

Check the redundancy. If it is correct, output d', otherwise reject.

8 ECMR (Elliptic Curve Miyaji message Recovery signature)

ECMR is based on a signature scheme defined in [5].

8.1 Domain and User Parameters

The keys for ECMR are produced as follows:

— A*'s private signature key* x_A*:* x_A *is a random integer in the interval [2,* n-2*]; and*

— A*'s public verification key* Y_A *computed as Clause 6.2.2.*

A also selects a hash function

Hash: $\boldsymbol{E}(\boldsymbol{F}(q)) \to \{0,1\}^{len\_n}$

8.2 Signature Generation Process

The input to the signature generation process consists of:

— *the domain parameters;*

— *the private signature key* x_A*; and*

— *the data d with added or natural redundancy is in* $\{0,1\}^{len\_n}$.

The data d is produced from the message, see Clause 6.3.3. The output of the signature generation process is a pair $(r,s) \in \{0,1\}^{len\_n} \times \boldsymbol{F}(n)$ that constitutes A's digital signature to the data.

To sign a data d, A executes the following steps:

8.2.1 Producing a randomizer and the pre-signature(Elliptic curve computations)

1. Select a random integer k in the interval [2, n-2].

2. Compute the elliptic curve point $(x_1, y_1)=kP$.

8.2.2 Computing the signature(Computations modulo the group order of G)

1. Set Π = $Hash(kP)$.

2. Compute $r = d$ XOR Π

3. Compute $s \equiv (rk\text{-}r\text{-}1)/(x_A+1)$ (mod n).

4. Erase k.

If the signature generation process yields either s=0 or $r = 0$ mod n, then the process of signature generation must be repeated with a new random value k.

8.2.3 Formatting the signed message

The pair $(r, s) \in \{0,1\}^{len\_n} \times \boldsymbol{F}(n)$ constitutes A's signature to the data d.

8.3 Signature Verification Process

The signature verification process consists of three steps: calculation of the message digest, elliptic curve computations, and signature checking.

The input to the signature verification process consists of:

— *the domain parameters;*

— A*'s public verification key* Y_A*;*

— *the received signature for* d*, represented as the two integers,* r' *and* s'*; and*

— *the non-recoverable message* M'_{clr} *(if any).*

To verify A's signature for d, B executes the following steps:

8.3.1 Signature size verification

Verify that $r' \neq 0$ mod n and $r' \in \{0,1\}^{len\_n}$ and $0 < s' < n$; if not, then reject the signature.

8.3.2 Recovering the pre-signature and the data input (Elliptic curve computations)

1. Compute $R_1' = ((1+r'+s')/(r'))P + (s'/r')\ Q$

2. Compute $\Pi' = Hash(R_1')$.

8.3.3 Recovering the data input or the message

Compute $d' = r'$ XOR Π'.

8.3.4 Checking the signature

Check the redundancy. If it is correct, output d', otherwise reject.

9 ECAO (Elliptic Curve Abe-Okamoto message recovery signature)

ECAO is based on a signature scheme defined in [6].

9.1 Domain and User Parameters

The keys of ECAO are produced as follows:

— A's *private signature key* x_A*:* x_A *is a random integer in the interval [2,* n-2*]; and*

— A*'s public verification key* Y_A *computed as Clause 6.2.2.*

A also selects a hash function

Hash ($\{0,1\}^* \rightarrow \{0,1\}^{len}$) where len $\geq$max($8L_2$, $8L_F - 8L_2$, number of bits of n) and construct three hash functions, where {0,1}* is an arbitrary bit data.

$Hash_1$ ($\{0,1\}^* \rightarrow \{0,1\}^{8L_2}$), $Hash_2$($\{0,1\}^* \rightarrow \{0,1\}^{8L_F - 8L_2}$), and $Hash_3$ ($\{0,1\}^* \rightarrow [1, n-2]$)

by taking the necessary amount of bits from the output of *Hash*.

Here we use the following notations:

$[X]^a$	a bits of X from the most significant octet.
$[X]_b$	b bits of X from the least significant octet.

9.1.1 Producing the Data Input

The input to the data input producing process consists of

— *the domain parameters; and*

— *message* M_{rec}.

The output of this process is data *d* computed as

$$d = Hash_1(M_{rec}) \,||\, (Hash_2(Hash_1(M_{rec})) \text{ XOR } M_{rec}).$$

The resulting *d* is in the range $0 \leq d < 2^{8L_F}$

9.2 Signature Generation Process

The input to the signature generation process consists of:

— *the domain parameters;*

— *the private signature key* x_A*;*

— *data* d*; and*

— *message* M_{clr} *(if any).*

The output of the signature generation process is a pair $(r,s) \in \{0,1\}^{8L_F} \times \mathbf{F}(n)$ that constitutes *A*'s digital signature to the data.

To sign data *d*, *A* executes the following steps:

9.2.1 Producing a randomizer and the pre-signature(Elliptic curve computations)

1. Select a random integer *k* in the interval [2, *n*-2].

2. Compute the elliptic curve point *kP* , and convert it into integer $x_1 = \pi(kP)$.

9.2.2 Computing the signature(Computations modulo the group order of G)

1. Compute $r = d$ XOR $[x_1]_{8L_F}$

2. Compute $\Pi = Hash_3\ (r \,||\, M_{\mathrm{clr}})$.

3. Compute $s \equiv k - x_A \Pi \pmod{n}$

4. Erase k.

If the signature generation process yields r=0 or s=0, then the process of signature generation must be repeated with a new random value k.

9.2.3 Formatting the signed message

The pair $(r, s) \in \{0,1\}^{8L_F} \times \boldsymbol{F}(n)$ constitutes A's signature to the data d.

9.3 Signature Verification Process

The signature verification process consists of three steps: calculation of the message digest, elliptic curve computations, and signature checking.

The input to the signature verification process consists of:

— *the domain parameters;*

— A*'s public verification key* Y_A*;*

— *the received signature for* d*, represented as two integers,* r' *and* s'*;*

— *hash functions,* $Hash_1$, $Hash_2$ *and* $Hash_3$*; and*

— *the non-recoverable message* M'_{clr} *(if any).*

To verify A's signature for d, B executes the following steps:

9.3.1 Signature size verification

Verify that $r' \neq 0$, $r' \in \{0,1\}^{8L_F}$ and $0 < s' < n$; if not, then reject the signature.

9.3.2 Recovering the pre-signature and the data input (Elliptic curve computations)

1. Compute $\Pi' = Hash_3\ (r' \,||\, M'_{clr})$.

2. Compute the elliptic curve point $x'_1 = \pi\,(s'P + \Pi' Q)$

3. Compute $d' = r'$ XOR $[x'_1]_{8L_F}$

9.3.3 Recovering the data input or the message

Compute $M'_{\mathrm{rec}} = [d']_{8L_F - 8L_2}$ XOR $Hash_2\ ([d']^{8L_2})$.

9.3.4 Checking the signature

Check whether the following equation holds or not: $[d']^{8L_2} = Hash_1\ (M'_{\mathrm{rec}})$. If it holds, output $M'_{\mathrm{rec}} \,||\, M'_{\mathrm{clr}}$, otherwise, reject.

10 ECPV (Elliptic Curve Pintsov-Vanstone message recovery signature)

ECPV is based on a signature scheme defined in [7].

10.1 Domain and User Parameters

The keys of ECPV are produced as follows:

— *A's private signature key* x_A*:* x_A *is a random integer in the interval [2,* n-2*]; and*

— *A's public verification key* Y_A *computed as Clause 6.2.2.*

The parties agree on the following:

Hash: $\{0,1\}^* \to [1, n-1]$;

SYM: $\{0,1\}^* \to \{0,1\}^{len\_n}$;

KDF: $\{0,1\}^* \times \{0,1\}^{len\_n} \to \{0,1\}^*$

NOTE Sym's key size corresponds to the security parameter and shall be chosen to achieve security objectives. The symmetric cipher may use XOR encryption. And, the key derivation function may use a hash-function.

10.2 Signature Generation Process

The input to the signature generation process consists of:

— *the domain parameters;*

— *the private signature key* x_A*;*

— *the message* M *to be signed;*

— M *is split to the recoverable part* M_{rec} *as being the leftmost octets of* M *as agreed upon and the remaining portion of the message* M_{clr}*; and*

— *the length of both* M_{rec} *and* M_{clr} *shall be fixed and be agreed upon by the communicating parties.*

The output of the signature generation process is a pair $(r,s) \in \{0,1\}^* \times \boldsymbol{F}(n)$ that constitutes *A*'s digital signature to the message.

To sign a message *M*, *A* executes the following steps:

10.2.1 Producing a randomizer and the pre-signature(Elliptic curve computations)

1. Select a random integer k in the interval [2, n-2].

2. Compute the elliptic curve point $R = (x_1, y_1) = kP$.

10.2.2 Producing the data input

1. Form the data input d by taking M_{rec} and the added redundancy, as agreed upon as part of the domain parameters, see clause 6.3.3.

2. Compute the symmetric key $\sigma = KDF(R)$.

3. Compute $r = SYM(d, \sigma)$.

4. Calculate $\Pi = Hash\ (r \,||\, M_{clr})$.

10.2.3 Computing the signature(Computations modulo the group order of G)

1. Compute $s \equiv k - x_A \Pi \pmod{n}$.

2. Erase k.

3. Output the signature (r, s) and the partial message part M_{clr} (which may be null).

If the signature generation process yields s=0, then the process of signature generation must be repeated with a new random value k.

10.2.4 Formatting the signed message

The pair $(r, s) \in \{0, 1\}^* \times \boldsymbol{F}(n)$ constitutes A's signature to the message M.

10.3 Signature Verification Process

The signature verification process consists of three steps: calculation of the message digest, elliptic curve computations, and signature checking.

The input to the signature verification process consists of:

— *the domain parameters;*

— A*'s public verification key* Y_A*;*

— *the received signature for* M*, represented as two integers,* r' *and* s'*; and*

— *the non-recoverable message* M'_{clr} *(if any);*

To verify A's signature for d, B executes the following steps:

10.3.1 Signature size verification

Verify that $0 < s' < n$; if not, then reject the signature.

10.3.2 Recovering the pre-signature and the data input (Elliptic curve computations)

1. Calculate $\Pi' = Hash(r' \,||\, M'_{clr})$.

2. Compute $R_1' = s'P + \Pi'Q$.

10.3.3 Recovering the data input or the message

1. If R_1' is the point at infinity, then reject the signature. Otherwise execute the following steps.

2. Compute the symmetric key $\sigma' = KDF(R_1')$.

3. Compute $d' = SYM^{-1}(r', \sigma')$.

4. Verify the added redundancy of d' and recover M'_{rec}. If the added redundancy is not correct, reject the signature.

5. Form the output message $M' = M'_{rec} \,||\, M'_{clr}$.

10.3.4 Checking the signature

Check the redundancy. If it is correct, output *d'*, otherwise reject.

11 ECKNR (Elliptic Curve KCDSA/Nyberg-Rueppel message recovery signature)

ECKNR is based on a signature scheme defined in [9].

11.1 Domain and User Parameters

The keys of ECKNR are produced as follows:

— A*'s private signature key* x_A*:* x_A *is a random integer in the interval [2,* n – 2*]; and*

— A*'s public verification key* $Y_A = (x_0, y_0)$ *computed as Clause 6.2.2.*

There is another user parameter used in ECKNR:

— *The hash-code,* z_A*, of* A*'s certification data, i.e.* $z_A = \mathrm{Hash}_2(\mathrm{Cert\_Data})$*. This certification data is considered public information and shall be available to all parties involved in the signing and verification of a message.*

Here, *Cert_Data* denotes the certification data of *A*, which contains at least *A*'s public verification key Y_A, and may contain *A*'s distinguished identifier or some of the domain parameters.

NOTE The hash function $Hash_2$ is not required to be a collision-resistant hash function. The most straightforward implementation would be for *Cert_Data* to be *A*'s public verification key itself. *Cert_Data* can be selected as *Cert_Data* = $\pi((x_0, 0)) + \pi((y_0, 0))\cdot q$, where *A*'s public verification key is $Y_A = (x_0, y_0)$.

The parties agree on the following:

$Hash_1$: $\boldsymbol{E}(\boldsymbol{F}(q)) \rightarrow [1, n-1]$.

11.2 Signature Generation Process

The input to the signature generation process consists of:

— *the domain parameters;*

— A*'s private signature key* x_A*;*

— A*'s hash-code of certification data* z_A*;*

— *the data* d *with added or natural redundancy in* $\{0,1\}^{len\_n}$*; and*

— *the non-recoverable message* M_{clr} *(if any).*

The data *d* is produced from a message, see Clause 6.3.3. The output of the signature generation process is a pair $(r,s) \in \{0,1\}^{len\_n} \times \boldsymbol{F}(n)$ that constitutes *A* 's digital signature to the data.

To sign a data *d*, *A* executes the following steps:

11.2.1 Producing a randomizer and the pre-signature(Elliptic curve computations)

1. Select a random integer *k* in the interval [2, *n* – 2].

2. Compute the elliptic curve point $(x_1, y_1) = kP$

3. Compute $\Pi = Hash_1(kP)$.

11.2.2 Computations modulo the group order of *G*(Arithmetic operations in *F*(*n*))

1. Compute $r = d$ XOR Π XOR $Hash_1(z_A \,||\, M_{clr})$.

2. Compute $s \equiv k - x_A r \pmod{n}$.

3. Erase *k*.

If the signature generation process yields either $s = 0$ or $r = 0 \bmod n$, then the process of signature generation must be repeated with a new random value *k*.

11.2.3 Formatting the signed message

The pair $(r, s) \in \{0,1\}^{len\_n} \times \boldsymbol{F}(n)$ constitutes *A's* signature to the data *d*.

11.3 Signature Verification Process

The signature verification process consists of three steps: calculation of the message digest, elliptic curve computations, and signature checking.

The input to the signature verification process consists of:

— *the domain parameters;*

— A*'s public verification key* Y_A*;*

— A*'s hash-code of certification data* z_A*;*

— *the received signature for* d*, represented as the two integers,* r' *and* s'*; and*

— *the non-recoverable message* M'_{clr} *(if any).*

To verify *A*'s signature for *d*, *B* executes the following steps:

11.3.1 Signature size verification

Verify that $r' \neq 0 \bmod n$ and $r' \in \{0,1\}^{len\_n}$ and $0 < s' < n$; if not, then reject the signature.

11.3.2 Recovering the pre-signature and the data input (Elliptic curve computations)

1. Compute $R' = s' P + r' Q$

2. Compute $\Pi' = Hash_1(R')$.

11.3.3 Recovering the data input or the message

Compute $d' = r'$ XOR Π' XOR $Hash_1(z_A \,||\, M'_{clr})$.

11.3.4 Checking the signature

Check the redundancy. If it is correct, output *d'*, otherwise reject.

Annex A
(informative)

Numerical examples

Throughout this annex we refer to ASCII coding of data strings; this is equivalent to coding using ISO 646.

A.1 Numerical examples for ECNR

NOTE Truncated hash token h is L octets of RIPEMD-160 (Dedicated Hash-Function 1 of ISO/IEC 10118-3).

A.1.1 Elliptic curve over a prime field

p	`ffd5d55f a9934410 d3eb8bc0 4648779f 13174945`
Equation of E	$y^2 \equiv x^3 + ax + b \pmod{p}$
a	`710062dc b53dc6e4 2f8227a4 fbac2240 bd3504d4`
b	`4163e75b b92147d5 4e09b0f1 3822b076 a0944359`
Number of points on E	`ffd5d55f a9934410 d3ed8b96 19eb7ba5 a5992d9e`
x-coordinate of G	`3c1e27d7 1f992260 cf3c31c9 0d80b635 e9fd0e68`
y-coordinate of G	`c436efc0 041bbf09 47a304a0 05f8d43a 36763031`
n(order of G)	`2aa3a38f f1988b58 235241ee 59a73f46 46443245`
len_n	`158 bits`
Signature key x_A	`24a3a993 ab59b12c e7379a12 3487647e 5ec9e0ce`
x-coordinate of Y_A	`e564ac ae27d227 1c4af829 cface6de cc8cdce6`
y-coordinate of Y_A	`7bd48ce1 08ffd3cf a38177f6 83b5bcf4 fd97a4a9`
Randomizer k	`8a8bea9 f2b40ce7 40067226 1d5c05e5 fd8ab326`
Π(x-coordinate of kG)	`177b7c44 ac2f7f79 96aefd27 c68d59e0 f8e01599`
y-coordinate of kG	`399ea116 298975bb 449d126f 6c97bddf c4e8782e`
Message to be signed	`plaintext`
$M(=M_{rec})$ (M_{clr} is empty)	`70 6c61696e 74657874`

Length $L(=L_1)$ of truncated hash-token	`10 octets`
Recoverable length L_{rec}	`9 octets`
Non-recoverable length L_{clr}	`0 octet`
Truncated hash-token h (=short redundancy)	`ee8c 9f8d08a9 ad4a318f`
Data input $d=h\|\|M$	`ee8c9f 8d08a9ad 4a318f70 6c61696e 74657874`
First part of signature r	`186a08e4 39382926 e0e08c98 32eec34f 6d458e0d`
Second part of signature s	`15de11da 7f10f7bf 73a46364 39e4801f 7be73687`

A.1.2 Elliptic curve over an extension field $F(2^m)$

Galois field $\mathbf{F}(2^{185})$ with the polynomial $x^{185} + x^{69} + 1$.

NOTE This is a standard polynomial basis implementation; and $\pi(kG) = 2^{184}s_{184} + 2^{183}s_{183} + \ldots + 2s_1 + s_0$, where the x-coordinate of kG is $s_{184}x^{184} + s_{183}x^{183} + \ldots + s_1x + s_0$.

Equation of E	$y^2 + xy = x^3 + ax^2 + b$
a	`0`
b	`1ee9`
Number of points on E	`1ffffff ffffffff ffffffff dbf2f889 b73e4841 75f94ebc`
x-coordinate of G	`39a936 dc2047c0 af0d2c51 dbda3b35 ec6bfcd8 79aafc4e`
y-coordinate of G	`cf2fdc 81f9cc7f 20049c7b 3a84e78b 42aae58a 845f0f3f`
n	`7fffff ffffffff ffffffff f6fcbe22 6dcf9210 5d7e53af`
len_n	`183 bits`
x_A	`425cf c2845d4b 3cd31b18 64794b36 7c7014fa dc53f4e7`
x-coordinate of Y_A	`1bca747 d3a2a53b 7f9e54ea 05298730 ca97a65a 613bf68c`
y-coordinate of Y_A	`1b83c41 c3c16c34 eef6797f c8eeb64e 53049615 f893287c`
k	`6ed447 060c5a4f ed681056 f63ab10a 7e376019 864011fa`
$\Pi(=\pi(kG))$	`01ffac83 70d6bf9b f444aa0b a43e4a05 4b9a46e1 5c60c2fe`

y-coordinate of *kG*	`16f5c4a 45843e31 85ee7483 f41ec114 a4e79cdf 97f8957a`
Message to be signed	`plaintext`
$M(=M_{rec})$ (M_{clr} is empty)	`70 6c61696e 74657874`
Length $L(=L_1)$ of truncated hash-token	`10 octets`
Recoverable length L_{rec}	`9 octets`
Non-recoverable length L_{clr}	`0 octet`
Truncated hash-token *h* (=short redundancy)	`06fd c555a5b1 8286cc37`
Data input $d=h\|\|M$	`6fdc5 55a5b182 86cc3770 6c61696e 74657874`
First part of signature *r*	`7fac83 70ddbd61 49ea5b8e 4614470e 6e8cfa1e b84b4065`
Second part of signature *s*	`13eed1 725ee5b5 f16193a0 248e3085 43c5865e 7e21f059`

A.1.3 Elliptic curve over an extension field $F(p^m)$

NOTE An element τ in $F(p^m)$ is defined as $t_6x^6 + t_5x^5 + t_4x^4 + t_3x^3 + t_2x^2 + t_1x + t_0$ and denoted as $t_6\,p^6 + t_5\,p^5 + t_4\,p^4 + t_3\,p^3 + t_2\,p^2 + t_1\,p + t_0$; and

$\pi(kG) = s_6\,p^6 + s_5\,p^5 + s_4p^4 + s_3p^3 + s_2p^2 + s_1p + s_0$, where the x-coordinate of *kG* is $s_6x^6 + s_5x^5 + s_4x^4 + s_3x^3 + s_2x^2 + s_1x + s_0$.

p	`fffffbe3`
m	7
Irreducible polynomial	$X^7 - 2$
Equation of *E*	$y^2 = x^3 + ax + b$
a	`00000000 00000000 00000000 00000000 00000000 00000000 70f4ddff`
b	`00000000 00000000 00000000 00000000 00000000 00000000 f0219950`
Number of points on *E*	`ffffe335 01634cf3 7c3f5cb3 ef9f00df e0061b8f 3577b073 660fe6a7`
x-coordinate of *G*	`2b959c3c 3fc61f9c 39ce8ca5 4b481e06 611ea01b e9724da2 4b9b35ca`
y-coordinate of *G*	`289b23f3 92417552 e5dbbde5 848c1c07 12ff3a17 e4a54ccd 1d183c33`
n	`1 0000ea1a f1cc4a84 dc41fef8 819a69cf d77178f9 0fc7cd65`

len_n	`193 bits`
x_A	`db5d34b8 a3275237 0140d761 3eef0826 643e5cfd f1501a02`
x-coordinate of Y_A	`70f8a8f0 6d7dd557 1fe605c2 6d6a6620 044ac9b3 60942aef ac0040aa`
y-coordinate of Y_A	`f376f4df d4b4c458 c147c0c5 1d9c7fce 4baac5ae 9e55ba74 b89fd22e`
k	`9d6bafcc a8b6f20f 9ddf8a49 d3d23bd3 ff4ff4cd 235880d2`
Π (=$\pi(kG)$)	`69d647d5 220cbedb f0f79131 83ac9fbc cd99889b d9560df5 bd8e714e`
y-coordinate of kG	`d22d0b1a a930b7ff 370c4cb8 1ac7e80e 8ef14428 b93b346d 9cc3779f`
Message to be signed	`plaintext`
$M(=M_{rec})$ (M_{clr} is empty)	`70 6c61696e 74657874`
Length $L(=L_1)$ of truncated hash-token	`10 octets`
Recoverable length L_{rec}	`9 octets`
Non-recoverable length L_{clr}	`0 octet`
Truncated hash-token h (=short redundancy)	`8380 6905a33f 61c22c29`
Date input $d=h\|\|M$	`838069 05a33f61 c22c2970 6c61696e 74657874`
First part of signature r	`97f1c386 342bf20d d9791d5c 1e5acd27 14f805e5`
Second part of signature s	`316eae17 ecf987fb 45f2c54b 1eabc719 8e371b90`

A.2 Numerical examples for ECMR

NOTE Truncated hash token h is first L octets of SHA-1; and

Hash(R)=Hash($x_R||y_R$) for a point R on E, where x_R and y_R are the x-coordinate and y-coordinate of R, respectively.

A.2.1 Elliptic curve over a prime field

p	`ffffffff ffffffff ffffffff ffffffff ffff7c67`
Equation of E	$y^2 \equiv x^3 + ax + b \pmod{p}$

a	ffffffff ffffffff ffffffff ffffffff ffff7c64
b	26c1d102 82415e10 a4995e19 80b59224 d7120957
Number of points on E	ffffffff ffffffff ffffc748 a4eea1b0 dc8744b9
x-coordinate of G	1
y-coordinate of G	22e0d7c6 1eb0627b 334456c7 a50b77fd a9007da6
n	ffffffff ffffffff ffffc748 a4eea1b0 dc8744b9
len_n	160 bits
Signature key x_A	7ef23de1 f6ac01df e3cfef54 7294c5fb c7680781
x-coordinate of Y_A	c68f8677 5fc02772 20d88366 0ef7b826 681545be
y-coordinate of Y_A	e0e7b379 6ee6d783 126f1032 a5a51fc6 ca4e8712
Randomizer k	5ddb0488 6e56a287 77011847 373e4956 8d2062bf
x-coordinate of kG	9d2f37d2 d58ef02d 9c7fbf31 9b9caad3 aa22aad6
y-coordinate of kG	ad72d089 3c03b6b5 4873fbae 4d392c0b 9c1b218d
$\Pi = (Hash(kG))$	e26e9f6d afcec43c a73b6766 b556bd98 2902ea18
Message to be signed	TestVector
$M(=M_{rec})$ (M_{clr} is empty)	5465 73745665 63746f72
Length $L(=L_1)$ of truncated hash-token	10 octets
Recoverable length L_{rec}	10 octets
Non-recoverable length L_{clr}	0 octet
Truncated hash-token $h=Hash(\Pi \| \| M)$ (short redundancy)	65be 817d749d 5963226b
Data input $d=h\|\|M$	2c2bfdae f4e9ff44 b2915465 73745665 63746f72
First part of signature r	ce4562c3 5b273b78 15aa3303 c622ebfd 4a76856a
Second part of signature s	605eb507 94115773 8582e8cb ba521f96 b629a8f0

A.2.2 Elliptic curve over an extension field $F(2^m)$

Galois field $\boldsymbol{F}(2^{163})$ with the polynomial $x^{163}+x^7+x^6+x^3+1$.

NOTE This is a standard polynomial basis implementation.

Equation of E	$y^2 + xy = x^3 + ax^2 + b$
a	`1`
b	`2 0a601907 b8c953ca 1481eb10 512f7874 4a3205fd`
Number of points on E	`8 00000000 00000000 000525fc efce1825 48469866`
x-coordinate of G	`3 f0eba162 86a2d57e a0991168 d4994637 e8343e36`
y-coordinate of G	`0 d51fbc6c 71a0094f a2cdd545 b11c5c0c 797324f1`
n	`4 00000000 00000000 000292fe 77e70c12 a4234c33`
len_n	`163 bits`
x_A	`3 7ef23de1 f6ac01df e3cfef54 7294c5fb c7680781`
x-coordinate of Y_A	`3 848272e0 37635f05 b0831795 5739f83d 75347cf5`
y-coordinate of Y_A	`0 1806581d ee02dd0b cbf4c97b 1d1e01b8 b688d705`
k	`3 42f5feea 5ddb0488 6e56a287 77011847 373e4956`
x-coordinate of kG	`7 3d34aa43 38b8af8b 43c1d8ab dd4c13b4 6d3d967c`
y-coordinate of kG	`2 3a5f0a08 0e85cee3 bc1cc63a 49c80d67 9c826ad6`
$\Pi = (Hash(kG))$	`0 3c41061e dc9743c7 8f690a76 2a4454ac 1967d7ea`
Message to be signed	`TestVector`
$M(=M_{rec})$ (M_{clr} is empty)	`5465 73745665 63746f72`
Length $L(=L_1)$ of truncated hash-token	`10 octets`
Recoverable length L_{rec}	`10 octets`
Non-recoverable length L_{clr}	`0 octet`
Truncated hash token h $=Hash(\Pi \vert\vert M)$	`86e3 ae610087 3fc57752`

(short redundancy)

Date input d=h||M	0 86e3ae61 00873fc5 77525465 73745665 63746f72
First part of signature r	0 baa2a87f dc107c02 f83b5e13 593002c9 7a13b898
Second part of signature s	0 e36343f4 efc35f88 c7231148 384d3d22 14c78a43

A.2.3 Elliptic curve over an extension field $F(p^m)$

NOTE An element τ in $F(p^m)$ is defined as $t_4x^4 + t_3x^3 + t_2x^2 + t_1x + t_0$ and denoted as $t_4\ t_3\ t_2\ t_1\ t_0$; and Hash(R)=Hash($x_R||y_R$) for a point R on E, where x_R $(=u_4x^4 + u_3x^3 + u_2x^2 + u_1x + u_0)$ and y_R $(=v_4x^4 + v_3x^3 + v_2x^2 + v_1x + v_0)$ are the x-coordinate and y-coordinate of R, and are denoted as the bit string $u_4||u_3||u_2||u_1||u_0$ and $v_4||v_3||v_2||v_1||v_0$, respectively.

p	ffffff47
m	5
Irreducible polynomial	$X^5 - 2$
Equation of E	$y^2 = x^3 + ax + b$
a	00000000 00000000 00000000 00000000 ffffff44
b	39cd7fda f41a7fb5 488651a5 e362f27f b449e900
Number of points on E	fffffc63 000538e9 fc3bbe32 da01dc69 c2516d77
x-coordinate of G	00000000 00000000 00000000 00000000 00000002
y-coordinate of G	8a45f6c7 82f3c45e e2716ce9 26573f3f c5105399
n	fffffc63 000538e9 fc3bbe32 da01dc69 c2516d77
len_n	160 bits
x_A	798499dc 805ada44 4ecbdc7f 88428cfd 4c80236d
x-coordinate of Y_A	41fb68fc 86430725 d867d3f3 889e8aaa 4d8e0245
y-coordinate of Y_A	29d4b48b b3404529 7cab9bf3 46a8ba97 56f5e8bb
Randomizer k	61e9f344 7b67aad5 257998e9 ffb134d9 12542c5b
x-coordinate of kG	87f07602 957d4ed3 b1afe556 0c3ddcd2 cbdaf449
y-coordinate of kG	ef546ec1 ce9dc05f 824eb69f c5e59735 8f8d9aa7
Π = ($Hash$(kG))	2fca2dfd d292dd16 e89b4939 6d2e9b42 778c6a97

Message to be signed	`TestVector`
$M(=M_{rec})$ (M_{clr} is empty)	`5465 73745665 63746f72`
Length $L(=L_1)$ of truncated hash-token	`10 octets`
Recoverable length L_{rec}	`10 octets`
Non-recoverable length L_{clr}	`0 octet`
Truncated hash token h $=Hash(\Pi\|\|M)$	`b83b ee7be6b9 2f1b31e2`
(short redundancy)	
Date input $d=h\|\|M$	`b83bee7b e6b92f1b 31e25465 73745665 63746f72`
First part of signature r	`97f1c386 342bf20d d9791d5c 1e5acd27 14f805e5`
Second part of signature s	`316eae17 ecf987fb 45f2c54b 1eabc719 8e371b90`

A.3 Numerical examples for ECAO

NOTE Data input $d=Hash1(M_{rec})|| (Hash2(Hash1(M_{rec}))$ XOR $M_{rec})$;

Output of Hash1 is first L_2 octets of SHA1;

Output of Hash2 is first L_n-L_2 octets of SHA1; and

Output of Hash3(Oct_Str) is first L_n octets of SHA1(Oct_Str || 0x01) || SHA1(Oct_Str || 0x02)

A.3.1 Elliptic curve over a prime field

p	`ffffffff ffffffff ffffffff ffffffff 7fffffff`
Equation of E	$y^2 \equiv x^3 + ax + b \pmod p$
a	`ffffffff ffffffff ffffffff ffffffff 7ffffffc`
b	`1c97befc 54bd7a8b 65acf89f 81d4d4ad c565fa45`
Number of points on E	`01 00000000 00000000 0001f4c8 f927aed3 ca752257`
x-coordinate of G	`4a96b568 8ef57328 46646989 68c38bb9 13cbfc82`
y-coordinate of G	`23a62855 3168947d 59dcc912 04235137 7ac5fb32`
n	`01 00000000 00000000 0001f4c8 f927aed3 ca752257`
len_n	`161 bits`

L_n	21 octets
L_F	20 octets
x_A	a662ee37 61adf2bb ca1c1695 9b1de2a4 3d4cd1bf
x-coordinate of Y_A	10937f21 48c30dac 9ea14406 ca4017ff 8376890b
y-coordinate of Y_A	3da5278b 068546df 5b450d53 899fbc54 97650bb7
Randomizer k	722c12ad afa68860 758d37b1 f23f5dad c292bdcc
x_1(x-coordinate of kG)	d6373358 7ab16b99 136165ac 20d7ec2d 78ebd586
y_1(y-coordinate of kG)	6d186ffd f47cd274 3c30e62a eb38e2a7 9f9a4cda
Message to be signed	This is a test message.
Length of the long redundancy L_2	10 octets
L_{rec}	10 octets
L_{clr}	13 octets
M_{rec}	5468 69732069 73206120
M_{clr}	74 65737420 6d657373 6167652e
Data input d	60d9f478 530be6c6 6d4fe3c3 3866d1cf dcb4601d
First part of signature $r = d$ XOR x_1	b6eec720 29ba8d5f 7e2e866f 18b13de2 a45fb59b
$\Pi = Hash3(r \| M_{clr})$	2a 30cfe534 aed47912 b5f44eb2 bc3ebd25 79482906
Second part of signature $s = (k - x_A\Pi) \bmod n$	56128639 08261a0e 25b4936d bd10fde4 0c4de99c

A.3.2 Elliptic curve over an extension field $F(2^m)$

Galois field $F(2^{191})$ with the polynomial $x^{191}+x^9+1$.

NOTE This is a standard polynomial basis implementation; and

$\pi(kG) = 2^{190}s_{190}+2^{189}s_{189}+\ldots+2s_1+s_0$, where the x-coordinate of kG is $s_{190}x^{190}+s_{189}x^{189}+\ldots+s_1x+s_0$.

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

A	2866537b 67675263 6a68f565 54e12640 276b649e f7526267
B	2e45ef57 1f00786f 67b0081b 9495a3d9 5462f5de 0aa185ec
Number of points on E	80000000 00000000 00000000 09441d21 8720cf91 2777734a
x-coordinate of G	36b3daf8 a23206f9 c4f299d7 b21a9c36 9137f2c8 4ae1aa0d
y-coordinate of G	765be734 33b3f95e 332932e7 0ea245ca 2418ea0e f98018fb
N	40000000 00000000 00000000 04a20e90 c39067c8 93bbb9a5
len_n	191 bits
L_n	24 octets
L_F	24 octets
x_A	340562e1 dda332f9 d2aec168 249b5696 ee39d0ed 4d03760f
x-coordinate of Y_A	5de37e75 6bd55d72 e3768cb3 96ffeb96 2614dea4 ce28a2e7
y-coordinate of Y_A	55c0e0e0 2f5fb132 caf416ef 85b229bb b8e13520 03125ba1
Randomizer k	1ae15cc3 8afb928e 2517229e ffc84c9e 0d4040b5 10d8b509
x_1(x-coordinate of kG)	4a655feb f02d44b8 f7d6a1cd f8b0bc72 152c550f 1e438b1f
y_1(y-coordinate of kG)	477056f2 486d4bff 5c0f0227 b13a4091 b7eb0d1e 69f7cb63
Message to be signed	This is a test message.
Length of the long redundancy L_2	10 octets
L_{rec}	14 octets
L_{clr}	9 octets
M_{rec}	5468 69732069 73206120 74657374
M_{clr}	20 6d657373 6167652e
Data input d	a54d88e0 6612d820 bc3b5d36 c5cba7ca 4284234f bb9d68fe
First part of signature $r = d$ XOR x_1	ef28d70b 963f9c98 4bedfcfb 3d7b1bb8 57a87640 a5dee3e1
Π = Hash3(r||M_{clr})	69b29c47 6286ccb0 308e4013 dfd9d3b1 f6af4741 3b4f5c93

Second part of signature $s = (k - x_A\Pi) \bmod n$	`2213bd8f 6bc61b30 8c4f6920 5fd6c75b 830ca377 6a8929a2`

A.3.3 Elliptic curve over an extension field $F(p^m)$

NOTE An element τ in $F(p^m)$ is defined as $t_6x^6 + t_5x^5 + t_4x^4 + t_3x^3 + t_2x^2 + t_1x + t_0$ and denoted as $t_6\, t_5\, t_4\, t_3\, t_2\, t_1\, t_0$; and $\pi(kG) = s_6\, p^6 + s_5\, p^5 + s_4p^4 + s_3p^3 + s_2p^2 + s_1p + s_0$, where the x-coordinate of kG is $s_6x^6 + s_5x^5 + s_4x^4 + s_3x^3 + s_2x^2 + s_1x + s_0$.

p	`1ffffffd`
m	`7`
Irreducible polynomial for $F(p^m)$	$x^7 - 2$
Equation of E	$y^2 = x^3 + ax + b$
a	`-3`
b	`-85`
Number of points on E	`07ff fffac000 0179ffff c4f0003b 4cd1f99c bb56a550 7d7cf177`
x-coordinate of G	`02fc be50e66c 0a15d192 f162af3c 73d25c36 1f3d4cf0 70db4a40`
y-coordinate of G	`07cb 982bb2c2 7b3872a9 df8c4f70 4fdc15f3 e349443d c2389b96`
n	`00004001 2e3d9401 6c815a2e af1e63ad f3754413 ff83c6d9`
len_n	175 bits
L_n	22 octets
L_F	26 octets
x_A	`355a ba76e699 16c6e966 f57f058c 27ea084f d1497115`
x-coordinate of Y_A	`0468 b5501754 f1e15325 07a34b24 8be13fbb d7cb2618 a21401fb`
y-coordinate of Y_A	`002e ca9e2fcd 871a1bb6 545b3429 58a59d71 404a3600 88f02728`
Randomizer k	`3480 2b589986 73e6e757 0206cda4 43061cdd 1a48b303`
x_1(x-coordinate of kG)	`0159 2559fcf1 5c267035 35094c44 f4b8cc66 3ffd45fd bbf5de3b`
y_1(y-coordinate of kG)	`078e b7e2a59f f892966a 32c213ad 7af83b3f 22d585b5 42b0533d`

Message to be signed	`This is a test message.`
Length of the long redundancy L_2	`10 octets`
L_{rec}	`16 octets`
L_{clr}	`7 octets`
M_{rec}	`54686973 20697320 61207465 7374206d`
M_{clr}	`657373 6167652e`
Data input *d*	`9a9e 9983231e 5555d72b b5a41d15 e9c948fc 3e129dba 60f81c85`
First part of signature $r = d$ XOR x_1	`9bc7 bcdadfef 0973a71e 80ad5151 1d71849a 01efd847 db0dc2be`
Π = Hash3($r \| M_{clr}$)	`da6c f7941008 b512dce2 420e8004 23ccc6f9 7661aae9`
Second part of signature $s = (k - x_A \Pi) \bmod n$	`0b6e 1f2d1c9f 6288fc2a 35d55726 f9dfe7f3 9476e215`

A.4 Numerical examples for ECPV

NOTE KDF(*kG*) is leftmost 144 bits of sha1 ($\pi(kG) \| C$);

C is a octet string of length 4 with the value `00000001` as hexadecimal; and

SYM uses XOR.

A.4.1 Elliptic curve over a prime field

p	`ffffffff ffffffff ffffffff fffffffe ffffac73`
Equation of *E*	$y^2 \equiv x^3 + ax + b \pmod{p}$
a	`0 00000000 00000000 00000000 00000000 00000000`
b	`0 00000000 00000000 00000000 00000000 00000007`
Number of points on *E*	`1 00000000 00000000 0001b8fa 16dfab9a ca16b6b3`
x-coordinate of *G*	`3b4c382c e37aa192 a4019e76 3036f4f5 dd4d7ebb`
y-coordinate of *G*	`938cf935 318fdced 6bc28286 531733c3 f03c4fee`
n	`1 00000000 00000000 0001b8fa 16dfab9a ca16b6b3`

len_n	`161 bits`
Signature key x_A	`e6a080e0 b2a7a850 ba71d26c 9606669a 4b4a6c18`
x-coordinate of Y_A	`8f5788a5 c97ac053 984045f4 c9ff325d d60065ae`
y-coordinate of Y_A	`a5329d2a 721b5787 9c215323 37211f64 23e577da`
M_{rec}	`13 0b546573 74205573 65722031`
(M_{rec} is the DER encoding of the *PrintableString* value ”Test User 1”.)	
M_{clr}	`fa 2b0cbe77`
(M_{clr} is a random nonce which has no redundancy and will be in the clear)	
Pad (added redundancy of 40 bits, total redundancy is over 80 bits)	`05 05050505`
$d = Pad \|\| M_{rec}$	`0505 05050513 0b546573 74205573 65722031`
Randomizer k	`d8a0abc5 b7a4029a c232cbcd a16819e1 b715f9f4`
x-coordinate of $R=kG$	`1af46ec4 e95daede 056bfa3b 370075f6 3cb2c34f`
y-coordinate of R	`c324f1ba 5324383c 371278d5 f3fe4fe2 9373702c`
σ = KDF(kG)	`20b9 f76f3b33 6a805e02 92ed0b71 c9aaa767`
r= d XOR σ	`25bc f26a3e20 61d43b71 e6cd5e02 acd88756`
h=sha1($r\|\|M_{clr}$)	`1b5361a3 9bc7ca45 6bbb6f2b f80e4e31 a01b4a1b`
$s = k\text{-}x_A h \bmod n$	`735171855371469914412919293672210175623097579946`(digit)

A.4.2 Elliptic curve over an extension field $F(2^m)$

Galois field $F(2^{163})$ with the polynomial $x^{163}+x^7+x^6+x^3+1$.

NOTE This is a standard polynomial basis implementation; and

$\pi(kG) = 2^{162}s_{162} + 2^{161}s_{161} + \ldots + 2s_1 + s_0$, where the x-coordinate of kG is $s_{162}x^{162} + s_{161}x^{161} + \ldots + s_1x + s_0$.

Equation of E	$y^2 + xy = x^3 + ax^2 + b$
a	`00 00000000 00000000 00000000 00000000 00000001`

b	`00 00000000 00000000 00000000 00000000 00000001`
Number of points on E	`08 00000000 00000000 00040211 45C1981b 33f14bde`
x-coordinate of G	`02 fe13c053 7bbc11ac aa07d793 de4e6d5e 5c94eee8`
y-coordinate of G	`02 89070fb0 5d38ff58 321f2e80 0536d538 ccdaa3d9`
n	`04 00000000 00000000 00020108 a2e0cc0d 99f8a5ef`
len_n	`163 bits`
x_A	`03 a41434aa 99c2ef40 c8495b2e d9739cb2 155a1e0d`
x-coordinate of Y_A	`03 7d529fa3 7e42195f 10111127 ffb2bb38 644806bc`
y-coordinate of Y_A	`04 47026eee 8b34157f 3eb51be5 185d2be0 249ed776`
M_{rec}	`13 0b546573 74205573 65722031`
(M_{rec} is the DER encoding of the *PrintableString* value "Test User 1".)	
M_{clr}	`fa 2b0cbe77`
(M_{clr} is a random nonce which has no redundancy and will be in the clear)	
Pad (added redundancy of 40 bits, total redundancy is over 80 bits)	`05 05050505`
$d = Pad \| M_{rec}$	`0505 05050513 0b546573 74205573 65722031`
Randomizer k	`a40b301c c315c257 d51d4422 34f5aff8 189d2b6c`
x-coordinate of $R=kG$	`04 994d2c41 aa30e529 52b0a94e c6511328 c502da9b`
y-coordinate of R	`03 1fc936d7 3163b858 bbc5326d 77c19839 46405264`
σ = KDF(kG)	`d5bd 8bd7309d 020d5946 1367e723 a3b63d79`
$r = d$ XOR σ	`d0b8 8ed2358e 09593c35 6747b250 c6c41d48`
h=sha1($r \| M_{clr}$)	`db9b5ebe 6b7b9690 54f9aecb 52b54a19 df4495ae`
$s = k - x_A h \bmod n$	`2748261816569194283991299038913308940368752275658(digit)`