Accredited Standards Committee X9 **May 31, 2001**
Title: X9-Financial Services
Accredited by the
American National Standards Institute

Working Draft

AMERICAN NATIONAL STANDARD

X9.92 - 2001

Public Key Cryptography For The Financial Services Industry:

PV-Digital Signature Scheme Giving Partial Message Recovery ©

Notice -- This document is a draft document. It has not yet been processed through the consensus procedures of X9 and ANSI.

Many changes which may greatly affect its contents can occur before this document is completed. The working group may not be held responsible for the contents of this document.

Implementation or design based on this draft is at the risk of the user. No advertisement or citation implying compliance with a "Standard" should appear as it is erroneous and misleading to so state.

Copies of the draft proposed American National Standard will be available from the X9 Secretariat when the document is finally announced for two months public comment. Notice of this announcement will be in the trade press.

Secretariat: American Bankers Association
Standards Department
1120 Connecticut Ave., N.W.
Washington, DC 20036

Contents

ANSI X9.92 – 2001,

Public Key Cryptography for The Financial Service Industry: PV-Digital Signature Scheme Giving Partial Message Recovery (PVS)

1 Scope

This standard defines methods for digital signature generation and verification for the protection of messages and data using Pintsov-Vanstone elliptic curve Digital Signature Algorithm giving partial message recovery (PVS).

PVS is a signature scheme with low message expansion (overhead) and variable length recoverable and non-recoverable message parts. PVS is ideally suited for short messages, yet is flexible enough to handle messages of any length.

The PVS shall be used in conjunction with an ANSI-X9-approved hash function, an ANSI-X9-approved key derivation function, and an ANSI-X9-approved symmetric encryption scheme. In addition, this PVS Standard provides the criteria for checking the message redundancy.

Supporting examples are also provided.

2 Definitions, Abbreviations and References

2.1 Definitions and Abbreviations

addition rule
An *addition rule* describes the addition of two elliptic curve points P_1 and P_2 to produce a third elliptic curve point P_3.

base point (*G*)
A selected point on an elliptic curve of large prime order n.

basis
A representation of the elements of the finite field F_{2^m}. Two special kinds of basis are *polynomial basis* and *normal basis.*

bit string
A bit string is an ordered sequence of 0's and 1's.

characteristic 2 finite field
A finite field containing 2^m elements, where $m \geq 1$ is an integer. In this Standard, only characteristic 2 fields containing 2^m elements with m prime are used.

cryptographic hash function
A (mathematical) function which maps values from a large (possibly very large) domain into a smaller range. The function satisfies the following properties:

1. it is computationally infeasible to find any input that maps to any pre-specified output;

2. it is computationally infeasible to find any two distinct inputs that map to the same output.

cryptographic key (key)
A parameter that determines the operation of a cryptographic function such as:

1. the transformation from plaintext to ciphertext and vice versa,

2. the synchronized generation of keying material,

3. a digital signature computation or verification.

cryptographic protocol
A cryptographic scheme in which an ordered sequence of sets of data is passed between two entities during an ordinary operation of the scheme.

cryptographic scheme
A cryptographic scheme consists of an unambiguous specification of a set of transformations capable of providing a cryptographic service when properly implemented and maintained.

cryptography
The discipline that embodies principles, means and methods for the transformation of data in order to hide its information content, prevent its undetected modification, prevent its unauthorized use, or a combination thereof.

cyclic group
The group of points $E(F_q)$ is said to be *cyclic* if there exists a point $P \in E(F_q)$ of order n, where $n = \#E(F_q)$. In this case, $E(F_q) = \{kP: 0 \le k \le n-1\}$, i.e. $E(F_q)$ can be expressed as the set of all scalar multiples of P.

data confidentiality
The assurance provided to entity U that data is unintelligible to entities other than U and V.

data integrity
The assurance provided to entity U that data has not been modified by entities other than U and V.

data origin authentication
The assurance provided to entity U that data is from V.

digital signature
The result of a cryptographic transformation of data that, when properly implemented, provides the services of:

1. origin authentication,

2. data integrity, and

3. signer non-repudiation.

EC
Elliptic curve.

elliptic curve
An *elliptic curve* over F_q is a set of points that satisfy a certain equation specified by 2 parameters a and b, which are elements of the field F_q.

elliptic curve key pair (Q, d)
Given particular elliptic curve domain parameters, an *elliptic curve key pair* consists of an elliptic curve public key (Q) and the corresponding elliptic curve private key (d).

elliptic curve private key (d)
Given particular elliptic curve domain parameters, an *elliptic curve private key, d,* is a statistically unique and unpredictable integer in the interval [1, n−1], where n is the prime order of the base point G.

elliptic curve public key (Q)
Given particular elliptic curve domain parameters, and an elliptic curve private key d, the corresponding *elliptic curve* public *key, Q,* is the elliptic curve point $Q = dG$, where G is the base point. Note that Q will never equal ○, since $1 \le d \le n-1$.

elliptic curve domain parameters
Elliptic curve domain parameters are comprised of a field size q, an indication of the basis used (in the case $q = 2^m$), an optional SEED, two elements a, b in F_q that define an elliptic curve E over F_q, a point $G = (x_G, y_G)$ of prime order in $E(F_q)$, the order n of G, and the cofactor h.

elliptic curve point
If E is an elliptic curve defined over a field F_q, then an *elliptic curve point* P is either: a pair of field elements (x_p, y_p) (where $x_p, y_p \in F_q$) such that the values $x = x_p$ and $y = y_p$ satisfy the equation defining E, or a special point ○ called the *point at infinity*. ○ is the identity element of the elliptic curve group.

encryption scheme
An encryption scheme is a cryptographic scheme capable of providing data confidentiality.

hash function
See cryptographic hash function.

hash value
The result of applying a cryptographic hash function to a bit string.

irreducible binary polynomial
A binary polynomial $f(x)$ is *irreducible* if it cannot be factored into a product of two or more binary polynomials, each of degree less than the degree of $f(x)$.

key
See cryptographic key.

key derivation function
A key derivation function is a function that takes as input a shared secret value and outputs keying data suitable for later cryptographic use.

keying data
Data suitable for use as cryptographic keys.

keying material
The data (e.g., keys, certificates and initialization vectors) necessary to establish and maintain cryptographic keying relationships.

non-repudiation
The assurance provided to entity U that U is able to prove to a third party that data is from V.

octet
An *octet* is a bit string of length 8. An octet is represented by a hexadecimal string of length 2. The first hexadecimal digit represents the four leftmost bits of the octet, and the second hexadecimal digit represents the four rightmost bits of the octet. For example, 9*D* represents the bit string 10011101. An octet also represents an integer in the interval [0, 255]. For example, 9*D* represents the integer 157.

octet string
An octet string is an ordered sequence of octets.

order of a curve
The *order of an elliptic curve E* defined over the field F_q is the number of points on E, including ○. This is denoted by $\#E(F_q)$.

order of a point
The *order of a point P* is the smallest positive integer n such that $nP = ○$ (the point at infinity).

owner
The entity whose identity is associated with a private/public key pair.

pentanomial
A polynomial of the form $x^m + x^{k3} + x^{k2} + x^{k1} + 1$, where $1 \le k1 < k2 < k3 \le m-1$.

prime finite field
A finite field containing p elements, where p is an odd prime number.

private key
In an asymmetric (public-key) system, that key of an entity's key pair which is known only by that entity.

protocol
See cryptographic protocol.

public key
In an asymmetric key system, that key of an entity's key pair which is publicly known.

PVS
Pintsov-Vanstone Digital Signature Algorithm giving partial message recovery.

reduction polynomial
The irreducible binary polynomial $f(x)$ of degree m that is used to determine a polynomial basis representation of F_{2^m}.

scalar multiplication
If k is a positive integer, then kP denotes the point obtained by adding together k copies of the point P. The process of computing kP from P and k is called *scalar multiplication*.

signature scheme
A signature scheme is a cryptographic scheme capable of providing data origin authentication, data integrity, and non-repudiation.

symmetric cryptographic scheme

A cryptographic scheme in which each transformation is controlled by the same key.

trinomial
A polynomial of the form $x^m + x^k + 1$, where $1 \leq k \leq m-1$.

valid elliptic curve domain parameters
A set of elliptic curve domain parameters that have been validated using the method specified in Section 5.1.1.2 or Section 5.1.2.2 of ANSI X9.62 [3].

XOR
Bitwise exclusive-or (also bitwise addition mod 2) of two bit strings of the same bit length.

***x*-coordinate**
The *x-coordinate* of an elliptic curve point, $P = (x_p, y_p)$, is x_p.

***y*-coordinate**
The *y-coordinate* of an elliptic curve point, $P = (x_p, y_p)$, is y_p.

2.2 Symbols and Notation

[*X*] Indicates that the inclusion of the bit string or octet string *X* is optional.

[*x*, *y*] The interval of integers between and including *x* and *y*.

$\lceil x \rceil$	Ceiling: the smallest integer $\geq x$. For example, $\lceil 5 \rceil = 5$ and $\lceil 5.3 \rceil = 6$.
$\lfloor x \rfloor$	Floor: the largest integer $\leq x$. For example, $\lfloor 5 \rfloor = 5$ and $\lfloor 5.3 \rfloor = 5$.
x mod n	The unique remainder r, $0 \leq r \leq n - 1$, when integer x is divided by n. For example, 23 mod 7 = 2.
$x \equiv y \pmod{n}$	x is congruent to y modulo n. That is, (x mod n) = (y mod n).
$x \times y$	x multiplied by y. x and y may be integers, field elements, or elliptic curve points and multiplication is performed in the corresponding group.
a, b	Elements of F_q that define an elliptic curve E over F_q.
d	Elliptic curve private key.
E	An elliptic curve over the field F_q defined by a and b.
$E(F_q)$	The set of all points on an elliptic curve E defined over F_q and including the point at infinity ○.
$\#E(F_q)$	If E is defined over F_q, then $\#E(F_q)$ denotes the number of points on the curve (including the point at infinity ○). $\#E(F_q)$ is called the order of the curve E.
F_{2^m}	The finite field containing $q = 2^m$ elements, where m is a positive integer.
F_p	The finite field containing $q = p$ elements, where p is a prime.
F_q	The finite field containing q elements. For this Standard, q shall either be an odd prime number ($q = p$, $p > 3$) or a power of 2 ($q = 2^m$).
G	A distinguished point on an elliptic curve called the *base point* or *generating point*.
gcd(x, y)	The greatest common divisor of integers x and y.
h	$h = \#E(F_q)/n$, where n is the order of the base point G. h is called the *cofactor*.
$\log_2 x$	The logarithm of x to the base 2.
m	The *degree* of the finite field F_{2^m}.
mod	Modulo.
mod n	Arithmetic modulo n.

n	The order of the base point G. For this Standard, n shall be greater than 2^{160} and $4\sqrt{q}$, and shall be a prime number. n is the primary security parameter. See Annex H for more information.
$\mathcal{O}$	A special point on an elliptic curve, called the point at infinity. This is the additive identity of the elliptic curve group.
p	An odd prime number.
P	An EC point.
q	The number of elements in the field F_q.
Q	Elliptic Curve public key.
t	The length of a field element in bits; $t = \lceil \log_2 q \rceil$. In particular, if $q = 2^m$, then a field element in F_{2^m} can be represented as a bit string of bit length $t = m$.
x_p	The x-coordinate of a point P.
$\|\|X\|\|$	Length in octets of the octet string X.
$X\|\|Y$	Concatenation of two strings X and Y. Both X and Y are either bit strings, or octet strings.
$X \oplus Y$	Bitwise exclusive-or (also bitwise addition mod 2) of two bit strings X and Y of the same bit length.
y_p	The y-coordinate of a point P.
Z	A shared secret value.
Z_p	The set of integers modulo p, where p is an odd prime number.

Subscript notation is used to indicate the base in which a particular value is being expressed if there is some possibility of ambiguity. For example, 01_{16} denotes that the value 01 is written in hexadecimal.

2.3 Normative References

The following standards contain provisions which, through reference in this text, constitute provisions of this American National Standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this American National Standard are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below.

Accredited Standards Committee X9 (ASC X9) maintains a register of currently valid financial industry standards.

1. ANSI X9.30.2-1997: *Public Key Cryptography for the Financial Services – Part 2: The Secure Hash Algorithm (SHA-1)*.

2. ANSI X9.52-1998: *Triple Data Encryption Algorithm, Modes of Operation*.

3. ANSI X9.62-1999: *Public Key Cryptography For The Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA).*

4. ANSI X9.63-1999: *Public Key Cryptography For The Financial Services Industry: Key Agreement and Key Transport Using Elliptic Curve Cryptography*.

5. ANSI X.9.91-2001: *Advanced Encryption Standard for the Financial Services Industry*.

3 Application

3.1 General

When information is transmitted from one party to another, the recipient may desire to know that the information has not been altered in transit. Furthermore, the recipient may wish to be certain of the originator's identity. The use of public-key cryptography digital signatures can provide assurance (1) of the identity of the signer, and (2) that the received messages has not been altered during transmission.

A digital signature is an electronic analog to a written signature. The digital signature may be used in proving to a third party that the information was, in fact, signed by the claimed originator. Unlike their written counterparts, digital signature also verify the integrity of information. Digital signatures may also be generated for stored data and programs so that the integrity of the data and programs may be verified at any later time.

3.2 The Use of the PVS Algorithm

The PVS is used by a *signatory* to generate a digital signature on data and by a *verifier* to verify the authenticity of the signature and recover part of the message from the signature. Each signatory has a public and private key. The private key is used in the signature generation process, and the public key is used in the signature verification process.

In the PVS scheme, part of the message is "masked" and can be "demasked" later to "recover" it in readable form. The signature still guarantees the integrity of the entire message. The process flow from the application's viewpoint is basically the same as that of ECDSA in ANSI X9.62 [3]: The signer will use the signing transformation to compute the signature on the message. The verifier, after being sent the (possibly empty)

non-recoverable message part and the signature, will check the validity of the signature using the verifying transformation and recover the recoverable part of the message.

There MAY be a concern regarding use of the PVS scheme due to the partial masking of message.

4 Mathematical Conventions

The mathematics for Elliptic Curve is specified in Section 4 of ANSI X9.62 [3].

5 Cryptographic Ingredients

This Section specifies the various cryptographic ingredients that are required by the PVS.

5.1 Cryptographic Hash Functions

This section specifies the cryptographic hash functions that shall be used by the PVS.

The hash function will be used to calculate the hash value associated with a bit string.

The hash function will be used by the key derivation function specified in Section 5.2 and by PVS scheme in Section 6.

Any ANSI-X9-approved hash function which offers 80 bits of security or more may be used, i.e., any ANSI-X9-approved hash function whose output is 160 bits or more. Possibilities therefore include the hash function SHA-1. SHA-1 is specified in ANSI X9.30 [1].

Hash values will be calculated as follows:

Prerequisites: The prerequisite for the operation of the hash function is that an ANSI-X9-approved hash function has been chosen. The maximum length of the input to the hash function is denoted by *maxhashlen* and the length of the output of the hash function by *hashlen.*

Input: The input to the hash function is a bit string *M* of length less than *maxhashlen* bits.

Actions: Calculate the hash value *H* corresponding to *M* using the established hash function. This process is denoted by:

$$H=Hash(M)$$

where *Hash* denotes the established hash function.

Output: The bit string *H* of length *hashlen* bits.

Note that the hash function operates on bit strings of length less than *maxhashlen* bits. For example, SHA-1 operates on bits string of length less than 2^{64} bits. In the sequel it is assumed that all hash function calls are indeed on bit strings of length less than *maxhashlen* bits. Any scheme attempting to call the hash function on a bit string of length greater than or equal to *maxhashlen* bits shall output 'invalid' and stop.

5.2 Key Derivation Functions

This section specifies the key derivation functions that shall be used by the PVS.

The key derivation function will be used to derive keying data from a shared secret bit string.

Any ANSI-X9-approved key derivation function may be used. Possibilities therefore include the key derivation function *KDF* specified in Section 5.6.3 of ANSI X9.63 [4].

Keying data shall be derived as follows:

Prerequisites: The prerequisite for the operation of the key derivation function is that an ANSI-X9-approved hash function has been chosen as specified in Section 5.1.

Input: The input to the key derivation function is:

1. A bit string *Z* which is the shared secret value.

2. An integer *keydatalen* which is the length in bits of the keying data to be generated. *keydatalen* shall be less than $hashlen \times (2^{32}-1)$.

3. (Optional) A bit string *SharedInfo* which consists of some data shared by the two entities intended to share the secret value *Z*, the default value for *SharedInfo* is the empty string.

Actions: Calculate the keying data *KeyData* corresponding to *Z* and *SharedInfo* (optional) using the established key derivation function. This process is denoted by:

KeyData=KDF(Z, SharedInfo)

Where *KDF* denotes the established key derivation function.

Output: The bit string *KeyData* of length *keydatalen* bits.

Note that the key derivation function produces keying data of length less than $hashlen \times (2^{32}-1)$ bits. In the sequel we assume that all key derivation function calls are indeed for bit strings of length less than $hashlen \times (2^{32}-1)$ bits. Any scheme attempting to

call the key derivation function for a bit string of length greater than or equal to *hashlen*×(2^{32}–1) bits shall output 'invalid' and stop.

5.3 Symmetric Encryption Schemes

This section specifies the symmetric encryption schemes that shall be used by the PVS.

The symmetric encryption scheme will be used to encrypt and to decrypt the recoverable message part in Section 5.4.

Any ANSI-X9-approved symmetric encryption schemes may be used. Possibilities therefore include the symmetric encryption scheme Triple-DES specified in ANSI X9.52 [1] and the symmetric encryption scheme AES specified in ANSI X.9.91 [4]. When the length of the message is larger than the block size of the encryption scheme, the Cipher Block Chaining (CBC) mode of operation specified in ANSI X.9.52 [1] should be used. When CBC mode is used, the Initialization Vector (IV) should be chosen as the zero vector.

Prerequisites: The prerequisites for the operations of the symmetric encryption and decryption scheme is that an ANSI-X9-approved symmetric encryption scheme has been chosen.

5.3.1 Symmetric Encryption

Input: The input to the symmetric encryption is:

1. A message *M*, which is a bit string of length *mlen* bits.

2. A key *KeyData,* which is bit string of length *keydatalen* bits.

Actions: Calculate the ciphertext *C* corresponding to *M* and *KeyData* using the encryption scheme. This process is denoted by:

C=encrypt(M, KeyData)

where *encrypt* denotes the established encryption scheme.

Output: The bit string *C*.

5.3.2 Symmetric Decryption

Input: The input to the symmetric decryption is:

1. A cipher *C*, which is a bit string of length *clen* bits.

2. A key *KeyData,* which is bit string of length *keydatalen* bits.

Actions: Calculate the plaintext *M* corresponding to *C* and *KeyData* using the decryption scheme. This process is denoted by:

M=decrypt(C, KeyData)

Where *decrypt* denotes the established decryption scheme.

Output: The bit string *M*.

5.4 Message Encoding and Decoding Methods

This section specifies the encoding operation and decoding operation primitives that shall be used to generate PVS signatures and to verify PVS signatures.

The method is parameterized by the following choices:

1. An integer *padOctlen* between 1 and 255 inclusive.

2. The method for combining a bit string *Z* with the (padded) recoverable message part, which shall be either:

   2.1. A stream cipher based on an ANSI-X9-approved key derivation function (specified in Section 5.2).

   2.2. An ANSI-X9-approved symmetric encryption scheme (specified in Section 5.3) based on an ANSI-X9-approved key derivation function (specified in Section 5.2).

   Let *KDF* denote the key derivation function chosen, and *keydatalen* denote the length of the output of the key derivation function.

NOTE:

1. This method cannot produce message representatives shorter than 8×*padOctLen* bits in length.

2. This method is, in principle, amenable to "single-pass" processing since the non-recoverable message part is not processed at all by this method (and in fact PVS, which employs this method, supports single-pass processing of the non-recoverable message part if the message representative *C* is transmitted *before* the non-recoverable message part).

3. The encoding method is also implicitly parameterized by any key derivation parameters of the key derivation function. As a default the key derivation parameter shall be the empty string.

5.4.1 Encoding Operation

Input: The input to the encoding operation is:

1. A recoverable message part, which is a bit string *M* of length *mlen* bits.

2. A bit string *Z*.

Actions: Compute the message representative *r* of *M* as follows:

1. Convert *padOctlen* to a bit string P_1 of length 8 bits with the primitive specified in Section 4.3.1 of ANSI X9.62 [3].

2. Let P_2 be the bit string formed from the bit string P_1 repeated *padOctlen* times. (Thus P_2 will have length $8 \times padOctlen$ bits.)

3. Let $T=P_2||M$.

4. If the method for combining the bit string *Z* with the recoverable message part is a stream cipher based on a key derivation function:

   4.1. Apply the key derivation function *KDF* to the bit string *Z* to produce a bit string *KeyData* of length $keydatalen=mlen+8\times padOctlen$ bits.

   4.2. Compute the message representative $r=T \oplus KeyData$.

5. If the method for combining the bit string *Z* with the recoverable message part is a symmetric encryption scheme combined with a key derivation method:

   5.1. Apply the key derivation function *KDF* to the bit string *Z* to produce a bit string *KeyData* of length *keydatalen* bits.

   5.2. Encrypt *T* under the key *KeyData* with the symmetric encryption scheme to produce the message representative

   $$r = encrypt(T,KeyData)$$

   according to the encryption scheme specified in Section 5.3.

The above process is denoted by:

$$r = MEM(M,Z).$$

Output: The bit string *r* as the message representative.

NOTE—The padding bit string P_2 is 01_{16} if *padOctLen* = 1, 0202_{16} if *padOctLen* = 2, 030303_{16} if *padOctLen* = 3, 04040404_{16} if *padOctLen* = 4, 0505050505_{16} if *padOctLen* = 5, and so on.

5.4.2 Decoding Operation

Input: The input to the decoding operation is:

1 The message representative, which is a bit string r of length $clen$($\geq 8\times padOctlen$) bits.

2 A bit string Z.

Actions: Compute the recoverable message part M as follows:

1. If the method for combining the bit string Z with the recoverable message part is a stream cipher based on a key derivation function:

   1.1. Apply the key derivation function KDF to the bit string Z to produce a bit string $KeyData$ of length $clen$ bits.

   1.2. Compute $T=r\oplus KeyData$.

2. If the method for combining the bit string Z with the recoverable message part is a symmetric encryption scheme combined with a key derivation method:

   2.1. Apply the key derivation function KDF to the bit string Z to produce a bit string $KeyData$ of length $keydatalen$ bits.

   2.2. Decrypt r under the key $KeyData$ with the symmetric decryption scheme to recover T:

   $$T=decrypt(r,KeyData)$$

   according to the decryption scheme specified in Section 5.3. (If the decryption operation outputs "error", output "invalid" and stop.)

3. Let $tlen$ be the length of T in bits. If $tlen<8\times padOctlen$, output "invalid" and stop.

4. Let P_1 be the leftmost 8 bits of T.

5. Convert P_1 to an integer $padOctlen'$ with the primitive specified in Section 4.3.2 of ANSI X9.62 [3].

6. If $padOctlen\neq padOctlen'$, output "invalid" and stop.

7. If $padOctlen\geq 2$, verify that each octet of the next $padOctlen-1$ octets after the first octet of T is equal to P_1. If not, output "invalid" and stop.

8. Let M be the rightmost $tlen-8\times padOctlen$ bits of T.

The above process is denoted by:

$$M = MEM^{-1}(r,Z).$$

Output: The bit string *M*.

5.5 Message Redundancy Criteria

One of the following message redundancy types is selected by users:

1. The inherent redundancy that shall be contained in a message.
2. The *PadOctLen* octets added redundancy as specified in Section 5.4.
3. Or both.

The amount of total redundancy is a security parameter and shall be chosen to achieve security objectives. It is recommended that a minimum redundancy criteria which is consistent with 80 bits of symmetric key security be used. The nature of the inherent redundancy may be anything agreed upon and able to be checked by the communicating parties, for example, many messages are known to be in ASCII or have specific formats.

6 Pintsov-Vanstone Signature (PVS) Algorithm

This section specifies the Pintsov-Vanstone digital signature algorithm with partial message recovery.

Implementations with different internal representations that produce equivalent results are allowed.

The signature scheme will be used as follows. The signatory will use the signing transformation to compute a signature on some data. The verifier, after being sent the non-recoverable message part and the signature, will recover the recoverable message part and check the validity of the signature using the verifying transformation.

Prerequisites: The following are the prerequisites for the use of PVS:

1. The signatory has chosen a message redundancy criteria.
2. (Optional) The signatory has an identity *I* which is a bit string. The identity could be embedded in the non-recoverable message *V* of the input (see Section 6.1).
3. The signatory has established a set of EC domain parameters for its use with the PVS signing transformation consisting of *q, a, b, G, n,* and *h*, along with an indication of the basis used if $q=2^m$. The parameters shall have been generated using the parameter generation primitives in Sections 5.1.1.1 and 5.1.2.1 of ANSI X9.62 [3]. Furthermore, the parameters shall have been validated by their generator and may optionally have been validated by the signatory (if the signatory is not the generator) using the parameter validation primitives in Sections 5.1.1.2 and 5.1.2.2 of ANSI X9.62 [3].

4. The signatory has chosen an ANSI-X9-approved hash function offering at least 80 bits of security for its use with the PVS signing transformation. (One possibility is SHA-1.) Let *Hash* denote the hash function chosen, and *hashlen* denote the length of the output of the hash function.

5. The signatory has chosen a message encryption (decryption) method described in Sections 5.3 and 5.4. Let *MEM* be the message encryption method chosen and MEM^{-1} be the corresponding message decryption method respectively.

6. The signatory has chosen an EC key pair (*d,Q*) associated with the EC domain parameters *q, a, b, G, n,* and *h* for its use with the PVS signing transformation. The key pair shall have been chosen using the key pair generation primitive in Section 5.2.1 of ANSI X9.62 [3].

7. The verifier has obtained in an authentic manner the message redundancy criteria, the elliptic curve domain parameters, the hash function, the message encryption (decryption) method, and the elliptic curve public key chosen by the signatory. Furthermore when an application is deemed to require validation of parameters by the verifier, the verifier shall have validated the EC domain parameters using the parameter validation primitives in Sections 5.1.1.2 and 5.1.2.2 of ANSI X9.62 [3], and when an application is deemed to require validation of public keys by the verifier, the verifier shall have validated the EC public key using the public key validation primitive in Section 5.2 of ANSI X9.62 [3]. As an alternative to the verifier doing the validation itself, it can receive assurance of validation from a trusted third party, (e.g., a CA).

6.1 Signature Generation

This section describes the PVS signature generation process.

Input: The input to the signature process is:

1. A message *(M,V)* which is a pair of bit string to be signed, *M* is called a recoverable message part and *V* is called a non-recoverable message part.

2. A valid set of elliptic curve domain parameters.

3. An elliptic curve private key, *d,* associated with the elliptic curve domain parameters.

Actions: Compute a signature on (*M,V*) consisting of a bit string *r* and an integer *s* as follows:

1. Verify that the message (consisting of the recoverable message part *M* and the non-recoverable message part *V*) meets the selected message redundancy criteria. If it does not, output "error". Note that the specific details of what meets the message redundancy criteria is outside the scope of this standard and is determined by the signatory and agreed to by the verifier.

2. Select a statistically unique and unpredictable integer k in the interval $[1,n-1]$. It is acceptable to use a random or pseudorandom number. If a pseudorandom number is used, it shall be generated using one of the procedures in an ANSI-X9-approved standard.

   If a pseudorandom generation method is used, the seed values used in the generation of k may be either be determined by internal means, be supplied by the caller, or both—this is an implementation choice. In all cases, the seed values have the same security requirements as the private key value. That is, they must be protected from unauthorized disclosure and be unpredictable.

   If the implementation allows a seed supplied by the caller, then the physical security of the device is of utmost importance. This is because if an adversary gained access to the signature generation device and were able to generate a signature with a seed of its choice for the per-message secret k, then the adversary could easily recover the private key.

3. Compute the elliptic curve point $(x_1, y_1)=kG$.

4. Convert the field element x_1 to a bit string Z with the primitive specified in Section 4.3.3 of ANSI X9.62 [3].

5. Use the selected message encoding method to compute: $r=MEM(M,Z)$.

6. Use the selected hash function to compute: $H=Hash(r||V)$ as specified in Section 5.1. H is a bit string of length *hashlen* bits.

7. Derive an integer e from H as follows:

   7.1. If $\lceil \log_2 n \rceil \geq hashlen$, set $E=H$. Otherwise set E equal to the leftmost $\lceil \log_2 n \rceil$ bits of H.

   7.2. Convert the bit string E to an integer e with the primitive specified in Section 4.3.2 of ANSI X9.62 [3].

8. Compute $s=k-de \bmod n$.

Output: The non-recoverable message part V, the message representative r, and the integer s with $1 \leq s \leq n-1$.

6.2 Signature Verification

The transformation specified as follows shall be used by the verifier to verify a purported signature and to recover the recoverable message part**.**

Input: The input to the verifying transformation is:

1. The received signature (V',r',s') where V' is a bit string of the non-recoverable message part, r' is a bit string of the recoverable message part representative, and s' is an integer..

2. A valid set of elliptic curve domain parameters.

3. A valid public key, Q, associated with the elliptic curve domain parameters.

Actions: Recover the signed message and verify the purported signature as follows:

1. If s' is not an integer in the interval $[1,n-1]$, then reject the signature and output "invalid".

2. Use the selected hash function to compute: $H'=Hash(r'||V')$ as specified in Section 5.1. H' is a bit string of length *hashlen* bits.

3. Derive an integer e' from H' as follows:

   3.1. If $\lceil \log_2 n \rceil \geq hashlen$, set $E'=H'$. Otherwise set E' equal to the leftmost $\lceil \log_2 n \rceil$ bits of H'.

   3.2. Convert the bit string E' to an integer e' with the primitive specified in Section 4.3.2 of ANSI X9.62 [3].

4. Compute the elliptic curve point $(x'_1,y'_1)=s'G+e'Q$. (If (x'_1,y'_1) is the point at infinity, then reject the signature and output "invalid".)

5. Convert the field element x'_1 to a bit string Z' with the primitive specified in Section 4.3.3 of ANSI X9.62 [3].

6. Use the selected message decoding method to compute: $M'=MEM^{-1}(r',Z')$.

7. Verify that the recovered message (consisting of the recoverable message part M' and the non-recoverable message part V') meets the redundancy criteria. If it does, output 'valid', otherwise output 'invalid'. Note that the specific details of what meets the message redundancy criteria is outside the scope of this standard and is determined by the signatory and agreed to by the verifier.

Output: An indication of whether the purported signature is valid or not---either 'valid' or 'invalid'. If the result is 'valid', then also output the recoverable message part M'.

7 ASN.1 Syntax

This section provides the syntax for the PVS according to Abstract Syntax Notation One (**ASN.1**). The syntax for finite field identification, finite field elements, elliptic curve points, elliptic curve domain parameters, and public keys can be found in ANSI X9.62

[3]. The syntax for key derivation functions can be found in ANSI X9.63 [4]. The syntax for SHA-1 can be found in ANSI X9.30.2 [1].

The object identifier **ansi-X9-92** represents the root of the tree containing all object identifiers defined in this standard, and has the following value:

ansi-X9-92 OBJECT IDENTIFIER::={iso(1) member-body(2) us(840) 1xxxx}

7.1 Syntax for Message Encoding Method

A message encoding and decoding method may be represented in a variety of ways using **ASN.1** algorithm object identifier to identify the cipher type. The object identifier **mem-stream**, **mem-tripleDES**, **mem-AES128, mem-AES192**, and **mem-AES256** are defined below:

memType **OBJECT IDENTIFIER::={ansi-X9-92 mem(1)}**
mem-stream **OBJECT IDENTIFIER::={memType 1}**
mem-SkipJack **OBJECT IDENTIFIER::={memType 2}**
mem-tripleDES **OBJECT IDENTIFIER::={memType 3}**
mem-AES128 **OBJECT IDENTIFIER::={memType 4}**
mem-AES192 **OBJECT IDENTIFIER::={memType 5}**
mem-AES256 **OBJECT IDENTIFIER::={memType 6}**

7.2 Syntax for Message Redundancy Criteria

The object identifiers **mrc-inherent, mrc-pad,** and **mrc-inherent-pad** are used to represent message redundancy criteria which are based on the inherent redundancy that shall be contained in a message, the padding string added redundancy as specified in Section 5.4, or the both:

mrcType **OBJECT IDENTIFIER::={ansi-X9-92 mrc(2)}**
mrc-inherent **OBJECT IDENTIFIER::={mrcType 1}**
mrc-pad **OBJECT IDENTIFIER::={mrcType 2}**
mrc-inherent-pad **OBJECT IDENTIFIER::={mrcType 3}**

7.3 Syntax for Digital Signatures

This section provides the syntax for the digital signatures defined in this Standard.

A signature may be represented in a variety of ways using **ASN.1** algorithm object identifier to identify the signature type and format. When PVS, KDF, TripleDES, and SHA-1 are used to sign an X.509 certificate and CRL, the signature shall be identified by the value **pvs-with-KDF-TDES-SHA1**, as defined below:

id-PVSigType **OBJECT IDENTIFIER::={ansi-X9-92 signatures(3)}**

pvs-with-KDF-TDES-SHA1 OBJECT IDENTIFIER::={id-PVSigType 1}

When **pvs-with-KDF-DES-SHA1** OID appears in the **algorithm** field of the **ASN.1** type **AlgorithmIdentifier**, and the **parameters** field is a value of type **NULL**, the PVS parameters for signature verification must be obtained from other sources, such as the **subjectPublicKeyInfo** field of the certificate of the issuer.

When a digital signature is identified by the OID **pvs-with-KDF-TDES-SHA1**, the digital signature shall be **ASN.1** encoded using the following syntax:

```
PVS-Sig-Value::=SEQUENCE{
        V               BIT STRING,
        r               BIT STRING,
        s               INTEGER
}
```

X.509 certificates and CRLs represent signatures as a bit string. Where a certificate or CRL is signed with PVS, KDF, DES, and SHA1, the entire encoding of a value **ASN.1** type **PVS-Sig-Value** shall be the value of the bit string.

Annex A
(Informative)
Bibliography

1. ANSI X9.30.2-1997: *Public Key Cryptography for the Financial Services – Part 2: The Secure Hash Algorithm (SHA-1).*

2. ANSI X9.52-1998: *Triple Data Encryption Algorithm, Modes of Operation*.

3. ANSI X9.62-1999: *Public Key Cryptography For The Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA).*

4. ANSI X9.63-1999: *Public Key Cryptography For The Financial Services Industry: Key Agreement and Key Transport Using Elliptic Curve Cryptography*.

5. ANSI X.9.91-2001: *Advanced Encryption Standard for the Financial Services Industry*.

6. IEEE 1363-2000, *Standard Specifications for Public Key Cryptography*.

7. IEEE P1363a, *Standard Specifications for Public Key Cryptography: Additional Techniques* (Draft).

8. ISO 15946-4, *Information Technology - Security Techniques – Cryptographic Techniques based on elliptic curves – Part 4: Digital Signatures Giving Message Recover*y (Draft).

9. L. Pintsov and S. Vanstone, Postal Revenue Collection in the Digital Age, in: *Financial Cryptography 2000*, Lecture Notes in Computer Science, Springer-Verlag.

Annex B
(Informative)
Mathematical Background

1. Elliptic curve related number-theoretic algorithms for PVS are specified in Annex A and Annex D of ANSI X9.62 [3].

2. Mathematical backgrounds for finite fields and elliptic curves over finite fields are specified in Annex B of ANSI X9.62 [3].

3. Tables of trinomials, pentanomials, and Gaussian normal bases are specified in Annex C of ANSI X9.62 [3].

4. Complex multiplication (CM) elliptic curve generation method is specified in Annex E of ANSI X9.62 [3].

5. An overview of elliptic curve systems is presented in Annex F of ANSI X9.62 [3].

6. Sample elliptic curves are presented in Annex J of ANSI X9.62 [3].

Annex C
(Informative)
Message Redundancy Criteria

This section provides user guidance for message redundancy criteria. The message redundancy comes from the inherent redundancy that shall be contained in a message, or the user added redundancy as specified in Section 5.4, or both. The nature of the inherent redundancy may be anything agreed upon and able to be checked by the communicating parties, for example, the inherent redundancy could come in any of the following formats:

1. The message is an ASCII file.

2. The message is in a specific format. For example, the message always begins with the company name and the message always ends with the name of the signatory representative.

3. The message contains certain key words.

4. Each paragraph ends with a period.

The amount of total redundancy is a scalable security parameter and shall be chosen to achieve security objectives. It is recommended that a minimum redundancy criteria which is consistent with 80 bits of symmetric key security be used. The redundancy in a padded message could be easily estimated. The user should also carefully estimate the redundancy contained in the inherent redundancy. For example, each ASCII character could contain one bit redundancy. The specific method for calculating the agreed redundancy is out of the scope of this Standard.

Annex D
(Informative)
Security Considerations

This annex is provided as initial guidance for implementers of this Standard. This information should be expected to change over the time. Implementers should be review the current state-of-the-art in attacks on elliptic curves systems, on hash functions, and on symmetric encryption schemes at the time of implementation.

D.1 Security Issues Related to Elliptic Curves

The following security considerations are discussed in the Annex H of ANSI X9.62 [3] and the Annex H of ANSI X9.63 [4]. The implementers are referred to there.

1. Attacks known on the elliptic curve discrete logarithm problem, which is the basis for the security of elliptic curves systems.

2. Security issues for elliptic curve domain parameters.

3. Security issues for elliptic curve key pairs.

D.2 Appropriate Key Lengths and Message Redundancy Criteria

In this section, we consider the appropriate key sizes and message redundancy criteria that should be used with PVS.

The goal of PVS signature scheme is to ensure that forging a valid signature is impossible without the knowledge of the corresponding private key.

This principle should be used to provide guidance on the size of the EC parameters selected and on the message redundancy criteria. For example, while the condition that $n>2^{160}$ is currently sufficient to provide security, it does not offer the same level of security as, say, a 80-bit symmetric scheme unless the amount of total redundancy in the message redundancy criteria is consistent with 80 bits of symmetric key security.

This section, therefore, provides guidance on the minimum size that n should be and on the minimum security that the message redundancy criteria should provide.

The guidance is made based on the following assumptions:

1. The best method to discover the value of a key for a symmetric key scheme is key exhaustion using a few known plaintext/ciphertext pairs. That is, for a 56-bit key, it takes 2^{56} trial encryptions to ensure that the correct key is recovered; for a 128-bit key, it takes 2^{128} trial encryptions, etc. This is a goal of any symmetric key scheme.

2. These symmetric key trial encryptions are able to be done in parallel. That is, *m* processors are used to attempt the trial encryptions, the time needed to exhaust the key space is reduced by a factor of *m*.

3. The best method for recovering an EC private key is to solve the particular ECDLP associated with the key. The best methods to solve the ECDLP are square root methods that take a number of operations which is approximately equal to the square root of the order *n* of the generator.

4. These EC operations for solving the ECDLP are able to be done in parallel.

5. For simplicity, an EC operation is assumed to take the same amount of time as the symmetric key encryption operation. In practice, this is a very conservative assumption: all known practical symmetric key algorithms are faster than known practical public algorithms.

The above assumptions lead to the guidance that the appropriate EC key size (that is, the size of *n*) is about the size of twice of the size of the symmetric key and that the amount of total redundancy in the message redundancy criteria is about the same size of the symmetric key. Table D-1 describes the bounds on *n* and the message redundancy that should be adopted to ensure that the PVS scheme conforms to the required security level.

Table D-1—Guidance on Aligning Elliptic Curve Size, Symmetric Key Size, and Message Redundancy Criteria

Symmetric Key Size (in bits)	Example Algorithm	Lower bound on *n*	Amount of Message Redundancy (in bits)
80	SKIPJACK	2^{160}	80
112	(Triple-DES)	2^{224}	112
128	AES	2^{256}	128
192	AES	2^{384}	192
256	AES	2^{512}	256

D.3 PVS

1. **Attacks on the hash function.** This standard specifies the use of an ANSI-approved hash function and the security of PVS is dependent on the security of the hash function chosen. If the hash function in use is broken, implementations of PVS should assess the effect of the attack on the security.

2. **Attacks on the symmetric encryption scheme.** This standard specifies the use of an ANSI-approved encryption scheme and the security of PVS is dependent on the security of the encryption scheme chosen. If the encryption scheme in use is broken, implementations of PVS should assess the effect of the attack on the security

3. **Repeated per-message secrets.** As with the possibility of repeated private keys, the possibility of a per-message secret k value repeating during signature generation may also be a concern. A k value repeats for two different messages, then it is possible for an adversary with access to both signatures to recover the associated private key. As with the private key, this event should never occur except by chance. As above, one way to address this concern is to use an ANSI X9 approved random or pseudorandom number generation method. Another way to address this possibility of an otherwise undetected hardware or software error or a poorly-seeded pseudorandom number generator is for a system intended for users with high security requirements to maintain a list of Z values previously output by signature generation so that it can detect if a k value ever repeats. If a repeated k value is detected, the associated signature should not be output and a possible error indicated. The owner of the system should try to determine what happened and what corrective action to take, including whether to continue to operate the system.

4. **The importance of the redundancy.** The number of redundancy bits (denoted by l in this section) is a scalable parameter of PVS, and is independent of the key length. For example, with key length of 160 bits and redundancy parameter $l = 40$, the existential forgery is possible with probability of about 2^{-40}. Users of PVS should determine the level of resistance desired against existential forgery, based on the importance of message being signed.

Annex E
(Informative)
Concrete Examples of Signature Overhead

Signature overhead means here the difference in length of the data resulting from signature generation and the length of the original message data.

E.1 23 bytes overhead

Assume that the recoverable message part *M* to be signed is 13 bytes which contains 7 bytes natural redundancy. To get 10 bytes of redundancy, 3 bytes of padding redundancy could be inserted. Moreover, if 20-byte of elliptic curve key, SHA-1, and TripleDES (or AES) are used, then the resulting *r* and *s* of the signature will be 16 bytes and 20 bytes respectively. The signature overhead is 36-13 = 23 bytes and total redundancy is 10 bytes which achieves 2^{-80} level security.

E.2 Signing extremely short message at 24 bytes overhead

Consider signing a short 1-byte recoverable message part *M* such as yes/no, buy/hold/sell, etc. To prevent replay attacks, such short messages often need to be sent together with a 3-byte sequence number. For the purpose of increasing forgery resistance, 4 bytes of padding redundancy could be inserted. With TripleDES, SHA-1, and 20-bytes elliptic curve key, the resulting *r* and *s* of the signature will be 8 bytes and 20 bytes respectively. The signature overhead is 28-4 = 24 bytes and total redundancy is 7 bytes which achieves 2^{-56} level security.

E.3 Signing and recoving longer messages at 20 bytes of overhead

If the message to be signed is 20 bytes or longer, it is reasonable to expect that certain formatting requirements or meaningfulness of the message will result in at least 10 bytes of natural redundancy. This obviates the need to insert additional redundancy. Therefore, the only overhead is the *s* part of the signature which is 20 bytes if 20-bytes elliptic curve key is used. In the worst case, when the message to be recovered has zero redundancy, 10 bytes of redundancy could be added which results in 30 bytes of redundancy.

Annex F
(Informative)
Numerical Examples

F.1 An example of PVS over the prime field F_p

Elliptic Curve Domain Parameter Setup:

1. The field F_p is generated by the prime:

   p = FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE FFFFAC73

2. The curve is E: $y^2=x^3+ax+b \pmod p$ over F_p where:

   a = 00000000 00000000 00000000 00000000 00000000
   b = 00000000 00000000 00000000 00000000 00000007

3. Base point G:

   $G = (G_x, G_y)$
   G_x = 3B4C382C E37AA192 A4019E76 3036F4F5 DD4D7EBB
   G_y = 938CF935 318FDCED 6BC28286 31733C3 F03C4FEE

G has prime order:

n = 01 00000000 00000000 0001B8FA 16DFAB9A CA16B6B3

h = 1.

Key Generation:

d = E6A080E0 B2A7A850 BA71D26C 9606669A 4B4A6C18

Public key is $Q = dG = (x_Q, y_Q)$:

x_Q = 8F5788A5 C97AC053 984045F4 C9FF325D D60065AE

y_Q = A5329D2A 721B5787 9C215323 37211F64 23E577DA

Message Redundancy Criteria:

The message input is a pair (M, V) where M is the recoverable message part. The redundancy of the message comes from the following two sources:

1. The inherent redundancy contained in the ASCII text M.

2. For each signature, 5 padding octets provide 40 bits redundancy.

Signature Generation:

The message input is a pair (M, V) = (13 0B546573 74205573 65722031, FA 2B0CBE77), where the recoverable message part M is the DER encoding of the *PrintableString* value "Test User 1".

1. Select a k in the interval $[1, n-1]$:

   k = D8A0ABC5 B7A4029A C232CBCD A16819E1 B715F9F4

2. Compute $kG = (x_1, y_1)$:

   x_1 = 1AF46EC4 E95DAEDE 056BFA3B 370075F6 3CB2C34F

   y_1 = C324F1BA 5324383C 371278D5 F3FE4FE2 9373702C

3. Convert x_1 to a bit string Z:

   Z = 1AF46EC4 E95DAEDE 056BFA3B 370075F6 3CB2C34F

4. Suppose that the method for combining the bit string Z with the recoverable message part is a stream cipher based on a key derivation function. Compute $r = MEM(M,Z)$.

   4.1. *padOctlen* = 5. P_1 = 05, P_2 = 0505050505

   4.2. $T = P_2||M$ = 0505 05050513 0B546573 74205573 65722031

   4.3. *keydatalen* = 144, *SharedInfo* is empty.

   $KeyData = KDF(Z)$ = 20B9 F76F3B33 6A805E02 92ED0B71 C9AAA767

   4.4. $r = T \oplus KeyData$ = 25BC F26A3E20 61D43B71 E6CD5E02 ACD88756

5. Compute the hash value:

   H = SHA1($r||V$) = 1B5361A3 9BC7CA45 6BBB6F2B F80E4E31 A01B4A1B

6. Compute e = 1560022182285506067002716486582364018485165614 35

7. Compute $s = k - de \bmod n$ = 735171855371469914412919293672210175623097579946

8. Output the signature:

   V = FA 2B0CBE77

   r = 25BC F26A3E20 61D43B71 E6CD5E02 ACD88756

   s = 735171855371469914412919293672210175623097579946 (digit)

Signature Verification:

V' = FA 2B0CBE77

r' = 25BC F26A3E20 61D43B71 E6CD5E02 ACD88756

s' = 735171855371469914412919293672210175623097579946 (digit)

1. Check that s' is an integer in the interval [1, n].

2. Compute the hash value:

   H' = SHA1($r'||V'$) = 1B5361A3 9BC7CA45 6BBB6F2B F80E4E31 A01B4A1B

3. Compute e' = 156002218228550606700271648658236401848516561435

4. Compute the elliptic curve point $(x'_1,y'_1)=s'G+e'Q$:

   x'_1 = 1AF46EC4 E95DAEDE 056BFA3B 370075F6 3CB2C34F

   y'_1 = C324F1BA 5324383C 371278D5 F3FE4FE2 9373702C

5. Convert x'_1 to a bit string Z':

   Z' = 1AF46EC4 E95DAEDE 056BFA3B 370075F6 3CB2C34F

6. The method for combining the bit string Z' with the recoverable message part is a stream cipher based on a key derivation function. Compute Compute $M' = MEM^{-1}(r', Z')$.

   6.1. *keydatalen* = *clen* = 144, *SharedInfo* is empty.

   KeyData = *KDF*(Z') = 20B9 F76F3B33 6A805E02 92ED0B71 C9AAA767

   6.2. $T'=r'\oplus KeyData$=0505 05050513 0B546573 74205573 65722031

   6.3. *tlen* = 144, which is greater than 8×*padOctlen* = 40.

   6.4. P_1 = 05, *padOctlen'* = 5.

   6.5. *padOctlen* = *padOctlen'*.

   6.6. Each octet of the next 4 octets after the first octet of T' is equal to P_1.

   6.7. M' = 13 0B546573 74205573 65722031

7. The value of M' is the DER encoding of the ASCII text "Test User 1" which satisfies the redundancy criteria (which contains 11 bits redundancy). In the previous decoding step, the 5

padding octets (40 bits) redundancy have been successfully found. Thus the signature is "valid". The message is:

$(M', V') = (13\ 0B546573\ 74205573\ 65722031, FA\ 2B0CBE77)$

F.2 An example of PVS over an extension field G$F(2^m)$

Elliptic Curve Domain Parameter Setup:

1. The field F_2^{163} is generated by the irreducible polynomial:

   $x^{163}+x^7+x^6+x^3+1$

2. The curve is E: $y^2+xy=x^3+ax^2+b$ over F_2^{163} where

   a = 00 00000000 00000000 00000000 00000000 00000001

   b = 00 00000000 00000000 00000000 00000000 00000001

3. Base point $G_y = G$:

   $G = (G_x, G_y)$
   G_x = 02 FE13C053 7BBC11AC AA07D793 DE4E6D5E 5C94EEE8
   G_y = 02 89070FB0 5D38FF58 321F2E80 0536D538 CCDAA3D9

G has prime order:

n = 04 00000000 00000000 00020108 A2E0CC0D 99F8A5EF

h = 2.

Key Generation:

d = 03 A41434AA 99C2EF40 C8495B2E D9739CB2 155A1E0D

Public key is $Q = dG = (x_Q, y_Q)$:

x_Q = 03 7D529FA3 7E42195F 10111127 FFB2BB38 644806BC

y_Q = 04 47026EEE 8B34157F 3EB51BE5 185D2BE0 249ED776

Message Redundancy Criteria:

The message input is a pair (M, V) where M is the recoverable message part. The redundancy of the message comes from the following two sources:

1. The inherent redundancy contained in the ASCII text M.

2. For each signature, 5 padding octets provide 40 bits redundancy.

Signature Generation:

The message input is a pair (M, V) = (13 0B546573 74205573 65722031, FA 2B0CBE77), where the recoverable message part M is the DER encoding of the *PrintableString* value "Test User 1".

1. Select a k in the interval [1, n−1]:

   k = A40B301C C315C257 D51D4422 34F5AFF8 189D2B6C

2. Compute $kG = (x_1, y_1)$:

   x_1 = 04 994D2C41 AA30E529 52B0A94E C6511328 C502DA9B

   y_1 = 03 1FC936D7 3163B858 BBC5326D 77C19839 46405264

3. Convert x_1 to a bit string Z:

   Z = 04 994D2C41 AA30E529 52B0A94E C6511328 C502DA9B

4. Suppose that the method for combining the bit string Z with the recoverable message part is a stream cipher based on a key derivation function. Compute $r = MEM(M,Z)$.

   4.1. *padOctlen* = 5. P_1 = 05, P_2 = 0505050505

   4.2. $T = P_2||M$ = 0505 05050513 0B546573 74205573 65722031

   4.3. *keydatalen* = 144, *SharedInfo* is empty.

   KeyData = *KDF*(Z) = D5BD 8BD7309D 020D5946 1367E723 A3B63D79

   4.4. $r = T \oplus KeyData$ = D0B8 8ED2358E 09593C35 6747B250 C6C41D48

5. Compute the hash value:

   H = SHA1($r||V$) = DB9B5EBE 6B7B9690 54F9AECB 52B54A19 DF4495AE

6. Compute e = 1253733847667589504694669707057885543542991132078

7. Compute $s = k - de \bmod n$ = 27482618165691942839912990389133089403687522756 58

8. Output the signature:

   V = FA 2B0CBE77

   r = D0B8 8ED2358E 09593C35 6747B250 C6C41D48

s = 27482618165691942839912990389133089403687522756 58 (digit)

Signature Verification:

V' = FA 2B0CBE77

r' = D0B8 8ED2358E 09593C35 6747B250 C6C41D48

s' = 27482618165691942839912990389133089403687522756 58 (digit)

1. Check that s' is an integer in the interval [1, n].
2. Compute the hash value:

   H' = SHA1($r'||V'$) = DB9B5EBE 6B7B9690 54F9AECB 52B54A19 DF4495AE

3. Compute e' = 125373384766758950469466970705788554354299113 2078
4. Compute the elliptic curve point $(x'_1,y'_1)=s'G+e'Q$:

   x'_1 = 04 994D2C41 AA30E529 52B0A94E C6511328 C502DA9B

   y'_1 = 03 1FC936D7 3163B858 BBC5326D 77C19839 46405264

5. Convert x'_1 to a bit string Z':

   Z' = 04 994D2C41 AA30E529 52B0A94E C6511328 C502DA9B

6. The method for combining the bit string Z' with the recoverable message part is a stream cipher based on a key derivation function. Compute Compute $M' = MEM^{-1}(r', Z')$.

   6.1. *keydatalen* = *clen* = 144, *SharedInfo* is empty.

   KeyData = *KDF*(Z') = 20B9 F76F3B33 6A805E02 92ED0B71 C9AAA767

   6.2. $T'=r'\oplus KeyData$=0505 05050513 0B546573 74205573 65722031

   6.3. *tlen* = 144, which is greater than 8×*padOctlen* = 40.

   6.4. P_1 = 05, *padOctlen'* = 5.

   6.5. *padOctlen* = *padOctlen'*.

   6.6. Each octet of the next 4 octets after the first octet of T' is equal to P_1.

   6.7. M' = 13 0B546573 74205573 65722031

7. The value of *M'* is the DER encoding of the ASCII text "Test User 1" which satisfies the redundancy criteria (which contains 11 bits redundancy). In the previous decoding step, the 5 padding octets (40 bits) redundancy have been successfully found. Thus the signature is "valid". The message is:

   (*M', V'*) = (13 0B546573 74205573 65722031, FA 2B0CBE77)