INTERNATIONAL STANDARD

ISO/IEC 14888-2

Second edition
2008-04-15

Information technology — Security techniques — Digital signatures with appendix

Part 2: Integer factorization based mechanisms

Technologies de l'information — Techniques de sécurité — Signatures numériques avec appendice

Partie 2: Mécanismes basés sur une factorisation entière

Reference number
ISO/IEC 14888-2:2008(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Contents

Page

Foreword

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

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 14888-2 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 27, *IT Security techniques*.

This second edition cancels and replaces the first edition (ISO/IEC 14888-2:1999), which has been technically revised.

ISO/IEC 14888 consists of the following parts, under the general title *Information technology — Security techniques — Digital signatures with appendix*:

— *Part 1: General*

— *Part 2: Integer factorization based mechanisms*

— *Part 3: Discrete logarithm based mechanisms*

Introduction

Digital signatures can be used to provide services such as entity authentication, data origin authentication, non-repudiation, and data integrity.

NOTE There are two series of International Standards specifying digital signatures. In both series, Part 2 specifies integer factorization based mechanisms and Part 3 specifies discrete logarithm based mechanisms.

- ISO/IEC 9796 [28] specifies signatures giving message recovery. As all or part of the message is recovered from the signature, the recoverable part of the message is not empty. The signed message consists of either the signature only (when the non-recoverable part of the message is empty), or both the signature and the non-recoverable part.
- ISO/IEC 14888 specifies signatures with appendix. As no part of the message is recovered from the signature, the recoverable part of the message is empty. The signed message consists of the signature and the whole message.

Most digital signature schemes involve three basic operations.

- An operation that produces key pairs. Each pair consists of a private signature key and a public verification key.
- An operation that makes use of a private signature key to produce signatures.
  - When, for a given message and private signature key, the probability of obtaining the same signature twice is negligible, the operation is probabilistic.
  - When, for a given message and private signature key, all the signatures are identical, the operation is deterministic.
- A deterministic operation that makes use of a public verification key to verify signed messages.

For each scheme, given the public verification key (but not the private signature key) and any set of signed messages (each message having been chosen by the attacker), the attacker should have a negligible probability of producing:

- a new signature for a previously signed message;
- a signature for a new message;
- the private signature key.

The title of ISO/IEC 14888-2 has changed, from *Identity-based mechanisms* (first edition) to *Integer factorization based mechanisms* (second edition).

a) The second edition includes the identity-based scheme specified in ISO/IEC 14888-2:1999, namely the GQ1 scheme. This scheme has been revised due to the withdrawal of ISO/IEC 9796:1991 in 1999.

b) Among the certificate-based schemes specified in ISO/IEC 14888-3:1998, it includes all the schemes based on the difficulty of factoring the modulus in use, namely, the RSA, RW and ESIGN schemes. These schemes have been revised due to the withdrawal of ISO/IEC 9796:1991 in 1999.

c) It takes into account ISO/IEC 14888-3:1998/Cor.1:2001, technical corrigendum to the ESIGN scheme.

d) It includes a format mechanism, namely the PSS mechanism, already specified in ISO/IEC 9796-2:2002, and details of how to use it in each of the RSA, RW, GQ1 and ESIGN schemes.

   NOTE Similar format mechanisms have proofs of security [2], even without a salt.

e) It includes new certificate-based schemes that use no format mechanism, namely, the GQ2, GPS1 and GPS2 schemes.

f) For each scheme and its options, as needed, it provides an object identifier.

ISO and IEC draw attention to the fact that it is claimed that compliance with this document may involve the use of patents.

ISO and IEC take no position concerning the evidence, validity and scope of these patent rights.

The holders of these patent rights have assured ISO and IEC that they are willing to negotiate licenses under reasonable and non-discriminatory terms and conditions with applicants throughout the world. In this respect, the statements of the holders of these patent rights are registered with ISO and IEC. Information may be obtained from the companies listed below:

Patent holder	Patent number(s)	Subject
NTT 20-2 Nishi-shinjuku 3-Chome Shinjuku-ku Tokyo 163-1419, Japan	US 4 625 076	ESIGN (see Clause 11)
France Telecom R&D [a] Service PIV 38-40 Rue du Général Leclerc F 92794 Issy les Moulineaux Cedex 9, France	US 5 140 634, EP 0 311 470 EP 0 666 664	GQ1 (see Clause 7) GPS1 (see Clause 9)
Philips International B.V. Corporate Patents and Trademarks P.O. Box 220 5600 AE Eindhoven, The Netherlands	US 5 140 634, EP 0 311 470	GQ1 (see Clause 7)
University of California Senior Licensing Officer Office of Technology Transfer 1111 Franklin Street, 5th floor Oakland, California 94607-5200, USA	US 6 266 771	PSS (see 6.4 when using salt and 11.4)

[a] France Telecom claims that patent applications are pending in relation to GQ2 (see Clause 8) and GPS2 (see Clause 10). The patent numbers will be provided when available. ISO/IEC will then request the appropriate statements.

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

Information technology — Security techniques — Digital signatures with appendix

Part 2:
Integer factorization based mechanisms

1 Scope

This part of ISO/IEC 14888 specifies digital signatures with appendix whose security is based on the difficulty of factoring the modulus in use. For each signature scheme, it specifies:

a) the relationships and constraints between all the data elements required for signing and verifying;

b) a signature mechanism, i.e., how to produce a signature of a message with the data elements required for signing;

c) a verification mechanism, i.e., how to verify a signature of a message with the data elements required for verifying.

The production of key pairs requires random bits and prime numbers. The production of signatures often requires random bits. Techniques for producing random bits and prime numbers are outside the scope of this part of ISO/IEC 14888. For further information, see ISO/IEC 18031 [33] and ISO/IEC 18032 [34].

Various means are available to obtain a reliable copy of the public verification key, e.g., a public key certificate. Techniques for managing keys and certificates are outside the scope of this part of ISO/IEC 14888. For further information, see ISO/IEC 9594-8 [27], ISO/IEC 11770 [31] and ISO/IEC 15945 [32].

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

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

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 14888-1 and the following apply.

3.1
modulus
integer whose factorization shall be kept secret and whose factors shall be infeasible to compute

3.2
representative
bit string produced by a format mechanism

3.3
salt
optional bit string for producing a representative

3.4
signature exponent
secret exponent for producing signatures

3.5
trailer
optional bit string on the right of a representative

3.6
verification exponent
public exponent for verifying signed messages and sometimes also for producing signatures

4 Symbols and abbreviated terms

For the purposes of this document, the following symbols and abbreviated terms apply.

$A \parallel B$	bit string resulting from concatenating the two bit strings A and B in that order
$A \oplus B$	bit string resulting from exclusive-oring the two bit strings A and B, of the same length
b	adaptation parameter (GQ2)
Cr	CRT coefficient
CRT	Chinese Remainder Theorem
$\|D\|$	bit length of D if D is a bit string, or bit size of D if D is a number (i.e., 0 if $D = 0$, or the unique integer i so that $2^{i-1} \leq D < 2^i$ if $D > 0$, e.g., $\|65\,537 = 2^{16}+1\| = 17$)
$\lfloor D \rfloor$	the greatest integer less than or equal to D
$\lceil D \rceil$	the least integer greater than or equal to D
E	salt (RSA, RW, ESIGN)
F	representative (RSA, RW, GQ1, ESIGN)
f	number of prime factors
G, G_i	public number
g, g_i	base number
$(g \mid n)$	Jacobi symbol of a positive integer g with respect to an odd composite integer n

NOTE 1 By definition, the Jacobi symbol of g with respect to n is the product of the Legendre symbols of g with respect to each prime factor of n (repeating the Legendre symbols for repeated prime factors). The Jacobi symbol [13, 15] can be efficiently computed without knowledge of the prime factors of n.

$(g\|p)$	Legendre symbol of a positive integer g with respect to an odd prime integer p

NOTE 2 By definition, if p is prime, then $(g|p) = g^{(p-1)/2} \bmod p$. This means that $(g|p)$ is zero if g is a multiple of p, and either $+1$ or -1 otherwise, depending on whether or not g is a square modulo p.

$\gcd(a, b)$	the greatest common divisor of the two positive integers a and b
H, HH	hash-codes
h	hash-function
$i \bmod n$	the unique integer j from 0 to $n-1$ such that n divides $i-j$
Id	sequence of identification data (GQ1)
$Indic$	indicator of a mechanism in use (hash-function, format mechanism, hash-variant)
k	security parameter (GQ2)
$\mathrm{lcm}(a, b)$	the least common multiple of the two positive integers a and b
M	message
m	number of base numbers (GQ2)
n	modulus
p_i	prime factor
Q, Q_i	private number
$Q_{i,j}$	private component (GQ2)
R	first part of signature (GQ1, GQ2, GPS1, GPS2)
r, r_i, $r_{i,j}$	random number (GQ1, GQ2, GPS1, GPS2, ESIGN)
S	signature (RSA, RW, ESIGN) or second part of signature (GQ1, GQ2, GPS1, GPS2)
s, s_i	signature exponent (RSA, RW, GQ1, GQ2)
T	coupon (GPS1, GPS2)
t	signature length parameter (GQ1, GQ2)
u, u_i	exponent (GQ1, GQ2)
v	verification exponent (RSA, RW, GQ1, GPS2, ESIGN)
W	bit string (GQ1, GQ2, GPS1, GPS2)
'XY'	notation using the hexadecimal digits '0' to '9' and 'A' to 'F', equal to XY to the base 16
x, y, z	integers
α	bit size of the moduli
γ	bit length of the representatives (RSA, RW, GQ1, ESIGN)
ε	bit length of the salts (format mechanisms)
τ	bit length of the trailers (format mechanisms)

5 General

5.1 Security requirements

The signature mechanism makes use of a set of data elements required for signing. This set includes the signer's private signature key, which is referred to simply as the “signature key” in this document. Some data elements of the signature key shall be kept secret (there is at least one secret data element).

NOTE Every secret data element should remain confined within a piece of hardware or software under the control of the signer, in such a way that it is infeasible for an attacker to extract it. Integrated circuit cards [24] may produce signatures. Protection profiles for signature production devices are outside the scope of this document.

The production of RSA and RW signatures is probabilistic when and only when every signature requires a fresh salt. The production of GQ1, GQ2, GPS1, GPS2 and ESIGN signatures is essentially probabilistic. When the production of signatures is probabilistic, every signer shall have the means to select random bits.

The verification mechanism makes use of a set of data elements required for verifying, all of which shall be made public within the domain.

- Every public data element common to all signers is known as a domain parameter.
- Every public data element specific to a single signer shall be part of the signer's public verification key, which is referred to simply as the “verification key” in this document.

Within a given domain, every verifier shall know the set of domain parameters and shall obtain a reliable copy of the signer's verification key.

The signer and the verifier shall have adequate assurance that the set of domain parameters is valid, i.e., that it satisfies the constraints specific to the scheme. Otherwise, there is no assurance of meeting the intended security even if the signed message is accepted. This assurance may be obtained in various ways, including one or more of:

a) selection of a set of values from a trusted published source, e.g., an International Standard;

b) production of a set of values by a trusted third party, e.g., a certification authority [27];

c) validation of a set of values by a trusted third party, e.g., a certification authority [27];

d) for the signer, production of a set of values by a trusted system;

e) for the signer and the verifier, validation of a set of values.

The signer and the verifier shall have adequate assurance that the verification key is valid, i.e., that it satisfies the constraints specific to the scheme. This assurance may be obtained in various ways, including one or more of:

a) access to a directory or verification of a certificate;

b) a key validation protocol operating on the verification key and possibly other information, perhaps involving an interaction with the piece of hardware or software producing signatures;

c) trust in another party's assertion of having obtained assurance that the verification key is valid;

d) trust that the key production has been implemented correctly.

Specific key validation protocols and methods for obtaining and conveying assurance of key validity are outside the scope of this document.

The security of every signature scheme specified in this document relies upon a modulus and a hash-function.

- A modulus is secure (i.e., factorization-resistant) as long as no factor has been revealed. In the context of use of the scheme, no entity shall be able to effectively factor the modulus in use.
- The hash-function in use shall be one of those specified in ISO/IEC 10118; it should be collision-resistant.

5.2 Verification keys

Table 1 summarizes the verification keys (see 6.1, 7.1, 8.1, 9.1, 10.1 and 11.1).

Table 1 — Verification keys

Scheme	Mandatory	Optional [a]		Optional [b]	
RSA, RW, ESIGN	n	v	$Indic$(h)	α	$Indic$(format, ε, τ)
GQ1 [c]		n, v	$Indic$(h)	α	$Indic$(variant), $Indic$(format, ε, τ)
GQ2	n		$Indic$(h)	b, $(g_1, g_2 \ldots g_m)$, α	$Indic$(variant)
GPS1	G	n	$Indic$(h)	g, α	$Indic$(variant)
GPS2	n	v	$Indic$(h)	g, α	$Indic$(variant)

[a] If not part of the verification key, such a data element shall be a domain parameter.
[b] If neither a domain parameter, nor part of the verification key, such a data element shall take a default value.
[c] The GQ1 verification key may be empty.

Every signature scheme specified in this document makes use of a modulus, denoted n.

- In the RSA, RW, GQ2, GPS2 and ESIGN schemes, the verification key shall include n.
- In the GQ1 and GPS1 schemes, either the domain parameters or the verification key shall include n.

NOTE The use of a given modulus is normally limited to a given period of time within a given domain.

To prescribe the bit size of the modulus in use, either the domain parameters or the verification key may include a data element, denoted α. If α is not included, then the default value of α is set equal to the bit size of the modulus in use (i.e., the modulus size is not prescribed).

In the GPS1 scheme, the verification key shall include the public number in use, denoted G.

For compatibility with public key infrastructures already deployed, even when all the signers use the same value within the domain, the verification key may include:

— the verification exponent in use, denoted v, in the RSA, RW, GQ1, GPS2 and ESIGN schemes;

— the modulus in use, denoted n, in the GQ1 and GPS1 schemes.

Every signature scheme specified in this document makes use of a hash-function, denoted h.

- In the RSA, RW and ESIGN schemes, a format mechanism makes use of h to convert messages into representatives, and to check recovered representatives.
- In the GQ1 scheme, a format mechanism makes use of h to convert sequences of identification data into public numbers, and a hash-variant makes use of h to produce bit strings.
- In the GQ2, GPS1 and GPS2 schemes, a hash-variant makes use of h to produce bit strings.

To indicate the hash-function in use, either the domain parameters or the verification key shall include a data element, denoted $Indic$(h).

This document specifies three format mechanisms (PSS in 6.4, 7.4 and 11.4; D1 and D2 in Annex D). Each format mechanism makes use of two parameters, denoted ε and τ. Set to 0, 64 or $|H|$, ε indicates the bit length of the salts. Set to 0, 8 or 16, τ indicates the bit length of the trailers.

This document specifies four hash-variants, where W denotes a bit string and M a message.

1) h(W || M) 2) h(W || h(M)) 3) h(h(W) || M) 4) h(h(W) || h(M))

To indicate the format mechanism in use, together with the options ε and τ in use, and/or the hash-variant in use, either the domain parameters or the verification key may include one or two data elements, denoted $Indic$(format, ε, τ) and $Indic$(variant), as needed.

Key precedence — When the domain parameters and the verification key include the same data element with different values, the verification key shall take precedence.

NOTE Within a given domain, owing to key precedence, different signers may make use of different hash-functions and/or different modulus sizes.

5.3 CRT technique

Consider two integers x_1 and x_2 that are co-prime, but not necessarily prime. By definition, the CRT coefficient of x_1 and x_2, denoted Cr, is the unique positive integer, less than x_1, such that $Cr \times x_2 - 1$ is a multiple of x_1.

Any integer X from $\{0, 1 \ldots x_1 \times x_2 - 1\}$ may be decomposed into the unique pair of components $X_1 = X \bmod x_1$ from $\{0, 1 \ldots x_1 - 1\}$ and $X_2 = X \bmod x_2$ from $\{0, 1 \ldots x_2 - 1\}$.

The CRT composition reverses the above decomposition. It makes use of the three integers x_1, x_2 and Cr to convert any two components X_1 from $\{0, 1 \ldots x_1 - 1\}$ and X_2 from $\{0, 1 \ldots x_2 - 1\}$, into the unique integer X from $\{0, 1 \ldots x_1 \times x_2 - 1\}$ such that $X_1 = X \bmod x_1$ and $X_2 = X \bmod x_2$.

$$Y = X_1 - X_2 \mod x_1;\ Z = Y \times Cr \mod x_1;\ X = Z \times x_2 + X_2$$

In order to convert three components X_1 from $\{0, 1 \ldots x_1 - 1\}$, X_2 from $\{0, 1 \ldots x_2 - 1\}$ and X_3 from $\{0, 1 \ldots x_3 - 1\}$, where x_1, x_2 and x_3 are pairwise co-prime, into the unique integer X from $\{0, 1 \ldots x_1 \times x_2 \times x_3 - 1\}$ so that $X_1 = X \bmod x_1$, $X_2 = X \bmod x_2$ and $X_3 = X \bmod x_3$, the CRT composition is used twice:

1) to compute T from $\{0, 1 \ldots x_1 \times x_2 - 1\}$ so that $X_1 = T \bmod x_1$ and $X_2 = T \bmod x_2$;

2) to compute X from $\{0, 1 \ldots x_1 \times x_2 \times x_3 - 1\}$ so that $T = X \bmod x_1 \times x_2$ and $X_3 = X \bmod x_3$.

When the prime factors of n are available (see 6.2, 7.1, 8.1, 8.2, 9.1, 9.2.2 and 10.2.2), the CRT technique reduces the complexity of arithmetic computations mod n (see B.2.3). Rather than directly computing a global result from $\{0, 1 \ldots n-1\}$, a set of components is computed and then converted into the global result.

NOTE The CRT technique efficiency increases in terms of the number of distinct prime factors.

5.4 Conversions between bit strings, integers and octet strings

A bit string, denoted D, consists of $|D|$ bits, where the value of each bit is 0 or 1; the bits are numbered from the leftmost bit, denoted d_1, to the rightmost bit, denoted $d_{|D|}$.

$$D = d_1\, d_2\, d_3 \ldots d_{|D|-1}\, d_{|D|}$$

To convert D into an integer, denoted A, the leftmost bit, denoted d_1, is the most significant bit, and the rightmost bit, denoted $d_{|D|}$, is the least significant bit.

$$A = 2^{|D|-1} \times d_1 + 2^{|D|-2} \times d_2 \ldots + 2^2 \times d_{|D|-2} + 2 \times d_{|D|-1} + d_{|D|}$$

The bit size of integer A, denoted $|A|$ (i.e., $2^{|A|-1} \le A < 2^{|A|}$ if $A > 0$, noting that $0 \le A < 2^{|D|}$), is either equal to $|D|$ if $d_1 = 1$, or less than $|D|$ if $d_1 = 0$. The binary representation of integer A by a bit string of length greater than $|A|$ is the unique bit string which, when converted to an integer, gives A.

When the bit length of a string is a multiple of eight, the bit string is conveniently represented by an octet string where each octet has a value from '00' to 'FF' in the hexadecimal notation. In an octet string, the octets are numbered from the leftmost octet to the rightmost octet. To convert an octet string into an integer, the leftmost octet is the most significant octet and the rightmost octet is the least significant octet.

6 RSA and RW schemes [1]

6.1 Data elements required for signing/verifying

The subsequent relationships and constraints apply to the following data elements:

- a verification exponent;
- a set of distinct prime factors;
- a modulus;
- a signature exponent;
- a set of CRT signature exponents.

The verification exponent is denoted v. The values $v = 0$ and $v = 1$ are forbidden.

NOTE The values v = 2, 3 and 65 537 (= 2^{16}+1) have some practical advantages.

The set of distinct prime factors is denoted $p_1, p_2 \dots p_f$ in ascending order ($f > 1$).

The RSA scheme makes use of an odd verification exponent. There may be more than two prime factors ($f \geq 2$). For i from 1 to f, v shall be co-prime to $p_i - 1$, i.e., $\gcd(v, p_i - 1) = 1$.

The RW scheme makes use of an even verification exponent. This document mandates the value $v = 2$, with only two prime factors ($f = 2$), both congruent to 3 mod 4, but not congruent to each other mod 8.

The modulus, denoted n, is the product of the prime factors ($n = p_1 \times ... \times p_f$). Its size shall be α bits.

The signature exponent is denoted s. It is any positive integer (the least one is often used) so that $v \times s - 1$ is a multiple of either $\mathrm{lcm}(p_1 - 1, \dots p_f - 1)$ if v is odd, or $\mathrm{lcm}(p_1 - 1, p_2 - 1)/2$ if $v = 2$.

The set of CRT signature exponents is denoted s_1 to s_f. For i from 1 to f, s_i is any positive integer (the least one is often used) so that $v \times s_i - 1$ is a multiple of either $p_i - 1$ if v is odd, or $(p_i - 1)/2$ if $v = 2$.

NOTE In the RW scheme, as a prime factor is congruent to 3 mod 8 and the other one to 7 mod 8, $n \equiv 5 \bmod 8$, $(\pm 2 \mid n) = -1$, $s = (n - p_1 - p_2 + 5)/8$, $s_1 = (p_1 + 1)/4$ and $s_2 = (p_2 + 1)/4$.

Signing requires a hash-function (see 5.1), a format mechanism and a signature key. The format mechanism specified in 6.4 is recommended; it makes use of two parameters, denoted ε and τ. The signature key takes either of two forms:

- With CRT: p_1 to p_f, f–1 CRT coefficients (see 5.3) and s_1 to s_f.
- Without CRT: n and s (n public).

NOTE The format mechanism specified in 6.4 is believed to be secure. The two format mechanisms specified in Annex D have a smaller safety margin.

Verifying requires a set of domain parameters and a verification key. Either the domain parameters or the verification key shall include v and $Indic$(h), and may include α (by default, $\alpha = |n|$) and $Indic$(format, ε, τ) (by default, 6.4 with the options $\varepsilon = |H|$ and $\tau = 8$). The verification key shall include n.

---

[1] The RSA scheme is due to Rivest, Shamir and Adleman [4, 19]. It makes use of a permutation of the ring of the integers modulo n.

The RW scheme is due to Rabin [18] and Williams [23]. It makes use of a permutation of a subset of the ring of the integers modulo n, namely, the set of the elements less than $n/2$ and having +1 as Jacobi symbol with respect to n.

6.2 Signature mechanism

Illustrated in Figure 1, the mechanism makes use of a hash-function, a format mechanism and a signature key, to sign a message (a bit string, denoted M), i.e., to produce a signature of M (a bit string, denoted S).

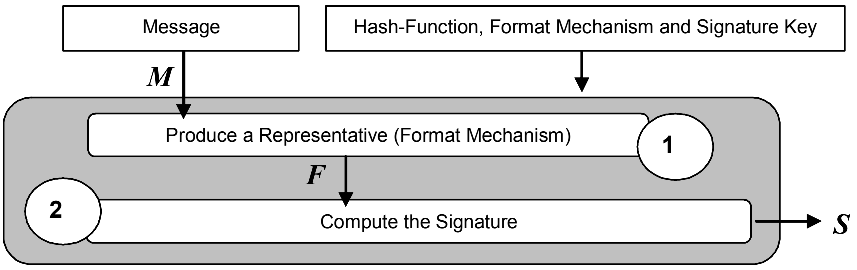

Figure 1 — Signing with RSA or RW

Stage 1 — Convert the message M into a representative of $\gamma = |n|$ bits, denoted F, in accordance with the format mechanism in use. The bit string F represents a number, divisible by four, also denoted F $(0 < F < n)$.

Stage 2 — Produce a number, denoted G $(0 < G < n)$.

- If v is odd, then $G = F$.
- If $v = 2$, evaluate the Jacobi symbol $(F \mid n)$ and force the Jacobi symbol $(G \mid n)$ to $+1$.
  - — If $(F \mid n) = +1$, then $G = F$.
  - — If $(F \mid n) = -1$, then $G = F / 2$.
  - — If $(F \mid n) = 0$ (a very unlikely case), then the procedure fails.

Produce a number, denoted S, in either of two ways.

- With CRT, for i from 1 to f, compute $G_i = G \bmod p_i$ and $S_i = G_i^{s_i} \bmod p_i$. The number S is the CRT composition (see 5.3) of S_1 to S_f.
- Without CRT, compute $S = G^s \bmod n$.

If $v = 2$, then the number S may be replaced by $n - S$.

The signature is any bit string representing S, often a string of $|n|$ bits, and is also denoted S.

6.3 Verification mechanism

Illustrated in Figure 2, the mechanism makes use of a set of domain parameters and a verification key (see Table 1), with key precedence (see 5.2), to verify a message and a signature of that message, i.e., the two bit strings, denoted M and S.

Stage 0 — Reject if $|n| \neq \alpha$, or if $v = 0$ or 1, or if n is not congruent to $5 \bmod 8$ when $v = 2$.

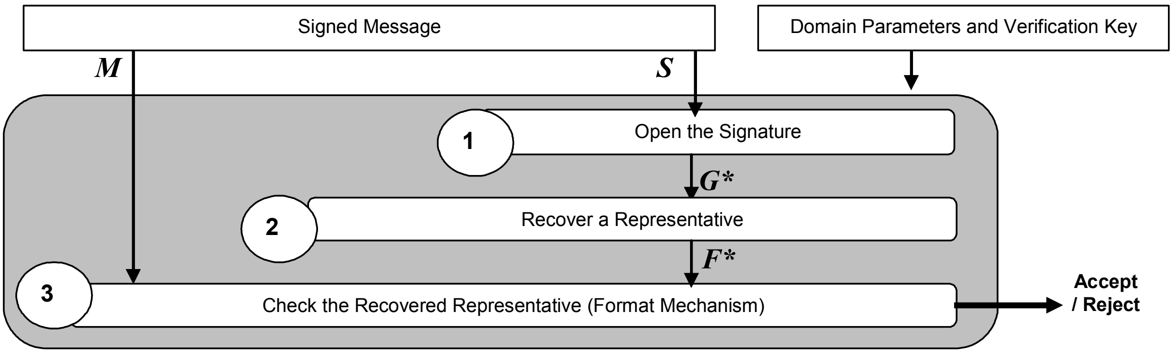

Figure 2 — Verifying with RSA or RW

Stage 1 — The bit string S represents a number, also denoted S. Reject if $S = 0$ or 1, or if $S \geq n-1$.

Compute $G^* = S^v \bmod n$.

Stage 2 — Recover a representative, denoted F^*.

- If v is odd, F^* is the string of $|n|$ bits representing G^*.
- If $v = 2$, F^* is the string of $|n|$ bits representing:
  - — G^* if G^* is congruent to $4 \bmod 8$;
  - — $n - G^*$ if G^* is congruent to $1 \bmod 8$;
  - — $2\,G^*$ if G^* is congruent to $6 \bmod 8$;
  - — $2\,(n - G^*)$ if G^* is congruent to $7 \bmod 8$.
  - — Reject in any other case (the trailer cannot be interpreted).

Stage 3 — Check the recovered representative F^* in accordance with the format mechanism in use.

6.4 Format mechanism [2]

Convert the message M, making use of two parameters (ε indicates the length of the salt and τ indicates the length of the trailer), into a representative of γ bits, denoted F. Figure 3 illustrates the mechanism.

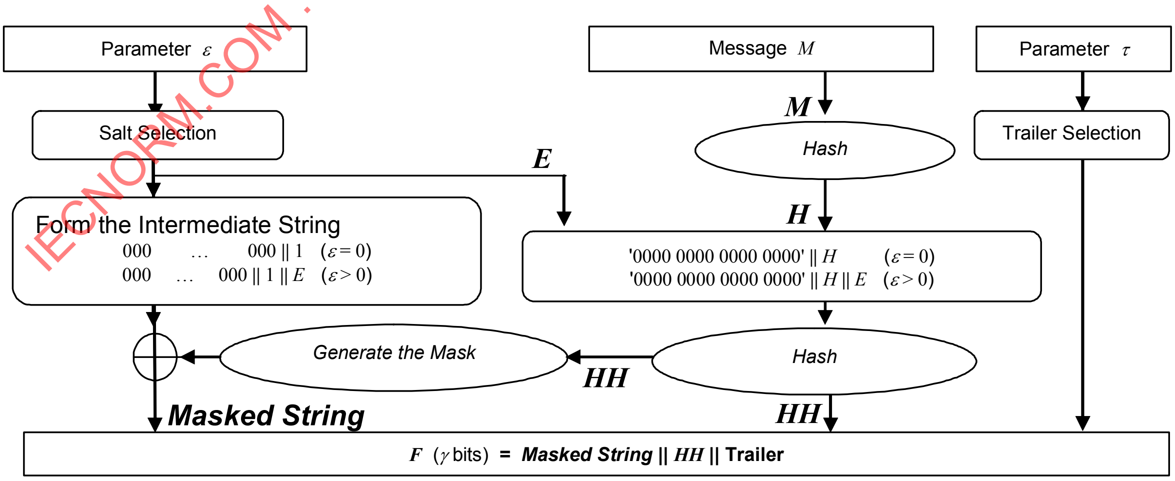

Figure 3 — Production of a representative

[2] This mechanism is due to Bellare and Rogaway [1]. When the salt has a fresh value for each signature, the resulting signature scheme is known as either RSA-PSS, or RW-PSS, where PSS stands for "Probabilistic Signature Scheme".

1) The options are as follows.

- Option $\varepsilon = 0$. The salt is the empty string and the production of signatures is deterministic.
- Option $\varepsilon = |H|$. The salt, denoted E, is a string of $|H|$ random bits.
  - If the salt has a fixed value for many signatures, then the production of signatures is deterministic.
  - If the salt has a fresh value for each signature, then the production of signatures is probabilistic.
- Option $\tau = 8$. The trailer is a single octet, set to 'BC'.
- Option $\tau = 16$. The trailer is two consecutive octets: the rightmost one is set to 'CC'; the leftmost one identifies the hash-function in use. The leftmost octet shall be interpreted as follows.
  - The range '00' to '7F' is reserved for ISO/IEC JTC 1 SC 27; ISO/IEC 10118 specifies a unique identifier in that range for every standardized hash-function, e.g., '31' refers to the first function in Part 3, namely RIPEMD-160, and '33' refers to the third function in Part 3, namely SHA-1.
  - The range '80' to 'FF' is reserved for proprietary use.

Trailer = *Hash-function identifier* || 'CC'

NOTE Some research [12] questions the benefits of using such an identifier in the trailer.

2) Hash M into a bit string, denoted H. From left to right, concatenate 8 octets, all set to '00', H and E. Hash the concatenation into a bit string, denoted HH.

$$H = \mathrm{h}(M) \qquad HH = \mathrm{h}(\text{'0000 0000 0000 0000'}) \,\|\, H \,\|\, E)$$

3) Produce a string of at least $\gamma - \tau - |H|$ bits from HH by the following procedure making use of two variables: a string of variable length, denoted *String*, and a string of 32 bits, denoted *Counter*.

   a) Set *String* to the empty string.

   b) Set *Counter* to zero.

   c) Replace *String* by *String* || $\mathrm{h}(HH \,\|$ *Counter*).

   d) Replace *Counter* by *Counter* $+ 1$.

   e) If $|H| \times$ *Counter* < $\gamma - \tau - |H|$, then go to stage c.

   Form a mask with the leftmost $\gamma - \tau - |H|$ bits of *String* where the leftmost bit has been forced to 0.

4) Form an intermediate string of $\gamma - \tau - |H|$ bits from left to right by concatenating:
   - $\gamma - \tau - |H| - 1 - \varepsilon$ bits, all zeroes;
   - a border bit, set to 1;
   - the salt E.

5) By exclusive-oring, apply the mask to the intermediate string, thereby producing a masked string.

6) Form F from left to right, by concatenating the masked string, HH and the trailer. Return F.

F = Masked string || HH || Trailer

Check a recovered representative of γ bits, denoted F^*, with respect to the message M and the two options ε and τ in use (indicated either by the verification key, or by the domain parameters, or by default).

1) Check the trailer as follows.
   - If the rightmost octet of F^* is set to 'BC', then the recovered option is $\tau^* = 8$.
   - If the rightmost octet of F^* is set to 'CC' and if the octet on the left of 'CC' identifies the hash-function in use, then the recovered option is $\tau^* = 16$.
   - Reject in any other case (the trailer cannot be interpreted) and also if τ^* and τ are different.

2) Split the leftmost $\gamma - \tau$ bits of F^* in two pieces: a masked string of $\gamma - \tau - |H|$ bits on the left and a string of $|H|$ bits, denoted HH^*, on the right.

3) Produce a mask of $|n| - \tau - |H|$ bits from HH^* as step 3 above.

4) By exclusive-oring, apply the mask to the masked string, thereby producing a recovered intermediate string where, starting from the left, the border bit is the first bit that is set to 1.

— If ε bits remain on the right of the border bit in the recovered intermediate string, then they form a bit string, denoted E^*.

— Otherwise, reject.

5) Hash M into a bit string, denoted H. From left to right, concatenate 8 octets, all set to '00', H and E^*. Hash the concatenation into a bit string, denoted HH.

$$H = \mathrm{h}(M) \qquad HH = \mathrm{h}(\text{'0000 0000 0000 0000'} \,||\, H \,||\, E^*)$$

6) Accept or reject depending on whether HH and HH^* are identical or different.

7 GQ1 scheme[3] (identity-based scheme)

7.1 Set of data elements required for signing/verifying

NOTE The set of prime factors is the secret of the entity that makes the modulus public; the modulus is either a domain parameter, or part of the verification key. Consequently, the scheme may be implemented in either of two ways.

1) If the modulus is a domain parameter, then the entity that makes the modulus public is a trusted authority that equips each signer with a private number, thus guaranteeing the sequence of identification data of the signer. For example, an issuer of integrated circuit cards [24] has a modulus.

- To personalize cards, a delegated entity makes use of the issuer's secret to sign sequences of identification data; in each card, it stores a sequence of identification data and a private number.
- During its life, the card uses its private number in accordance with zero-knowledge techniques.

2) If the modulus is part of a verification key, then for each session, the signer is equipped with a private number, thus guaranteeing the sequence of session identification data. In a local area network, a trusted authority supervises each login operation and manages a directory where any verifier can obtain a trusted copy of the modulus of every entity.

- When a computer connects the local area network, i.e., during a login operation, it makes use of the secret of the appropriate entity to produce a private number by a single-sign-on of a sequence of session identification data.
- During the session, the computer cannot use the secret of the entity (a long-term secret) because it does not know it any more; it uses the private number in accordance with zero-knowledge techniques. The private number (a short-term secret) only lasts for a few hours: its utility disappears after the session.

The subsequent relationships and constraints apply to the following data elements:

- a verification exponent and a signature length parameter:
- a set of distinct prime factors;
- a modulus;
- a sequence of identification data;
- a public number;
- a private number.

[3] The GQ1 scheme is due to Guillou and Quisquater [9, 10]. It makes use of zero-knowledge techniques for proving, without revealing, the knowledge of the RSA signature of a sequence of identification data (see also ISO/IEC 9798-5 [30]).

The verification exponent, denoted v, shall be a prime number. The signature length parameter is denoted t. The product $(|v|-1) \times t$ shall be less than or equal to $|H|$.

NOTE For $(|v|-1) \times t = 80$, typical values of v and t are $(2^{80}+13, 1)$, $(2^{40}+15, 2)$, $(2^{20}+7, 4)$, $(2^{16}+1, 5)$.

The set of distinct prime factors is denoted $p_1, p_2 \ldots p_f$ in ascending order ($f > 1$).

For i from 1 to f, v shall not divide $p_i - 1$.

The modulus, denoted n, is the product of the prime factors ($n = p_1 \times \ldots \times p_f$). Its size shall be α bits.

The creation of every signer requires the following three stages.

Stage 1 — Select a sequence of identification data, denoted Id. It is a bit string, uniquely and meaningfully identifying the signer in accordance with a convention agreed at domain level.

NOTE The sequence of identification data includes e.g., an identifier such as an account number, a serial number, an expiry date and time, rights. The presence of an expiry date and time in the sequence enforces its expiry; the presence of a serial number simplifies its revocation.

Stage 2 — Convert Id into a representative of $\gamma = |n|$ bits in accordance with the format mechanism in use. It represents the public number, denoted G ($1 < G < n$).

Stage 3 — Produce the private number, denoted Q, in either of two ways.

- With CRT, for i from 1 to f, compute a number, denoted s_i, as the least positive integer so that $v \times s_i - 1$ is a multiple of $p_i - 1$, then $u_i = p_i - 1 - s_i$, $G_i = G \bmod p_i$ and $Q_i = G_i^{u_i} \bmod p_i$. The number Q is the CRT composition (see 5.3) of Q_1 to Q_f.
- Without CRT, compute a number, denoted s, as the least positive integer so that $v \times s - 1$ is a multiple of $\mathrm{lcm}(p_1-1, \ldots p_f-1)$, then $u = \mathrm{lcm}(p_1-1, \ldots p_f-1) - s$ and $Q = G^u \bmod n$.

NOTE The number Q is the inverse $\bmod\ n$ of the signature defined in 6.1. The pair G and Q verifies $G \times Q^v \bmod n = 1$.

Signing requires a hash-function (see 5.1), a hash-variant, a format mechanism and a signature key. The format mechanism specified in 7.4 is recommended. The signature key consists of t, v, n and Q (t, v, n public).

Verifying requires a set of domain parameters, a verification key and Id. The domain parameters may include t (by default, $t = 1$). Either the domain parameters or the verification key shall include v, n and $Indic$(h), and may include α (by default, $\alpha = |n|$), $Indic$(variant) (by default, the first variant) and $Indic$(format, ε, τ) (by default, 7.4).

7.2 Signature mechanism

Illustrated in Figure 4, the mechanism makes use of a hash-function, a hash-variant and a signature key, to sign a message (a bit string, denoted M), i.e., to produce a signature of M (two bit strings, denoted R and S).

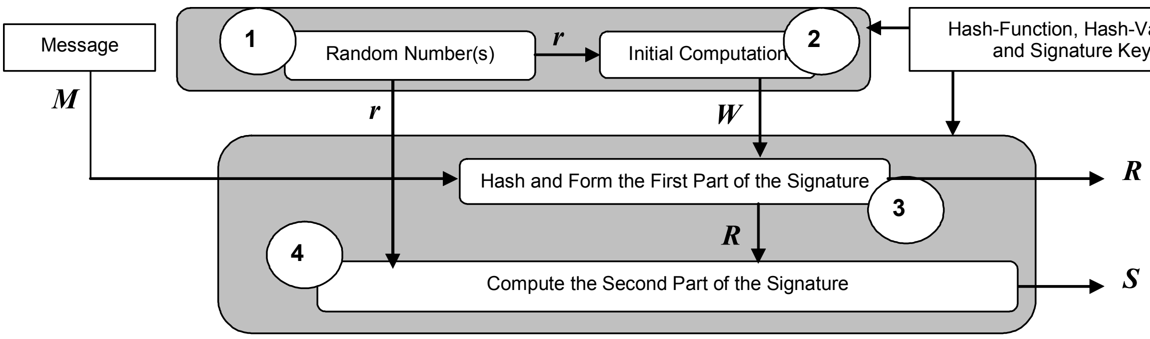

Figure 4 — Signing with GQ1

Stage 1 — Select t strings of $|n|$ random bits.

They represent random numbers to be kept secret, denoted r_1 to r_t (globally denoted r in Figure 1).

NOTE The probability that a string of $|n|$ random bits represents zero or a multiple of any prime factor of n is so negligible that such a possibility need not be formally checked.

Stage 2 — For i from 1 to t, compute $r_i^{\,v} \bmod n$ and represent it by a string of $|n|$ bits, denoted W_i.

Form a string of $|n| \times t$ bits, denoted W, with $W_1 \,||\, W_2 \,||\, \ldots \,||\, W_t$.

Stage 3 — Produce a bit string, denoted H, in accordance with the hash-variant in use.

$H =$ $h(W \,||\, M)$ in the first variant
$h(W \,||\, h(M))$ in the second variant
$h(h(W) \,||\, M)$ in the third variant
$h(h(W) \,||\, h(M))$ in the fourth variant

Form the first part of the signature, denoted R, with the leftmost $(|v| - 1) \times t$ bits of H.

Stage 4 — Split R into t strings of $|v|-1$ bits, namely $R_1 \,||\, R_2 \,||\, \ldots \,||\, R_t$. Each bit string R_i represents a number, also denoted R_i (less than $2^{|v|-1}$, and therefore, less than v).

For i from 1 to t, compute $r_i \times Q^{R_i} \bmod n$ and represent it by a string of $|n|$ bits, denoted S_i.

Form the second part of the signature, denoted S, with $S_1 \,||\, S_2 \,||\, \ldots \,||\, S_t$ ($|n| \times t$ bits).

7.3 Verification mechanism

Illustrated in Figure 5, the mechanism makes use of a set of domain parameters, a verification key (see Table 1), with key precedence (see 5.2), and a sequence of identification data (a bit string, denoted Id), to verify a message and a signature of that message, i.e., the three bit strings, denoted M, R and S.

Stage 0 — Reject if $|n| \neq \alpha$, or if v is not odd and prime, or if $|R| \neq (|v| - 1) \times t$, or if $|S| \neq |n| \times t$, or if Id is expired or revoked.

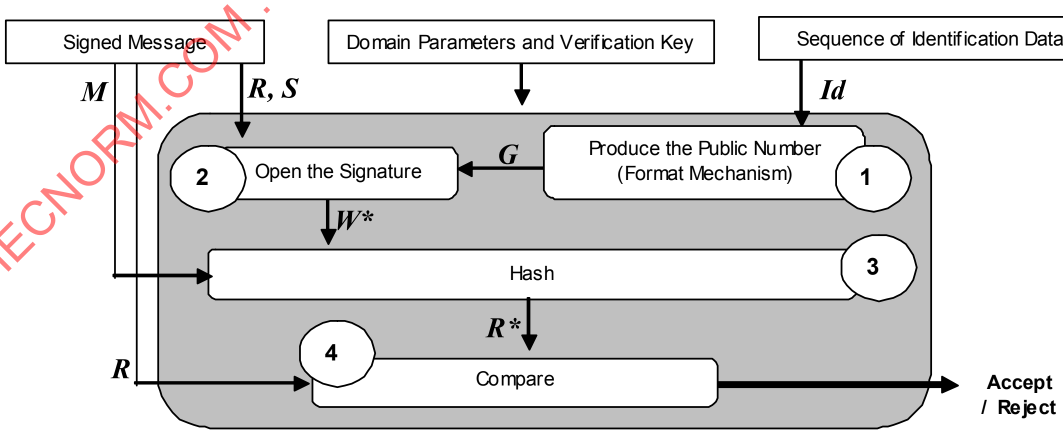

Figure 5 — Verifying with GQ1

Stage 1 — Convert Id into a representative of $\gamma = |n|$ bits in accordance with the format mechanism in use.

This bit string represents the public number G ($0 < G < n$).

NOTE Once formed, the number G can be cached for further use.

Stage 2 — Split R into t strings of $|v|-1$ bits as $R_1 \| R_2 \| \ldots \| R_t$ and S into t strings of $|n|$ bits as $S_1 \| S_2 \| \ldots \| S_t$. Each string R_i or S_i represents a number, also denoted R_i or S_i. Reject if $S_i = 0$ or $\geq n$.

For i from 1 to t, compute $S_i^{\,v} \times G^{R_i} \bmod n$ and represent it by a string of $|n|$ bits, denoted W_i*.

Form a string of $|n| \times t$ bits, denoted W*, with $W_1* \| W_2* \| \ldots \| W_t*$.

Stage 3 — Produce a bit string, denoted H*, in accordance with the hash-variant in use.

$H* =$ $\mathrm{h}(W* \| M)$ in the first variant
$\mathrm{h}(W* \| \mathrm{h}(M))$ in the second variant
$\mathrm{h}(\mathrm{h}(W*) \| M)$ in the third variant
$\mathrm{h}(\mathrm{h}(W*) \| \mathrm{h}(M))$ in the fourth variant

Form a bit string, denoted R*, with the leftmost $(|v|-1) \times t$ bits of H*.

Stage 4 — Accept or reject depending on whether R and $R*$ are identical or different.

7.4 Format mechanism [4]

Convert the sequence of identification data Id into a representative of γ bits, denoted F.

1) Hash Id into a bit string, denoted H. Concatenate 8 octets, all set to '00', on the left of the bit string H. Hash the concatenation into a bit string, denoted HH.

$$H = \mathrm{h}(Id) \qquad HH = \mathrm{h}('00000000\ 00000000' \| H)$$

2) Produce a string of at least $\gamma - |H|$ bits from HH by the following procedure making use of two variables: a string of variable length, denoted *String*, and a string of 32 bits, denoted *Counter*.

   a) Set *String* to the empty string.

   b) Set *Counter* to zero.

   c) Replace *String* by *String* || $\mathrm{h}(HH \|$ *Counter*).

   d) Replace *Counter* by *Counter* + 1.

   e) If $|H| \times$ *Counter* $< \gamma - |H|$, then go to stage c.

   Form a masked string with the leftmost $\gamma - |H|$ bits of *String* where the leftmost bit has been forced to 0 and the rightmost bit reversed.

3) Form F by concatenating the masked string on the left of HH.

$$F = \text{Masked string} \| HH$$

4) If the leftmost $\gamma - 1$ bits of F are all zeroes (a very unlikely case), then the procedure fails (the bit string Id is not appropriate). Otherwise, return F.

---

[4] This mechanism is due to Bellare and Rogaway [1]. The specific options are $\varepsilon = \tau = 0$ (no salt; no trailer; see 6.4). Its use is limited to a deterministic production of representatives.

8 GQ2 scheme [5]

8.1 Set of data elements required for signing/verifying

The subsequent relationships and constraints apply to the following data elements:

- a security parameter, a number of base numbers and a signature length parameter;
- a set of base numbers;
- a set of distinct prime factors;
- a modulus;
- an adaptation parameter;
- a set of private components;
- a set of private numbers.

The security parameter is denoted k. The number of base numbers is denoted m. The signature length parameter is denoted t. The product $k \times m \times t$ shall be less than or equal to $|H|$.

NOTE For $k \times m \times t = 80$, typical values of k, m and t are (80,1,1), (40,2,1), (20,4,1), (16,5,1), (10,8,1) and (8,10,1).

The set of base numbers is denoted $g_1, g_2 \ldots g_m$ in ascending order. They shall be m distinct prime numbers, less than 256.

The set of distinct prime factors is denoted $p_1, p_2 \ldots p_f$ in ascending order ($f > 1$). Each prime factor p_j has the unique form $p_j = 1 + q_j \times 2^{hj}$ where q_j is odd (i.e., $p_j - 1$ is divisible by 2^{hj}, but not by 2^{hj+1}).

At least one base number g_i and two prime factors p_j and p_{jj} shall have Legendre symbols as follows.

- If $h_j = h_{jj}$, then $(g_i \mid p_j) = -(g_i \mid p_{jj})$.
- If $h_j > h_{jj}$, then $(g_i \mid p_j) = -1$.

NOTE Key production occurs in either of two cases.

- Given a set of base numbers, e.g., the first prime numbers $2, 3, 5, 7, 11, 13, 17, 19 \ldots$, select a set of prime factors.
- Given a set of prime factors, e.g., the prime factors of an RSA or RW modulus, select a set of base numbers.

The modulus, denoted n, is the product of the prime factors ($n = p_1 \times \ldots \times p_f$). Its size shall be α bits.

The adaptation parameter, denoted b, is the greatest number from the f numbers h_1 to h_f.

The set of private components is denoted $Q_{1,1}$ to $Q_{m,f}$. For each prime factor p_j, m private components (one per base number g_i) are computed as follows.

$$s_j = \left(\frac{q_j+1}{2}\right)^{b+k} \bmod q_j\,; \qquad u_j = q_j - s_j\,; \qquad Q_{i,j} = g_i^{\,2^b \times u_j} \bmod p_j$$

The set of private numbers is denoted Q_1 to Q_m. For i from 1 to m, the number Q_i is the CRT composition (see 5.3) of $Q_{i,1}$ to $Q_{i,f}$.

---

[5] The GQ2 scheme is due to Guillou and Quisquater [11]. It makes use of zero-knowledge techniques for proving, without revealing, the knowledge of a decomposition of the modulus (see also ISO/IEC 9798-5 [30]).

NOTE 1 Alternatively: $q = \mathrm{lcm}(q_1, \ldots, q_f)$; $s = \left(\frac{q+1}{2}\right)^{b+k} \bmod q$; $u = q - s$; $Q_i = g_i^{2^b \times u} \bmod n$.

NOTE 2 For $v = 2^{b+k}$ and $G_i = g_i^{2^b}$, every pair G_i and Q_i verifies $G_i \times Q_i^v \bmod n = 1$.

NOTE 3 If $\chi_i = g_i \times Q_i^{2^k} \bmod n \neq 1$, then in the successive b squares $\bmod n$ of χ_i, the number preceding the unity is a square root $\bmod n$ of unity, denoted ω_i, that is either trivial ($\omega_i = n - 1$) or not ($1 < \omega_i < n - 1$). If g_i verifies the constraint on the Legendre symbols, then ω_i is non trivial, i.e., n divides $\omega_i^2 - 1$, but neither $\omega_i - 1$, nor $\omega_i + 1$, which provides the non trivial decomposition $n = \gcd(n, \omega_i - 1) \times \gcd(n, \omega_i + 1)$.

NOTE 4 If the Jacobi symbols are $(\pm g_i \mid n) = -1$ (which implies $n \equiv 1 \bmod 4$), then ω_i is non trivial.

Signing requires a hash-function (see 5.1), a hash-variant and a signature key. The signature key takes either of two forms:

- With CRT: k, t, b, p_1 to p_f, f–1 CRT coefficients (see 5.3) and $Q_{1,1}$ to $Q_{m,f}$ (k, t, b public).
- Without CRT: k, t, b, n and Q_1 to Q_m (k, t, b, n public).

Verifying requires a set of domain parameters and a verification key. The domain parameters shall include k, and may include m (by default, $m = 1$) and t (by default, $t = 1$). Either the domain parameters or the verification key shall include $Indic$(h), and may include $g_1, g_2 \ldots g_m$ (by default, the first m prime numbers, i.e., $2, 3, 5, 7, 11$ and so on), α (by default, $\alpha = |n|$) and $Indic$(variant) (by default, the first variant). The verification key shall include n and may include b (by default, $b = 1$).

8.2 Signature mechanism

Illustrated in Figure 6, the mechanism makes use of a hash-function, a hash-variant and a signature key, to sign a message (a bit string, denoted M), i.e., to produce a signature of M (two bit strings, denoted R and S).

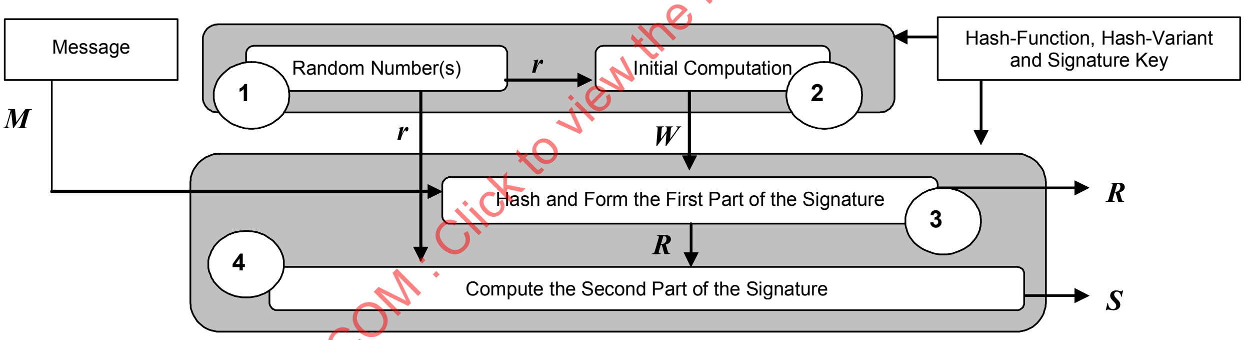

Figure 6 — Signing with GQ2

Stage 1 — Produce random numbers (globally denoted r in Figure 1) in either of two ways.

- With CRT, for j from 1 to f, select t strings of $|p_j|$ random bits. They represent numbers to be kept secret, denoted $r_{1,1}$ to $r_{t,f}$.

NOTE The probability that a string of $|p_j|$ random bits represents zero or p_i is so negligible that such a possibility need not be formally checked.

- Without CRT, select t strings of $|n|$ random bits. They represent numbers to be kept secret, denoted r_1 to r_t.

NOTE The probability that a string of $|n|$ random bits represents zero or a multiple of any prime factor of n is so negligible that such a possibility need not be formally checked.

Stage 2 — Produce bit strings, denoted W_1 to W_t, in either of two ways.

- With CRT, for i from 1 to t and j from 1 to f, compute $W_{i,j} = r_{i,j}^{2^{b+k}} \bmod p_j$. For i from 1 to t, represent the CRT composition (see 5.3) of $W_{i,1}$ to $W_{i,f}$ by a string of $|n|$ bits, denoted W_i.

- Without CRT, for i from 1 to t, compute $r_i^{2^{b+k}} \bmod n$ and represent it by a string of $|n|$ bits, denoted W_i.

Form a bit string, denoted W, with $W_1 \| W_2 \| \ldots \| W_t$ ($|n| \times t$ bits).

Stage 3 — Produce a bit string, denoted H, in accordance with the hash-variant in use.

$H =$ h($W \| M$) in the first variant
h($W \|$ h(M)) in the second variant
h(h(W) $\| M$) in the third variant
h(h(W) $\|$ h(M)) in the fourth variant

Form the first part of the signature, denoted R, with the leftmost $k \times m \times t$ bits of H.

Stage 4 — Split R into t strings of $k \times m$ bits as $R_1 \| R_2 \| \ldots \| R_t$. Split each R_i into m strings of k bits as $R_{i,1} \| R_{i,2} \| \ldots \| R_{i,m}$. Each string $R_{i,j}$ consists of k bits, from the leftmost bit, denoted $R_{i,j,1}$, to the rightmost bit, denoted $R_{i,j,k}$. Each string $R_{i,j}$ represents a number, also denoted $R_{i,j}$ ($< 2^k$).

Produce numbers, denoted S_1 to S_t, in either of two ways.

- With CRT, for i from 1 to t and j from 1 to f, compute $S_{i,j} = r_{i,j} \times Q_{1,j}^{R_{i,1}} \times \ldots Q_{m,j}^{R_{i,m}} \bmod p_j$. For i from 1 to t, the number S_i is the CRT composition (see 5.3) of $S_{i,1}$ to $S_{i,f}$.

- Without CRT, for i from 1 to t, compute $S_i = r_i \times Q_1^{R_{i,1}} \times \ldots Q_m^{R_{i,m}} \bmod n$.

Any number S_i may be replaced by $n - S_i$.

Represent each number S_i by a string of $|n|$ bits, also denoted S_i.

Form the second part of the signature, denoted S, with $S_1 \| S_2 \| \ldots \| S_t$ ($|n| \times t$ bits).

8.3 Verification mechanism

Illustrated in Figure 7, the mechanism makes use of a set of domain parameters and a verification key (see Table 1), with key precedence (see 5.2), to verify a message and a signature of that message, i.e., the three bit strings, denoted M, R and S.

Stage 0 — Reject if $|n| \neq \alpha$, or if $|R| \neq k \times m \times t$, or if $|S| \neq |n| \times t$, or if the m base numbers are not distinct prime numbers less than 256.

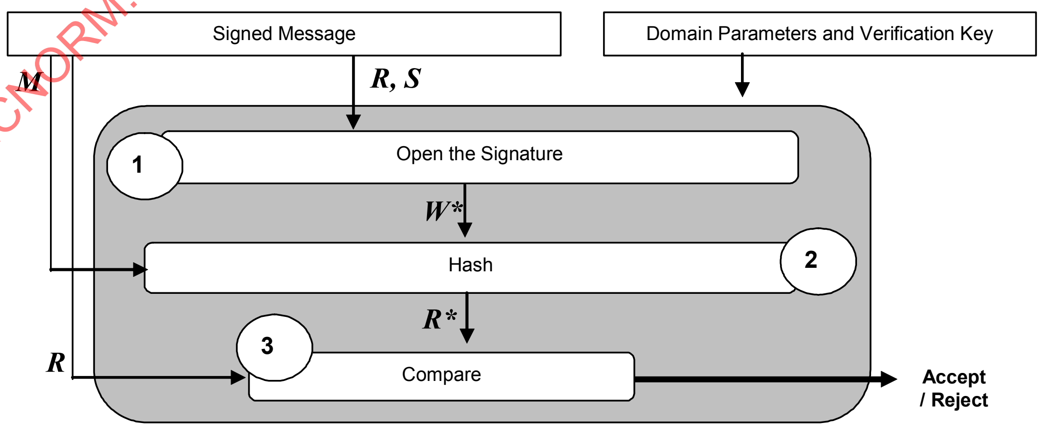

Figure 7 — Verifying with GQ2

Stage 1 — Split S into t strings of $|n|$ bits as $S_1 \| S_2 \| \ldots \| S_t$. Each bit string S_i represents a number, also denoted S_i. Reject if any $S_i = 0$ or $\geq n$.

Split R into t strings of $k \times m$ bits as $R_1 \| R_2 \| \dots \| R_t$. Split each R_i into m strings of k bits as $R_{i,1} \| R_{i,2} \| \dots \| R_{i,m}$. Each string $R_{i,j}$ consists of k bits, from the leftmost bit, denoted $R_{i,j,1}$, to the rightmost bit, denoted $R_{i,j,k}$. Each string $R_{i,j}$ represents a number, also denoted $R_{i,j}$ (< 2^k).

For i from 1 to t, compute $S_i^{2^{b+k}} \times (g_1^{2^b})^{R_{i,1}} \times \dots (g_m^{2^b})^{R_{i,m}} \bmod n$ and represent it by a string of $|n|$ bits, denoted W_i*.

NOTE Starting from a value set equal to S, k multiplications are interleaved with $b+k$ squares. The l-th square is followed by the l-th multiplication: for j from 1 to m, the appropriate bit (namely, $R_{i,j,l}$) states whether the current value is multiplied by g_j (bit set to 1) or not (bit set to 0). The final value, after the ($b+k$)-th square, gives W*.

Form a bit string, denoted W*, with $W_1{*} \| W_2{*} \| \dots \| W_t{*}$ ($|n| \times t$ bits).

Stage 2 — Produce a bit string, denoted H*, in accordance with the hash-variant in use.

H* = h(W* || M) in the first variant
h(W* || h(M)) in the second variant
h(h(W*) || M) in the third variant
h(h(W*) || h(M)) in the fourth variant

Form a bit string, denoted R*, with the leftmost $k \times m \times t$ bits of H*.

Stage 3 — Accept or reject depending on whether R and R* are identical or different.

9 GPS1 scheme [6]

9.1 Set of data elements required for signing/verifying

The subsequent relationships and constraints apply to the following data elements:

- a modulus;
- a set of prime factors;
- a private number;
- a base number;
- a public number.

The modulus is denoted n. Its size shall be α bits. Its decomposition into prime factors need not be known.

If available, the set of prime factors is denoted $p_1, p_2 \dots p_f$ in ascending order ($f > 1$).

The private number, denoted Q, is represented by a string of $|H|$ random bits.

The base number is denoted g. The values $g = 0$ and $g = 1$ are forbidden.

NOTE The value $g = 2$ has some practical advantages.

The public number, denoted G, is produced in either of two ways.

- With CRT, for i from 1 to f, compute $Q_i = Q \bmod (p_i - 1)$ and $G_i = g^{Q_i} \bmod p_i$. The number G is the CRT composition (see 5.3) of G_1 to G_f.

---

[6] The GPS1 scheme is due to Girault, Poupard and Stern [5, 17]. It makes use of zero-knowledge techniques for proving, without revealing, the knowledge of a private number (see also ISO/IEC 9798-5 [30]).

- Without CRT, compute $G = g^{Q} \bmod n$.

Signing requires a hash-function (see 5.1), a hash-variant and a signature key. The signature key takes either of two forms:

- With CRT: p_1 to p_f, f–1 CRT coefficients (see 5.3), g and Q (g public);
- Without CRT: n, g and Q (n, g public).

Verifying requires a set of domain parameters and a verification key. Either the domain parameters or the verification key shall include n and $Indic$(h), and may include g (by default, $g = 2$), α (by default, $\alpha = |n|$) and $Indic$(variant) (by default, the third variant). The verification key shall include G.

9.2 Signature mechanism

9.2.1 General

Illustrated in Figure 8, the mechanism makes use of a hash-function, a hash-variant and a signature key, to sign a message (a bit string, denoted M), i.e., to produce a signature of M (two bit strings, denoted R and S).

Every signer shall be equipped with one or more coupons. By definition, a coupon is a bit string, independent from the message, pre-computed from a string of random bits, to be kept secret and to be used only once.

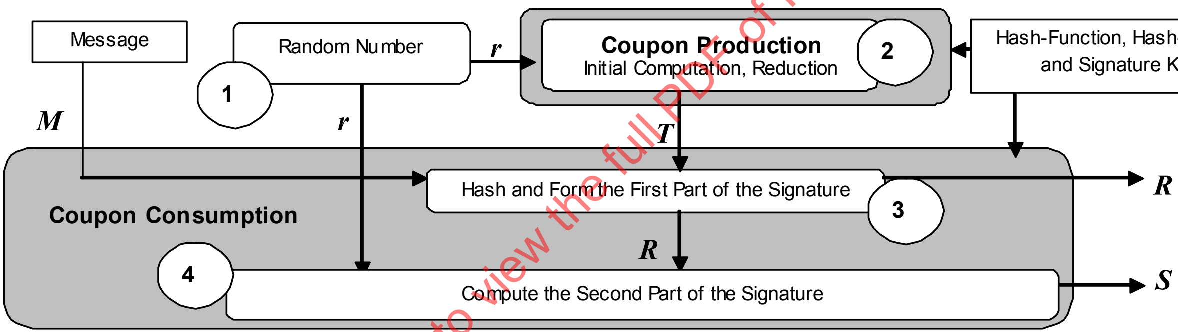

Figure 8 — Signing with GPS1

9.2.2 Random number

Stage 1 — Select a string of $2\,|H| + 80$ random bits.

It represents a random number to be kept secret, denoted r.

The random number production is associated with either coupon production or coupon consumption.

- Stage 2 makes use of r and either n or p_1 to p_f.
- Stage 4 makes use of r and Q.

NOTE If the signing device produces strings of pseudorandom bits as a deterministic function of the coupon indices, then it stores the index of the last produced coupon and the index of the last consumed coupon. Otherwise, it stores the bit strings until consuming the coupons.

9.2.3 Coupon production

Stage 2 — Produce a bit string, denoted W, in either of two ways.

- With CRT, for i from 1 to f, compute $r_i = r \bmod (p_i - 1)$ and $W_i = g^{r_i} \bmod p_i$. Represent the CRT composition (see 5.3) of W_1 to W_f by a string of $|n|$ bits, denoted W.

- Without CRT, compute $g^r \bmod n$ and represent it by a string of $|n|$ bits, denoted W.

The coupon, denoted T, is set equal to the hash-code $h(W)$.

9.2.4 Coupon consumption

Stage 3 — Produce the first part of the signature, denoted R, in accordance with the hash-variant in use.

$R =$ $h(T \| M)$, i.e., $= h(h(W) \| M)$ in the third variant
$h(T \| h(M))$, i.e., $= h(h(W) \| h(M))$ in the fourth variant

Stage 4 — The bit string R represents a number, also denoted R.

Compute $S = r - R \times Q$.

The second part of the signature, also denoted S, is the string of $2\,|H| + 80$ bits representing the number S.

9.3 Verification mechanism

Illustrated in Figure 9, the mechanism makes use of a set of domain parameters and a verification key (see Table 1), with key precedence (see 5.2), to verify a message and a signature of that message, i.e., the three bit strings, denoted M, R and S.

Stage 0 — Reject if $|n| \neq \alpha$, or if $g = 0$ or 1, or if $|R| \neq |H|$, or if $|S| \neq 2\,|H| + 80$.

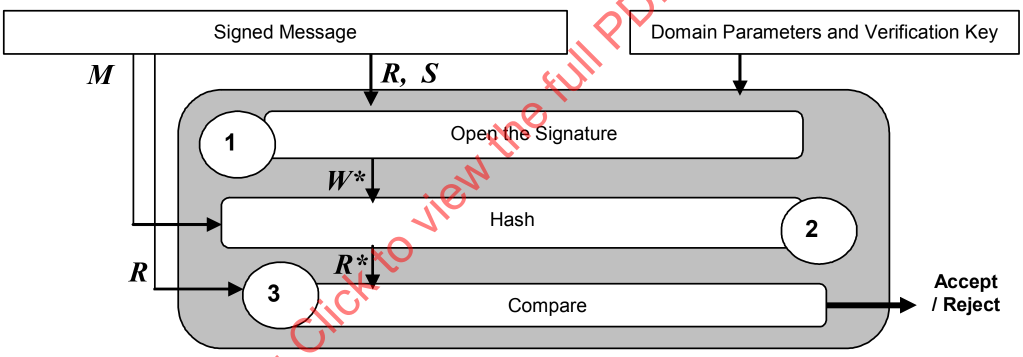

Figure 9 — Verifying with GPS1

Stage 1 — The bit strings R and S represent two numbers, also denoted R and S.

Compute $G^R \times g^S \bmod n$ and represent it by a string of $|n|$ bits, denoted W^*.

Stage 2 — Produce a bit string, denoted R^*, in accordance with the hash-variant in use.

$R^* =$ $h(h(W^*) \| M)$ in the third variant
$h(h(W^*) \| h(M))$ in the fourth variant

NOTE The hash-code $h(W^*)$ is the recovered coupon.

Stage 3 — Accept or reject depending on whether R and R^* are identical or different.

10 GPS2 scheme [7]

10.1 Set of data elements required for signing/verifying

The subsequent relationships and constraints apply to the following data elements:

- a verification exponent;
- a set of distinct prime factors;
- a modulus;
- a private number;
- a base number.

The verification exponent, denoted v, shall be a prime number so that $|v| = |H| + 1$.

NOTE If $|H| = 160$, then the value $v = 2^{160}+7$ has some practical advantages.

The set of distinct prime factors is denoted $p_1, p_2 \ldots p_f$ in ascending order ($f > 1$).

For i from 1 to f, v shall not divide $p_i - 1$.

The modulus, denoted n, is the product of the prime factors ($n = p_1 \times \ldots \times p_f$). Its size shall be α bits.

The private number is denoted Q. It is any positive integer (the least one is often used) so that $v \times Q - 1$ is a multiple of $\mathrm{lcm}(p_1 - 1, \ldots p_f - 1)$. It is represented by a string of $|n|$ bits.

NOTE The number Q has the same definition as the signature exponent specified in 6.1.

The base number is denoted g. The values $g = 0$ and $g = 1$ are forbidden.

NOTE The value $g = 2$ has some practical advantages.

Signing requires a hash-function (see 5.1), a hash-variant and a signature key. The signature key takes either of two forms:

- With CRT: v, p_1 to p_f, $f-1$ CRT coefficients (see 5.3), Q and g (v, g public);
- Without CRT: v, n, Q and g (v, n, g public).

Verifying requires a set of domain parameters and a verification key. Either the domain parameters or the verification key shall include v and $Indic(\mathrm{h})$, and may include g (by default, $g = 2$), α (by default, $\alpha = |n|$) and $Indic$(variant) (by default, the third variant). The verification key shall include n.

10.2 Signature mechanism

10.2.1 General

Illustrated in Figure 10, the mechanism makes use of a hash-function, a hash-variant and a signature key, to sign a message (a bit string, denoted M), i.e., to produce a signature of M (two bit strings, denoted R and S).

Every signer shall be equipped with one or more coupons. By definition, a coupon is a bit string, independent from the message, pre-computed from a string of random bits, to be kept secret and to be used only once.

---

[7] The GPS2 scheme is due to Girault and Paillès [6]. It makes use of zero-knowledge techniques for proving, without revealing, the knowledge of the RSA signature exponent (see also ISO/IEC 9798-5 [30]).

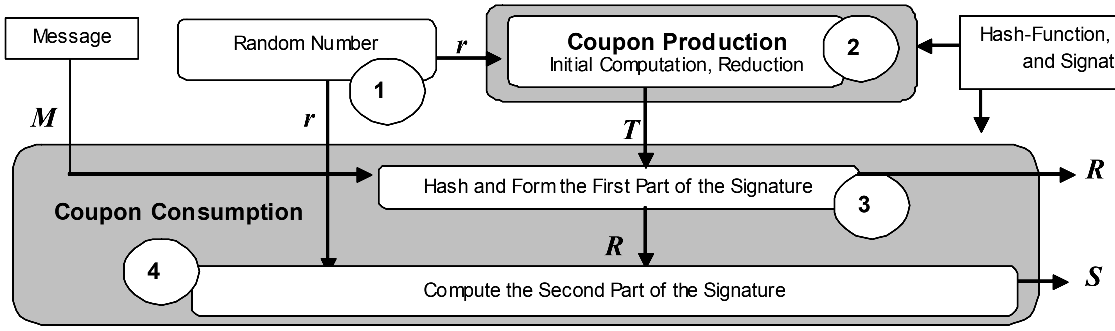

Figure 10 — Signing with GPS2

10.2.2 Random number

Stage 1 — Select a string of $|n| + |H| + 80$ random bits.

It represents a random number to be kept secret, denoted r.

The random number production is associated with either coupon production or coupon consumption.

- Stage 2 makes use of r, v and either n or p_1 to p_f.
- Stage 4 makes use of r and Q.

NOTE If the signing device produces strings of pseudorandom bits as a deterministic function of the coupon indices, then it stores the index of the last produced coupon and the index of the last consumed coupon. Otherwise, it stores the bit strings until consuming the coupons.

10.2.3 Coupon production

Stage 2 — Produce a bit string, denoted W, in either of two ways.

- With CRT, for i from 1 to f, compute $r_i = v \times r \bmod (p_i - 1)$ and $W_i = g^{r_i} \bmod p_i$. Represent the CRT composition (see 5.3) of W_1 to W_f by a string of $|n|$ bits, denoted W.
- Without CRT, compute $g^{v \times r} \bmod n$ and represent it by a string of $|n|$ bits, denoted W.

The coupon, denoted T, is set equal to the hash-code $\mathrm{h}(W)$.

10.2.4 Coupon consumption

Stage 3 — Produce the first part of the signature, denoted R, in accordance with the hash-variant in use.

$R =$ $\mathrm{h}(T \| M)$, i.e., $= \mathrm{h}(\mathrm{h}(W) \| M)$ in the third variant
$\mathrm{h}(T \| \mathrm{h}(M))$, i.e., $= \mathrm{h}(\mathrm{h}(W) \| \mathrm{h}(M))$ in the fourth variant

Stage 4 — The bit string R represents a number, also denoted R.

Compute $S = r - R \times Q$.

The second part of the signature, also denoted S, is the string of $|n| + |H| + 80$ bits representing the number S.

10.3 Verification mechanism

Illustrated in Figure 11, the mechanism makes use of a set of domain parameters and a verification key (see Table 1), with key precedence (see 5.2), to verify a message and a signature of that message, i.e., the three bit strings, denoted M, R and S.

Stage 0 — Reject if $|n| \neq \alpha$, or if v is not odd and prime, or if $g = 0$ or $= 1$, or if $|R| \neq |H|$, or if $|S| \neq |n| + |H| + 80$.

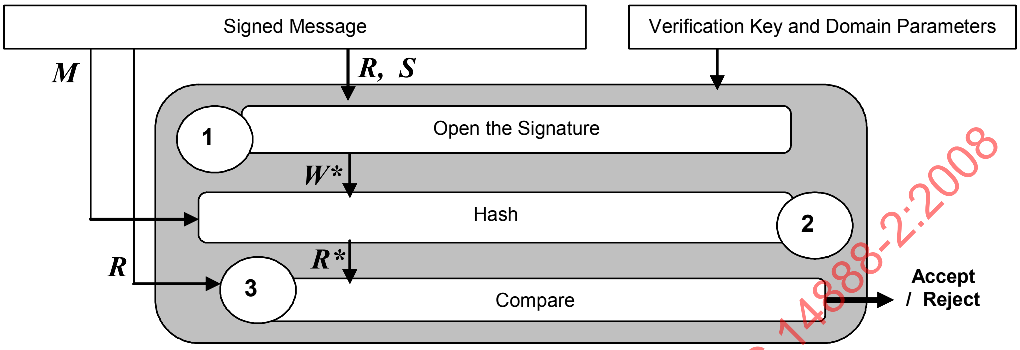

Figure 11 — Verifying with GPS2

Stage 1 — The bit strings R and S represent two numbers, also denoted R and S.

Compute $g^{v \times S + R} \bmod n$ and represent it by a string of $|n|$ bits, denoted W^*.

Stage 2 — Produce a bit string, denoted R^*, in accordance with the hash-variant in use.

$R^* =$ $\mathrm{h}(\mathrm{h}(W^*) \,\|\, M)$ in the third variant
$\mathrm{h}(\mathrm{h}(W^*) \,\|\, \mathrm{h}(M))$ in the fourth variant

NOTE The hash-code $\mathrm{h}(W^*)$ is the recovered coupon.

Stage 3 — Accept or reject depending on whether R and R^* are identical or different.

11 ESIGN scheme [8]

11.1 Set of data elements required for signing/verifying

The subsequent relationships and constraints apply to the following data elements:

- a verification exponent;
- a pair of distinct prime factors;
- a modulus.

The verification exponent, denoted v, shall be greater than or equal to eight, but less than $2^{\alpha-1}$.

NOTE The value $v = 1024$ has some practical advantages.

The pair of distinct prime factors is denoted p_1 and p_2 in ascending order. The size of each prime factor shall be α / 3 bits (α shall be a multiple of three).

[8] The ESIGN scheme is due to Fujioka, Okamoto and Miyaguchi [3]. It makes use of the approximate v-th root problem.

NOTE For example, α = 1023 (and not 1024), 1536, 2046 or 2049 (and not 2048), 2304.

The modulus, denoted n, is the product $p_1 \times p_2^2$ (repeat the largest prime factor). Its size shall be α bits.

- The greatest common divisor of v and n shall be 1, i.e., $\gcd(v, n) = 1$.
- The greatest common divisor of v, $p_1 - 1$ and $p_2 - 1$ shall be at most α, i.e., $\gcd(v, p_1 - 1, p_2 - 1) \leq \alpha$.

NOTE Any value of v less than or equal to α satisfies both constraints.

Signing requires a hash-function (see 5.1), a format mechanism and a signature key. The format mechanism specified in 11.4 is recommended. The signature key consists of v, p_1 and p_2 (v public).

Verifying requires a set of domain parameters and a verification key. Either the domain parameters or the verification key shall include v and $Indic(\mathrm{h})$, and may include α (by default, $\alpha = |n|$) and $Indic(\mathrm{format}, \varepsilon, \tau)$ (by default, 11.4). The verification key shall include n.

11.2 Signature mechanism

Illustrated in Figure 12, the mechanism makes use of a hash-function, a format mechanism and a signature key, to sign a message (a bit string, denoted M), i.e., to produce a signature of M (a bit string, denoted S).

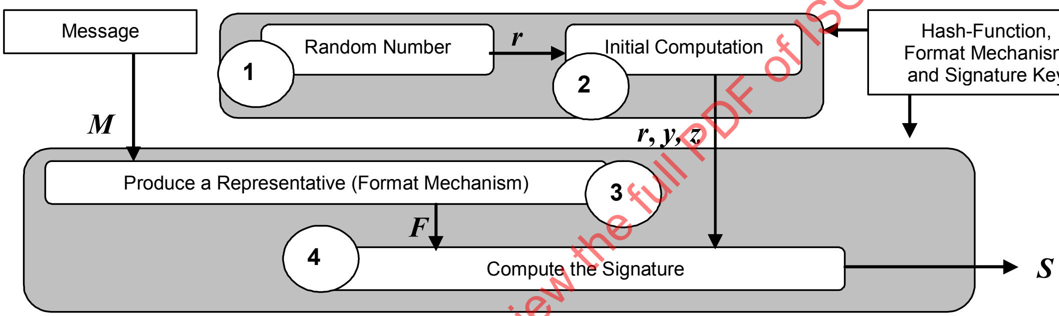

Figure 12 — Signing with ESIGN

Stage 1 — Select a string of $2\,|n|\,/\,3$ random bits.

It represents a random number to be kept secret, denoted r. The number r shall be less than $p_1 \times p_2$.

NOTE The probability that a string of $2\,|n|\,/\,3$ random bits represents zero or a multiple of any prime factor of n is so negligible that such a possibility need not be formally checked.

Stage 2 — Compute two numbers, denoted y ($< n$) and z ($< p_2$). The number z shall be kept secret.

$$y = r^{v} \bmod n$$
$$z = (v \times r^{v-1})^{-1} \bmod p_2$$

NOTE The formula $z = r \times (v \times y)^{-1} \bmod p_2$ has some computational advantages.

Stage 3 — Convert the message M into a representative of $\gamma = |n|\,/\,3$ bits, denoted F, in accordance with the format mechanism in use. The bit string F represents a number, also denoted F ($0 < F < p_1$).

Stage 4 — Compute a number, denoted S ($0 < S < n$).

$$a = (2^{2\,|n|\,/\,3}\, F - y) \bmod n$$
$$w = \lceil a\,/\,(p_1 \times p_2) \rceil$$

If $w \times p_1 \times p_2 - a \geq 2^{(2\,|n|\,/\,3)\,-1}$ (a case occurring at most half the time in general), then return to stage 1.

$$S = r + (w \times z \bmod p_2) \times p_1 \times p_2 \bmod n$$

If v is even, then the number S may be replaced by $n - S$.

The signature, also denoted S, is any bit string representing the number S, often a string of $|n|$ bits.

11.3 Verification mechanism

Illustrated in Figure 13, the mechanism makes use of a set of domain parameters and a verification key (see Table 1), with key precedence (see 5.2), to verify a message and a signature of that message, i.e., the two bit strings, denoted M and S.

Stage 0 — Reject if α is not a multiple of three, or if $|n| \neq \alpha$, or if $v < 8$, or if $v \geq 2^{\alpha-1}$.

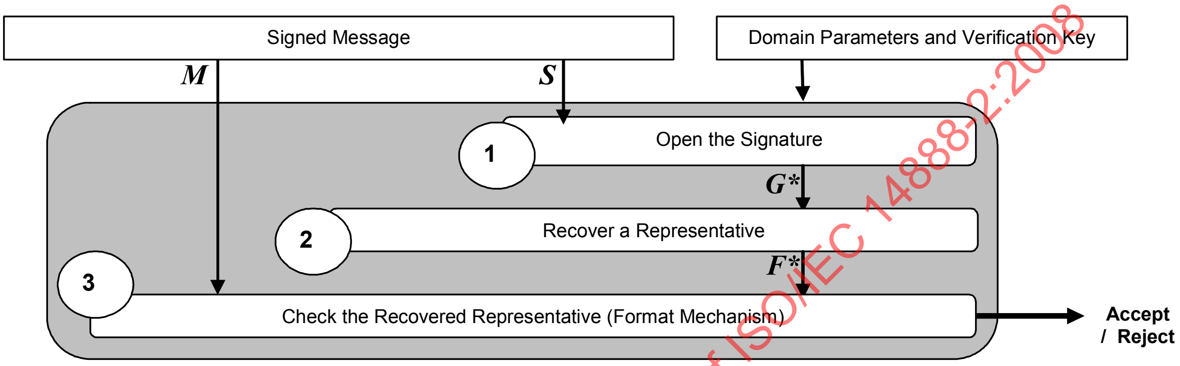

Figure 13 — Verifying with ESIGN

Stage 1 — The bit string S represents a number, also denoted S. Reject if $S = 0$ or 1, or if $S \geq n-1$.

Compute $S^v \bmod n$ and represent it by a string of $|n|$ bits, denoted G^*.

Stage 2 — Recover a representative, denoted F^*, as the leftmost $\gamma = |n| / 3$ bits of G^*.

Stage 3 — Check the recovered representative F^* in accordance with the format mechanism in use.

11.4 Format mechanism [9]

Convert the message M into a representative of γ bits, denoted F.

1) Select a fresh string of $\varepsilon = |H|$ random bits. It forms a salt, denoted E.

2) Hash M into a bit string, denoted H. From left to right, concatenate 8 octets, all set to '00', H and the salt E. Hash the concatenation into a bit string, denoted HH.

$$H = \mathrm{h}(M) \qquad HH = \mathrm{h}('0000\ 0000\ 0000\ 0000' \,\|\, H \,\|\, E)$$

3) Produce a string of at least $\gamma - |H|$ bits from HH by the following procedure making use of two variables: a bit string of variable length, denoted *String*, and a string of 32 bits, denoted *Counter*.

   a) Set *String* to the empty string.

   b) Set *Counter* to zero.

   c) Replace *String* by *String* $\|$ $\mathrm{h}(HH \,\|\, Counter)$.

   d) Replace *Counter* by *Counter* $+ 1$.

   e) If $|H| \times Counter < \gamma - |H|$, then go to stage c.

---

[9] This mechanism is due to Bellare and Rogaway [1]. The specific options are $\varepsilon = |H|$ and $\tau = 0$ (each signature requires a fresh salt; no trailer; see 6.4). The ESIGN scheme combined with this format mechanism is known as ESIGN-PSS, where PSS stands for "Probabilistic Signature Scheme".

Form a mask with the leftmost $\gamma - |H|$ bits of *String* where the leftmost bit has been forced to 0.

4) Form an intermediate string of $\gamma - |H|$ bits from left to right by concatenating:

— $\gamma - |H| - \varepsilon - 1$ bits, all zeroes;

— a border bit, set to 1;

— the salt E.

5) By exclusive-oring, apply the mask to the intermediate string, thereby producing a masked string.

6) Form F by concatenating the masked string on the left of HH.

$$F = \text{Masked string} \mathbin{||} HH$$

7) If the γ bits of F are all zeroes (a very unlikely case), then return to stage 1 (the salt E is not appropriate). Otherwise, return F.

Check a recovered representative of γ bits, denoted F^*, with respect to the message M.

1) If the γ bits of F^* are all zeroes, then reject. Otherwise, continue.

2) From the rightmost $|H|$ bits of F^*, denoted HH^*, produce a mask of $\gamma - |H|$ bits as step 3 above.

3) By exclusive-oring, apply the mask to the leftmost $\gamma - |H|$ bits of F^*, thereby producing a recovered intermediate string where, starting from the left, the border bit is the first bit that is set to 1.

— If ε bits remain on the right of the border bit in the recovered intermediate string, then they form a bit string, denoted E^*.

— Otherwise, reject.

4) Hash M into a bit string, denoted H. From left to right, concatenate 8 octets, all set to '00', H and E^*. Hash the concatenation into a bit string, denoted HH.

$$H = \mathrm{h}(M) \qquad HH = \mathrm{h}(\text{'0000 0000 0000 0000'} \mathbin{||} H \mathbin{||} E^*)$$

5) Accept or reject depending on whether HH and HH^* are identical or different.

Annex A
(normative)

Object identifiers

A.1 Formal definition

Table A.1 summarizes the options specified in this document.

Table A.1 — Options specified in this document

Scheme	Format mechanism a)	Hash-variant
RSA	formatPSS, formatD1, formatD2	novariant
RW	formatPSS, formatD1, formatD2	novariant
GQ1	formatPSS	variant1, variant2, variant3, variant4
GQ2	noformat	variant1, variant2, variant3, variant4
GPS1	noformat	variant3, variant4
GPS2	noformat	variant3, variant4
ESIGN	formatPSS	novariant

a) This document specifies three implementations of formatPSS: 6.4 ($\varepsilon = 0$ or $|H|$ and $\tau = 8$ or 16) for RSA and RW, 7.4 ($\varepsilon = \tau = 0$) for GQ1 and 11.4 ($\varepsilon = |H|$ and $\tau = 0$) for ESIGN. Annex D specifies formatD1 ($\varepsilon = 0$ and $\tau = 8$ or 16) and formatD2 ($\varepsilon = 64$ and $\tau = 8$).

The following module is in compliance with the notation specified in ISO/IEC 8824-1 [25].

```
IntegerFactorizationBasedDigitalSignaturesWithAppendix {
   iso(1) standard(0) digital-signatures-with-appendix(14888) part2(2)
      asn1-module(1) integer-factorization-based-mechanisms(0) version1(1)}

DEFINITIONS EXPLICIT TAGS ::= BEGIN

EXPORTS ALL;

IMPORTS

   HashFunctions
      FROM DedicatedHashFunctions {
         iso(1) standard(0) hash-functions(10118) part(3)
            asn1-module(1) dedicated-hash-functions(0) } ;

SignatureWithAppendix ::= SEQUENCE {
   algorithm  ALGORITHM.&id({SchemeOptions}),
   parameters ALGORITHM.&Type({SchemeOptions}{@algorithm}) OPTIONAL
}

SchemeOptions ALGORITHM ::= {
   RSA      |
   RW       |
   GQ1      |
   GQ2      |
   GPS1     |
   GPS2     |
   ESIGN,
```

```
   ...   -- Expect additional signature scheme objects --
}

-- Integer factorization signature scheme object sets --


-- RSA scheme options --

RSA ALGORITHM ::= {
   rsa-formatPSS-novariant   |
   rsa-formatD1-novariant  |
   rsa-formatD2-novariant,

   ...   -- Expect additional RSA scheme objects --
}

rsa-formatPSS-novariant ALGORITHM ::= {
   OID id-rsa-formatPSS-novariant PARMS HashFunctions
}

rsa-formatD1-novariant ALGORITHM ::= {
   OID id-rsa-formatD1-novariant PARMS HashFunctions
}

rsa-formatD2-novariant ALGORITHM ::= {
   OID id-rsa-formatD2-novariant PARMS HashFunctions
}


-- RW scheme options --

RW ALGORITHM ::= {
   rw-formatPSS-novariant   |
   rw-formatD1-novariant  |
   rw-formatD2-novariant,

   ...   -- Expect additional RW scheme objects --
}

rw-formatPSS-novariant ALGORITHM ::= {
   OID id-rw-formatPSS-novariant PARMS HashFunctions
}

rw-formatD1-novariant ALGORITHM ::= {
   OID id-rw-formatD1-novariant PARMS HashFunctions
}

rw-formatD2-novariant ALGORITHM ::= {
   OID id-rw-formatD2-novariant PARMS HashFunctions
}


-- GQ1 scheme options --

GQ1 ALGORITHM ::= {
   gq1-formatPSS-variant1  |
   gq1-formatPSS-variant2  |
   gq1-formatPSS-variant3  |
   gq1-formatPSS-variant4,
```



```
   ...   -- Expect additional GQ1 scheme objects --
}

gq1-formatPSS-variant1 ALGORITHM ::= {
   OID id-gq1-formatPSS-variant1 PARMS HashFunctions
}

gq1-formatPSS-variant2 ALGORITHM ::= {
   OID id-gq1-formatPSS-variant2 PARMS HashFunctions
}

gq1-formatPSS-variant3 ALGORITHM ::= {
   OID id-gq1-formatPSS-variant3 PARMS HashFunctions
}

gq1-formatPSS-variant4 ALGORITHM ::= {
   OID id-gq1-formatPSS-variant4 PARMS HashFunctions
}


-- GQ2 scheme options --

GQ2 ALGORITHM ::= {
   gq2-noformat-variant1  |
   gq2-noformat-variant2  |
   gq2-noformat-variant3  |
   gq2-noformat-variant4,

   ...   -- Expect additional GQ2 scheme objects --
}

gq2-noformat-variant1 ALGORITHM ::= {
   OID id-gq2-noformat-variant1 PARMS HashFunctions
}

gq2-noformat-variant2 ALGORITHM ::= {
   OID id-gq2-noformat-variant2 PARMS HashFunctions
}

gq2-noformat-variant3 ALGORITHM ::= {
   OID id-gq2-noformat-variant3 PARMS HashFunctions
}

gq2-noformat-variant4 ALGORITHM ::= {
   OID id-gq2-noformat-variant4 PARMS HashFunctions
}


-- GPS1 scheme options --

GPS1 ALGORITHM ::= {
   id-gps1-noformat-variant3 |
   id-gps1-noformat-variant4,

   ...   -- Expect additional GPS1 scheme objects --
}
```

```
gps1-noformat-variant3 ALGORITHM ::= {
   OID id-gps1-noformat-variant3 PARMS HashFunctions
}

gps1-noformat-variant4 ALGORITHM ::= {
   OID id-gps1-noformat-variant4 PARMS HashFunctions
}


-- GPS2 scheme options --

GPS2 ALGORITHM ::= {
   id-gps2-noformat-variant3 |
   id-gps2-noformat-variant4,

   ...   -- Expect additional GPS2 scheme objects --
}

gps2-noformat-variant3 ALGORITHM ::= {
   OID id-gps2-noformat-variant3 PARMS HashFunctions
}

gps2-noformat-variant4 ALGORITHM ::= {
   OID id-gps2-noformat-variant4 PARMS HashFunctions
}


-- ESIGN scheme options --

ESIGN ALGORITHM ::= {
   esign-formatPSS-novariant,

   ...   -- Expect additional ESIGN scheme objects --
}

esign-formatPSS-novariant ALGORITHM ::= {
   OID id-esign-formatPSS-novariant PARMS HashFunctions
}



-- Cryptographic algorithm identification --

ALGORITHM ::= CLASS {
   &id    OBJECT IDENTIFIER  UNIQUE,
   &Type  OPTIONAL
}
  WITH SYNTAX { OID &id [PARMS &Type] }

OID ::= OBJECT IDENTIFIER -- alias

is14888-2 OID ::= {
   iso(1) standard(0) digital-signatures-with-appendix(14888) part2(2) }

signatureScheme OID ::= { is14888-2 scheme(0) }

-- Integer factorization signature scheme identifiers --
```

```
rsa   OID ::= { signatureScheme rsa(1) }
rw    OID ::= { signatureScheme rw(2) }
gq1   OID ::= { signatureScheme gq1(3) }
gq2   OID ::= { signatureScheme gq2(4) }
gps1  OID ::= { signatureScheme gps1(5) }
gps2  OID ::= { signatureScheme gps2(6) }
esign OID ::= { signatureScheme esign(7) }

-- Table A.1 format mechanism option types

noformat RELATIVE-OID ::= { noformat(0) }
formatPSS  RELATIVE-OID ::= { formatPSS(1) }
formatD1 RELATIVE-OID ::= { formatD1(10) }  -- see D.2
formatD2 RELATIVE-OID ::= { formatD2(11) }  -- see D.3

-- Table A.1 hash-variant option types

novariant RELATIVE-OID ::= { novariant(0) }
variant1  RELATIVE-OID ::= { variant1(1) }
variant2  RELATIVE-OID ::= { variant2(2) }
variant3  RELATIVE-OID ::= { variant3(3) }
variant4  RELATIVE-OID ::= { variant4(4) }

-- Table A.1 integer factorization signature scheme options

id-rsa-formatPSS-novariant OID ::= { rsa formatPSS novariant }
id-rsa-formatD1-novariant OID ::= { rsa formatD1 novariant }
id-rsa-formatD2-novariant OID ::= { rsa formatD2 novariant }

id-rw-formatPSS-novariant OID ::= { rw formatPSS novariant }
id-rw-formatD1-novariant OID ::= { rw formatD1 novariant }
id-rw-formatD2-novariant OID ::= { rw formatD2 novariant }

id-gq1-formatPSS-variant1 OID ::= { gq1 formatPSS variant1 }
id-gq1-formatPSS-variant2 OID ::= { gq1 formatPSS variant2 }
id-gq1-formatPSS-variant3 OID ::= { gq1 formatPSS variant3 }
id-gq1-formatPSS-variant4 OID ::= { gq1 formatPSS variant4 }

id-gq2-noformat-variant1 OID ::= { gq2 noformat variant1 }
id-gq2-noformat-variant2 OID ::= { gq2 noformat variant2 }
id-gq2-noformat-variant3 OID ::= { gq2 noformat variant3 }
id-gq2-noformat-variant4 OID ::= { gq2 noformat variant4 }

id-gps1-noformat-variant3 OID ::= { gps1 noformat variant3 }
id-gps1-noformat-variant4 OID ::= { gps1 noformat variant4 }

id-gps2-noformat-variant3 OID ::= { gps2 noformat variant3 }
id-gps2-noformat-variant4 OID ::= { gps2 noformat variant4 }

id-esign-formatPSS-novariant  OID ::= { esign formatPSS novariant }

END -- IntegerFactorizationBasedDigitalSignaturesWithAppendix --
```

NOTE In accordance with the Basic Encoding Rules of ASN.1 (see ISO/IEC 8825-1 [26]), each identifier is one or more series of octets; bit 8 (the most significant bit) is 0 in the last octet of a series and 1 in the previous octets if the series is several octets. The concatenation of bits 7 to 1 of the octets of a series codes an integer. Each integer shall be encoded on the fewest possible octets, that is, the octet '80' is invalid in the first position of a series.

- The first octet is set equal to '28', i.e., 40 in decimal, for identifying an ISO standard (see ISO/IEC 8825-1 [26]).

- The subsequent two octets are set equal to 'F428'. As 14888 is equal to '3A28' in hexadecimal, i.e., 0011 1010 0010 1000, i.e., two blocks of seven bits: 1110100 0101000. After insertion of the appropriate value of bit 8 in each octet, the coding of the series is therefore 11110100 00101000, i.e., 'F428'.
- The subsequent octet is set equal to '02' for identifying Part 2.
- The subsequent octet is set equal to '00' for identifying the arc denoted mechanism(0).
- The subsequent octet identifies a signature scheme with a value from '01' to '07'.
- The subsequent octet identifies a format mechanism with a value from '00', '01', '0A' or '0B' in accordance with Table A.1.
- The subsequent octet identifies a hash-variant with a value from '00' to '04' in accordance with Table A.1.

EXAMPLE 1 The data element '28 F4 28 02 00 01 01 00' means {iso standard 14888 2 0 1 1 0}, i.e., the first signature scheme with PSS as format mechanism, within ISO/IEC 14888-2, i.e., RSA-PSS. It may be conveyed in a BER-TLV data object with the universal class tag '06'.

Data object = {'06 08 28 F4 28 02 00 01 01 00'}

EXAMPLE 2 The data element '28 F4 28 02 00 03 01 01' means {iso standard 14888 2 0 3 1 1}, i.e., the third signature scheme (GQ1) with PSS as format mechanism and the first hash-variant ($\mathrm{h}(W \parallel M)$), within ISO/IEC 14888-2. It may be conveyed in a BER-TLV data object with the universal class tag '06'.

Data object = {'06 06 28 F4 28 02 00 03 01 01'}

EXAMPLE 3 The data element '28 F4 28 02 00 04 00 02' means {iso standard 14888 2 0 4 0 4}, i.e., the fourth signature scheme (GQ2) with the second hash-variant ($\mathrm{h}(W \parallel \mathrm{h}(M))$), within ISO/IEC 14888-2. It may be conveyed in a BER-TLV data object with the universal class tag '06'.

Data object = {'06 06 28 F4 28 02 01 04 00 04'}

EXAMPLE 4 The data element '28 F4 28 02 00 07 01 00' means {iso standard 14888 2 0 7 1 0}, i.e., the seventh signature scheme with PSS as format mechanism, within ISO/IEC 14888-2 (ESIGN-PSS). It may be conveyed in a BER-TLV data object with the universal class tag '06'.

Data object = {'06 06 28 F4 28 02 00 07 01 00'}

A.2 Use of subsequent object identifiers

Each signature scheme specified in this document uses a hash-function, a sequence containing a hash-function algorithm identifier and any associated parameters. Therefore, the signature scheme object identifier may be followed by one of the dedicated hash-function algorithm identifiers specified in ISO/IEC 10118-3 and any associated parameters.

Using the ASN.1 XML value notation, the ESIGN-PSS scheme (format mechanism 1 and no hash-variant, as defined in this document), and the SHA-256 hash-function defined in ISO/IEC 10118-3, a value of type SignatureWithAppendix would be represented as:

```
<SignatureWithAppendix>
   <algorithm> 1.0.14888.2.0.7.1.0 </algorithm>
   <parameters>
      <HashFunctions>
         <algorithm> 2.16.840.1.101.3.4.2.1 </algorithm>
         <parameters/>
       </HashFunctions>
   </parameters>
</SignatureWithAppendix>
```

Annex B
(informative)

Guidance on parameter choice and comparison of signature schemes

B.1 Guidance on parameter choice

B.1.1 Modulus sizes

In this document, every signature scheme makes use of a modulus that is the product of large prime factors, at least two of them being distinct. All the prime factors should be of roughly the same size.

In 1995, Odlyzko [16] estimated the future difficulty of integer factorization. As a conclusion at the end of the quoted article [16], Kaliski stressed the importance of variable sizes for the moduli in implementations and provided recommendations on modulus sizes: 768 bits for short term security, 1 024 bits for medium term security, and 2 048 bits for long term security. For a comprehensive analysis of modulus sizes, see also Silverman [20], and Lenstra and Verheul [14].

Table B.1 specifies three security ranges for the moduli ($|n|$ bits): medium, long and very long terms. It also sets related conditions in terms of the modulus size in bits; these conditions are about the prime factors ($|p|$ bits), the hash-codes ($|H|$ bits) and the first part of a signature (the leftmost $|R|$ bits of the output of the hash-variant): any infringement compromises security.

- If a prime factor p is too small, it can be recovered.
- If the hash-code H is too short, it is possible to find two bit strings having the same hash-code.
- If the first part R is too short, it is possible to sign without knowing the private number(s) (see B.1.3).

Table B.1 — Conditions on $|p|$, $|H|$ and $|R|$ in terms of $|n|$

$\|n\|$	$\|p\|$	$\|H\|$ RSA, RW, GQ1, ESIGN	$\|H\|$ GQ2, GPS1, GPS2	$\|R\|$ GQ1, GQ2, GPS1, GPS2
From 750 to 1599	> 340	≥ 160	≥ 128	≥ 80
From 1600 to 2999	> 510	≥ 224	≥ 160	≥ 112
From 3000 to 4999	> 680	≥ 256	≥ 192	≥ 144

B.1.2 Modulus and prime factors

Throughout the standard, the number of prime factors is denoted f and the large prime factors are denoted p_1, p_2 ... p_f in ascending order. The modulus is the product of the prime factors.

$$n = p_1 \times p_2 \times \ldots p_f$$

For practical advantages, the modulus size should be a multiple of f and then, while in accordance with Table B.1, every prime factor should be of the same size.

NOTE In ESIGN, the largest prime factor is repeated: $n = p_1 \times {p_2}^2$.

The following method defines successive variable intervals for successively selecting large prime factors, the size of which is denoted π. Hereafter the current value of the product of the prime factors is denoted z.

- The first prime factor is selected within the interval from $2^{\pi-1}$ to 2^{π}. The initial value of z is set equal to the first prime factor.

- This stage is repeated $f-1$ times. A new prime factor is selected within the interval from $(2^{|z|}/z)\ 2^{\pi-1}$ to 2^{π}. The current value of z is multiplied by the new prime factor.
- The prime factors are denoted p_1 to p_f in ascending order, and n is set equal to the final value of z.

The following method defines a single interval, of slightly reduced size, for selecting every prime factor.

- Every prime factor is selected within the interval from $\gamma\ 2^{\pi}$ to 2^{π}, where γ denotes the f-th root of 1/2.

NOTE The value of γ may be approximated by a rational number greater than γ (e.g., 5/7 for the square root of 1/2, 4/5 for the cube root of 1/2).

B.1.3 Schemes making use of zero-knowledge techniques

B.1.3.1 Zero-knowledge triples

Goldwasser, Micali and Rackoff [7] introduced the concept of zero-knowledge. The GQ1, GQ2, GPS1 and GPS2 schemes make use of zero-knowledge techniques.

NOTE ISO/IEC 9798-5 [30] specifies authentication mechanisms using zero-knowledge techniques.

For example, in GQ1, the following stages aim at proving the knowledge of a private number Q.

- Select a random positive integer r $(0 < r < n)$.
- Compute $W = r^{v} \bmod n$ $(0 < W < n)$.
- In response to any challenge R $(0 \le R < v)$, compute $S = r \times Q^{R} \bmod n$ $(0 < S < n)$.

In the GQ1 scheme, the verifier reveals integer R as a challenge after receiving witness W. On his side, the verifier computes another witness $W^* = S^{v} \times G^{R} \bmod n$. The authentication is successful if and only if witnesses W and W^* are identical and non zero.

Consequently, the set of all the GQ1 triples $\{W, R, S\}$ may be seen as a set of v permutations (indexed by R) of the ring of the integers modulo n. The permutation for $R = 0$ is the RSA permutation.

B.1.3.2 Generic security

To derive a signature scheme from such a zero-knowledge authentication exchange, the interaction with the verifier is eliminated. The number W is represented by a string of $|n|$ bits, also denoted W. Firstly, the bit strings W and M are hashed (e.g., $\mathrm{h}(W \| M)$, see the hash-variants in 5.2) Secondly, the number represented by the resulting bit string is reduced, e.g., by $\bmod\ v$ reduction, to an integer R from 0 to $v-1$. The global operation is denoted as $R = \mathrm{R}(W, M)$. The bit string M is associated with a ZK triple $\{W, R, S\}$. The signature is practically limited to the pair (R, S).

The GQ1 triple shall be both valid and linked to M.

- The GQ1 triple is valid if and only if $0 < W < n$ and W identical to $W^* = S^{v} \times G^{R} \bmod n$.
- The GQ1 triple is linked to M if and only if $0 \le R < v$ and R identical to the bit string $R^* = \mathrm{R}(W^*, M)$.

The following attack aims at evaluating the appropriate size of the GQ1 verification exponent v when $t = 1$. For a given message M and any S from 1 to $n-1$, for $i = 1, 2, \ldots$, and so on, compute $x = S^{v} \times G^{i} \bmod n$, then $y = \mathrm{R}(x, M)$, less than v, in accordance with the specified method, and so on, until $y = i$. Then as the GQ1 triple $\{S^{v} \times G^{i} \bmod n, i, S\}$ is valid and linked to the message M, the signed message (M, i, S) is valid. Such an attack requires an amount of computation of the order of v. Consequently, to avoid such an attack due to the existing computing power, Table B.1 indicates the minimum length of R. All the possible values of R shall be equally probable.

A similar attack applies to the GQ2 scheme.

A similar attack also applies to the GPS1 and GPS2 schemes. Even if the reduction of the length of R does not reduce the workloads, the coupon length should be reduced as much as possible. For consistency, the coupon and R should have the same length.

B.1.3.3 Random parameter sizes

In the GQ1, GQ2, GPS1 and GPS2 schemes, a random parameter r is converted into an initial element W and then a second part of signature S is computed in response to any first part of signature R. W, R and S form a ZK triple, denoted $\{W, R, S\}$, satisfying a relationship for opening signatures. The set of all the ZK triples is a family of R permutations of the set (or a subset) of the ring of the integers modulo n.

It is important that the signer chooses random parameters in such a way that the probability of predicting their value and the probability of the same value being selected twice within the signer's lifetime are negligible. If, for example, a signer uses the same value twice, then he will create an interlocked pair of triples, i.e., responses to two challenges for the same non-zero witness, denoted $\{W, R_1, S_1\}$ and $\{W, R_2, S_2\}$. Such a pair is sometimes named a claw (see [8]) in the family of permutations.

- In the GQ1, GPS1 and GPS2 schemes, the private number is easily deduced from any interlocked pair of triples. The private number allows impersonating the signer.
- The GQ2 key constraint ensures that, for any values of m and k, more than one half of all the interlocked pairs of triples reveal a non-trivial square root $\bmod n$ of unity. Such a number induces a decomposition of n, i.e., the factorization if $f = 2$. The factors allow impersonating the signer.

The relationship for opening signatures computes a witness W from any pair (R, S) selected at random, i.e., producing triples at random. It is important that the set of all the ZK triples is so large that the advantage obtained by producing in advance as many triples as possible remains negligible.

As a conclusion, the random bit strings are strings of:

- $|n|$ bits in the GQ1 and GQ2 schemes;
- $2\,|H| + 80$ bits in the GPS1 scheme;
- $|n| + |H| + 80$ bits in the GPS2 scheme.

B.2 Comparison of signature schemes

B.2.1 Symbols and abbreviated terms

The comparison makes use of the following measures: the size of the set of the data elements for signing, the complexity of the computations for signing, the complexity of the computations for verifying and the size of the set of the data elements for verifying.

NOTE If the signer is a portable device (e.g., an integrated circuit card [24]), then the complexity of computation and communication, and the required storage may be crucial, since the processing and storage capacities of the cards are very limited in comparison with those allowed for the verifier.

For the purposes of this annex, the following symbols and abbreviated terms apply.

$\mathrm{HW}(v)$ number of bits set to 1 in the binary representation of v, e.g., $\mathrm{HW}(65\,537 = 2^{16}+1) = 2$

M_α computational complexity of a modular multiplication

X_α computational complexity of a modular square

π bit size of each prime factor ($\pi = |p_1| = |p_2|$)

B.2.2 Complexity of modular operations

This clause evaluates the computational complexity of modular operations, namely the modular multiplication, the modular square, the modular exponentiation and the combined modular exponentiation.

The **modular multiplication** is defined as $A \times B \bmod C$. It may be performed as two consecutive operations: a multiplication followed by a reduction. In practice, the workload due to the multiplication is approximately equal to the workload due to the reduction.

- When A and B have the same size as C, the multiplication provides a result that is twice as long as C.
- The reduction provides the remainder of the division of the result by C.

When A and B have the same size as C, the modular multiplication complexity is denoted $M_{|C|}$.

If the modulus is f times longer than the prime factors, i.e., $\alpha = f \times \pi$, then the ratio between a multiplication modulo n and a multiplication modulo a prime factor is approximately f^2 ($M_\alpha \approx f^2\, M_\pi$). Consequently, the value of $M_{|C|}$ is proportional to $|C|^2$.

For example, if there are two prime factors, i.e., $\alpha = 2\,\pi$, then $M_\alpha \approx 4\, M_\pi$.

The **modular square** is defined as $A^2 \bmod C$. It may be performed as two consecutive operations: a square followed by a reduction.

- When A has the same size as C, the square provides a result that is twice as long as C. According to Menezes, van Oorschot and Vanstone [15], the complexity of the square is half that of the multiplication.

  NOTE As $A \times B = ((A+B)^2 - (A-B)^2) / 4$, the multiplication may result from using twice a squaring routine.

- The reduction provides the remainder of the division of the result by C. The complexity of the reduction is as above.

When A has the same size as C, the modular square complexity is denoted $X_{|C|}$.

$$X_{|C|} \approx 0{,}75\, M_{|C|}$$

The **modular exponentiation** is defined as $A^B \bmod C$. It may be performed as the square and multiply algorithm [13, 15], i.e., $|B| - 1$ modular squares and $\mathrm{HW}(B) - 1$ modular multiplications by A.

The **combined modular exponentiation** is defined as $A_1^{B1} \times \ldots \times A_x^{Bx} \bmod C$. It may be performed as $\max\{|B_1|, \ldots |B_x|\} - 1$ modular squares and $\mathrm{HW}(B_1) + \ldots + \mathrm{HW}(B_x) - 1$ modular multiplications by A_i.

- If A_i is small (i.e., $|A_i| \le 8$), then the modular multiplications due to B_i are negligible in comparison with the modular squares.
- The modular exponentiation is either short, or medium, or long, in accordance with the exponent size in bits ($\max\{|B_1|, \ldots, |B_x|\}$) is either small (up to 40), or medium (80, 160, 240 to 280), or large ($|C|$, $|C|+80$ up to $|C|+120$).

B.2.3 Complexity of the CRT technique with two prime factors of the same size

The CRT composition involves a modular multiplication modulo a factor, and one multiplication of two integers of the same size as a factor, resulting in an integer of the same size as the modulus. When the two factors have the same size, e.g., $\pi = |p_1| = |p_2| = \alpha / 2$, the composition complexity is denoted ChC.

$$ChC \approx 1{,}5\, M_\pi \approx (3/8)\, M_\alpha$$

The CRT decomposition involves two reductions modulo a factor. When the two factors have the same size, e.g., $\pi = |p_1| = |p_2| = \alpha / 2$, the decomposition complexity is denoted ChD.

$$ChD \approx M_\pi \approx 0{,}25\, M_\alpha$$

For example, the CRT technique reduces the complexity of production of one RSA or RW signature from one exponentiation $\bmod n$ (i.e., $(5/4)\,\alpha M_\alpha$) to one ChD plus two exponentiations $\bmod p_i$ (with exponents reduced $\bmod p_i - 1$) plus one ChC (i.e., $(1 + 2{,}5\,\pi + 1{,}5)\, M_\pi = 2{,}5\,(\pi + 1)\, M_\pi$). As $\alpha = 2\,\pi$ and $M_\alpha \approx 4\, M_\pi$, the reduced complexity is $\approx (5/16)\,\alpha M_\alpha$.

B.2.4 Complexity analysis

B.2.4.1 RSA and RW

For signing without CRT: n and s	$2\ \alpha$ bits
Signature computation: the s-th power mod n	$(\lvert s\rvert -1)\ X_\alpha + (\mathrm{HW}(s) -1)\ M_\alpha$
Total ($\lvert s\rvert = \alpha$, $\mathrm{HW}(s) = \alpha / 2$ and $X_\alpha = 0{,}75\ M_\alpha$)	$(5/4)\ \alpha\ M_\alpha$
For signing with CRT: p_1, p_2, Cr, s_1 and s_2	$2{,}5\ \alpha$ bits
Signature computation: the s_i-th power mod p_i	$(5/16)\ \alpha\ M_\alpha$
For verifying: n, v	α bits (v negligible)
RSA signature opening: the v-th power mod n	$(\lvert v\rvert -1)\ X_\alpha + (\mathrm{HW}(v) -1)\ M_\alpha$
Total	$(0{,}75\ \lvert v\rvert + \mathrm{HW}(v) -1{,}75)\ M_\alpha$
EXAMPLE $13\ M_\alpha$ if $v = 2^{16}+1$, and $1{,}75\ M_\alpha$ if $v = 3$	
RW signature opening: the square mod n ($X_\alpha \approx 0{,}75\ M_\alpha$)	$0{,}75\ M_\alpha$

B.2.4.2 GQ1

For signing: n, v, t, Q	$2\ \alpha$ bits (v, t negligible)
Initial computation: $W_i = r_i^{\ v} \bmod n$	$(\lvert v\rvert -1)\ X_\alpha + (\mathrm{HW}(v) -1)\ M_\alpha$
Second part of signature: $S_i = r_i \times Q^{Ri} \bmod n$	$M_\alpha + (\lvert R_i\rvert -1)\ X_\alpha + (\mathrm{HW}(R_i) -1)\ M_\alpha$
As $\lvert R_i\rvert = \lvert v\rvert -1$ and $\mathrm{HW}(R_i) = (\lvert v\rvert -1)/2$,	$(2\ \lvert v\rvert + \mathrm{HW}(v) - 3{,}75)\ M_\alpha$
Repeated t times,	$(t \times (2\ (\lvert v\rvert -1) + \mathrm{HW}(v) -1{,}75))\ M_\alpha$
Total	$(2\ (\lvert v\rvert -1) \times t + t \times (\mathrm{HW}(v) -1{,}75))\ M_\alpha$
For verifying: n, v and t	α bits (v, t negligible)
Signature opening: $W_i{*} = S_i^{\ v} \times G^{Ri} \bmod n$	$(\lvert v\rvert -1)\ X_\alpha + (\mathrm{HW}(R_i) + \mathrm{HW}(v) -1)\ M_\alpha$
As $\lvert R_i\rvert = \lvert v\rvert -1$ and $\mathrm{HW}(R_i) = (\lvert v\rvert - 1)/2$,	$(1{,}25\ \lvert v\rvert + \mathrm{HW}(v) - 2{,}25)\ M_\alpha$
Repeated t times,	$(t \times (1{,}25\ (\lvert v\rvert -1) + \mathrm{HW}(v) -1))\ M_\alpha$
Total	$(1{,}25\ (\lvert v\rvert -1) \times t + t \times (\mathrm{HW}(v) -1))\ M_\alpha$

B.2.4.3 GQ2

For signing without CRT: n, k and b, Q_1 to Q_m	$(m+1)\ \alpha$ bits (k, b negligible)
Initial computation: $W = r^{2^{b+k}} \bmod n$	$(k + b)\ X_\alpha$
Second part of signature: $S = r \times \prod_{i=1}^{m} Q_i^{R_i} \bmod n$	$(\lvert R_i\rvert_{\max} -1)\ X_\alpha + (\mathrm{HW}(R_1) + \ldots \mathrm{HW}(R_m) -1)\ M_\alpha + M_\alpha$
As $\lvert R_i\rvert = k$ and $\mathrm{HW}(R_m) = k / 2$,	$(k -1)\ X_\alpha + 0{,}5\ k\ m\ M_\alpha$
Total	$(0{,}5\ k\ (m + 3) + 0{,}75\ (b -1))\ M_\alpha$
For signing with CRT: p_1, p_2, Cr, k and b, $Q_{1,1}$ to $Q_{m,2}$	$(m+1{,}5)\ \alpha$ bits (k, b negligible)
Initial computation: $W_i = r_i^{2^{k+b}} \bmod p_i$	$2\ (k + b)\ X_\pi + ChC$
Second part of signature: $S_j = r_j \times \prod_{i=1}^{m} Q_{i,j}^{R_i} \bmod p_j$	$2\ (\lvert R_i\rvert_{\max} -1)\ X_\pi + 2\ (\mathrm{HW}(R_1) + \ldots \mathrm{HW}(R_m))\ M_\pi + ChC$
As $\lvert R_i\rvert = k$ and $\mathrm{HW}(R_m) = k / 2$,	$2\ (k -1)\ X_\pi + k\ \ m\ \ M_\pi + ChC$
Total ($ChC \approx 1{,}5\ M_\pi$ and $M_\pi \approx M_\alpha / 4$)	$(0{,}25\ k\ (m + 3) + 0{,}375\ (b +1))\ M_\alpha$
For verifying: n, k, b, g_1 to g_m	α bits (k, b, g_1 to g_m negligible)
Signature opening: $W{*} = S^{2^{k+b}} \times \prod_{i=1}^{m} (g_i^{2^b})^{R_i} \bmod n$	$(k + b)\ X_\alpha$ ($\times\ g_1$ to g_m negligible)
Total	$(0{,}75\ (k + b))\ M_\alpha$

B.2.4.4 GPS1

For producing coupons without CRT: n	α bits
Initial computation: $W = 2^r \mod n$	$(\lvert r\rvert - 1)\ X_\alpha$
Total $(\lvert r\rvert = 2\ \lvert H\rvert + 80)$	$(1,5\ \lvert H\rvert + 60)\ M_\alpha$
For producing coupons with CRT: p_1, p_2, Cr	$1,5\ \alpha$ bits
Initial computation: $W_i = 2^r \mod p_i$	$2\ (\lvert r\rvert - 1)\ X_\pi + ChC$
As $\lvert r\rvert < 0,5 \times \lvert n\rvert$ and $ChC \approx 1,5 \times M_\pi$	
Total $(M_\pi \approx M_\alpha / 4)$	$(0,75\ \lvert H\rvert + 30)\ M_\alpha$
Coupons and element for consuming coupons: Q	$\lvert H\rvert$ bits + ($\lvert H\rvert$ bits per coupon)
Second part of signature: $S = r - R \times Q$	$0,5\ (\lvert R\rvert / \alpha)\ (\lvert Q\rvert / \alpha)\ M_\alpha$
Coupon consumption $(\lvert R\rvert = \lvert H\rvert$ and $\lvert Q\rvert = \lvert H\rvert)$	$0,5\ (\lvert H\rvert / \alpha)^2\ M_\alpha$
For verifying: n and G	$2\ \alpha$ bits
Signature opening: $W^* = 2^S \times G^R \mod n$	$(\lvert S\rvert - 1)\ X_\alpha + (\mathrm{HW}(R) - 1)\ M_\alpha$
Total $(\mathrm{HW}(R) = \lvert H\rvert / 2$ and $\lvert S\rvert = 2\ \lvert H\rvert + 80)$	$(2\ \lvert H\rvert + 60)\ M_\alpha$

B.2.4.5 GPS2

For producing coupons without CRT: n and v	α bits (v negligible)
Initial computation: $W_j = 2^{r \times v} \mod n$	$(\lvert r\rvert + \lvert v\rvert)\ X_\alpha$
Total $(\lvert r\rvert = \alpha + \lvert H\rvert + 80)$	$0,75\ (\alpha + 2\ \lvert H\rvert + 80)\ M_\alpha$
For producing coupons with CRT: p_1, p_2, Cr, v	$1,5\ \alpha$ bits (v negligible)
Initial computation: $W_i = 2^{r \times v \bmod p_i - 1} \mod p_i$	$2\ (\pi - 1)\ X_\pi + ChC$
As $2 \times \pi = \alpha$ and $ChC \approx 1,5 \times M_\pi$,	$0,75\ \alpha M_\pi$
Total $(M_\pi \approx M_\alpha / 4)$	$(3 / 16)\ \alpha M_\alpha$
Coupons and element for consuming coupons: Q	α bits + (β bits per coupon)
Second part of signature: $S = r - R \times Q$	$0,5\ (\lvert R\rvert / \alpha)\ (\lvert Q\rvert / \alpha)\ M_\alpha$
As $\lvert R\rvert = \lvert H\rvert$; $\lvert Q\rvert = \alpha$,	$0,5\ (\lvert H\rvert / \alpha)\ M_\alpha$
For verifying: n and v	α bits (v negligible)
Signature opening: $W^* = 2^{R + v \times S} \mod n$	$(\lvert S \times v\rvert - 1)\ X_\alpha + (\mathrm{HW}(S \times v) - 1)\ M_\alpha$
Total $(\lvert S \times v\rvert = \alpha + \lvert H\rvert + 80$, $\mathrm{HW}(S \times v) = (\alpha + \lvert H\rvert + 80) / 2)$	$1,25\ (\alpha + \lvert H\rvert + 80)\ M_\alpha$

B.2.4.6 ESIGN

For signing: v, p_1 and p_2 $(\lvert p_1\rvert = \lvert p_2\rvert = \alpha / 3)$	$0,67\ \alpha$ bits (v negligible)
$p_1 \times p_2$	$0,5\ M_\pi$
$\times p_2)$	M_π
$y = r^v \mod (p_1 \times p_2 \times p_2)$	$2\ (\lvert v\rvert - 1)\ X_\alpha + 2\ (\mathrm{HW}(v) - 1)\ M_\alpha$
$z = ((r \mod p_2) \times (v \times (y \mod p_2))^{-1}) \mod p_2$	$2\ (4\ M_\pi + I_\pi)$
$S = \lceil (2^{2 \times \alpha / 3} \times F - y) / (p_1 \times p_2) \rceil$	$2\ M_\alpha$
$\times z \mod p_2$	$2\ M_\pi$
$\times p_1 \times p_2 + r \mod (p_1 \times p_2 \times p_2)$	$2\ M_\alpha$
Total (assuming $I_\pi = 10\ M_\pi$ and $M_\alpha = 9\ M_\pi$)	$(1,5\ \lvert v\rvert + 2\ \mathrm{HW}(v) + 4)\ M_\alpha$
For verifying: n and v	α bits (v negligible)
Signature opening: $S^v \mod n$	$(\lvert v\rvert - 1)\ X_\alpha + (\mathrm{HW}(v) - 1)\ M_\alpha$
Total	$(0,75\ \lvert v\rvert + \mathrm{HW}(v) - 1,75)\ M_\alpha$

B.2.4.7 Summary of the evaluations

Table B.2 summarizes the evaluations detailed from B.2.4.1 to B.2.4.6.

Table B.2 — Summary of the evaluations

	Signature mechanism			Verification mechanism	
	CRT	Storage (bits)	Complexity (M_α)	Storage (bits)	Complexity (M_α)
RSA/RW	No	$2\ \alpha$	$1{,}25\ \alpha$	α	**RSA :** $0{,}75\ \lvert v\rvert + \mathrm{HW}(v) - 1{,}75$
	Yes	$2{,}5\ \alpha$	$0{,}3125\ \alpha$		**RW :** $0{,}75$
GQ1	No	$2\ \alpha$	$2\ (\lvert v\rvert - 1) \times t + t \times (\mathrm{HW}(v) - 1{,}75)$	α	$1{,}25\ (\lvert v\rvert - 1) \times t + t \times (\mathrm{HW}(v) - 1)$
GQ2	No	$(m+1)\ \alpha$	$0{,}5\ k\ (m+3) + 0{,}75\ (b-1)$	α	$0{,}75\ (k+b)$
	Yes	$(m+1{,}5)\ \alpha$	$0{,}25\ k\ (m+3) + 0{,}375\ (b+1)$		
GPS1-P	No	α	$1{,}5\ \lvert H\rvert + 60$	$2 \times \alpha$	$2\ \lvert H\rvert + 60$
	Yes	$1{,}5\ \alpha$	$0{,}75\ \lvert H\rvert + 30$		
GPS1-C	No	$\lvert H\rvert + (\lvert H\rvert$ per coupon)	$0{,}5\ \lvert H\rvert / \alpha$		
GPS2-P	No	α	$0{,}75\ (\alpha + 2\ (\lvert v\rvert - 1)) + 60$	α	$1{,}25\ (\alpha + \lvert H\rvert) + 100$
	Yes	$1{,}5\ \alpha$	$0{,}1875\ \alpha$		
GPS2-C	No	$\alpha + (\lvert H\rvert$ per coupon)	$0{,}5\ \lvert H\rvert / \alpha$		
ESIGN	No	$0{,}67\ \alpha$	$1{,}5\ \lvert v\rvert + 2\ \mathrm{HW}(v) + 4$	α	$0{,}75\ \lvert v\rvert + \mathrm{HW}(v) - 1{,}75$

— For producing GQ1 and ESIGN signatures, the CRT technique is irrelevant.

— The production of GPS1 and GPS2 signatures implies two stages: coupon production, denoted P, and coupon consumption, denoted C. The coupons are produced in advance, possibly in another device.

B.2.4.8 Complexity for different modulus sizes

In addition to $|H| = 160$ (e.g., RIPEMD-160 and SHA-1), the comparison makes use of the following values.

RSA — $v = 2^{16}+1$, $\varepsilon = 160$ and $\tau = 0$ (i.e., $|v| = 17$, $\mathrm{HW}(v) = 2$, a salt of 160 bits and no trailer)

RW — $v = 2$, $\varepsilon = 160$ and $\tau = 0$ (i.e., $|v| = 2$, $\mathrm{HW}(v) = 1$, a salt of 160 bits and no trailer)

GQ1 — $v = 2^{80}+13$ and $t = 1$ (i.e., $(|v| - 1) \times t = 80$ and $\mathrm{HW}(v) = 4$)

GQ2 — $b = 1$, $m = 10$ and $k = 8$ (base numbers = the first ten prime numbers: 2 to 29, and $k \times m = 80$)

GPS1 — $g = 2$ ($|R| = |Q| = |H| = 160$, $|G| = \alpha$ and $|r| = 2\ |H| + 80 = 400$)

GPS2 — $g = 2$, $v = 2^{160}+7$ (i.e., $|R| = 160$, $|Q| = \alpha$ and $|r| = \alpha + |H| + 80 = \alpha + 240$)

ESIGN — $v = 2^{10}$ (i.e., $|v| = 11$ and $\mathrm{HW}(v) = 1$)

Table B.3 compares the complexity for different sizes of the modulus: $\alpha = 1\ 024$, $1\ 536$ and $2\ 048$. The unit for the complexity is M_{1024} ($M_{1536} \approx 2{,}25\ M_{1024}$; $M_{2048} \approx 4\ M_{1024}$).

To use the mechanisms specified for GQ1, GQ2, GPS1 and GPS2 for authentication, the verifier transmits a challenge and waits for a response within a limited delay, as specified in Annex F. Table B.3 includes authentication with the following size for the first part of the signature, denoted R. Then the response is not a signature: it is a “non transmissible” proof.

- For GQ1, $m = 1$ and $v = 2^{16}+1$ ($|R| = 16$).
- For GQ2, $b = 1$, $k = 8$ and $m = 2$ ($|R| = 16$).

Table B.3 — Complexity for different modulus sizes

		Signature mechanism (M_{1024})			Verification mechanism (M_{1024})		
	CRT	$\alpha = 1024$	$\alpha = 1536$	$\alpha = 2048$	$\alpha = 1024$	$\alpha = 1536$	$\alpha = 2048$
RSA/RW	No	1280	4320	10240	**RSA** 13	29,25	52
	Yes	320	1080	2560	**RW** 0,75	1,69	3
GQ1	No	162,25	365,06	649	103	231,75	412
Authentication	*Yes*	*34,25*	*77,06*	*137,00*	*22,25*	*50,06*	*89,00*
GQ2	No	52,00	117,00	208	6,75	15,19	27
	Yes	26,75	60,19	107			
Authentication	*Yes*	*10,75*	*24,19*	*43*	*6,75*	*15,19*	*27*
GPS1-P	No	300	675	1200	380	855	1520
	Yes	150	337,5	600			
GPS1-C	No	0,012					
GPS2-P	No	1068	3267	7344	1580	4995	11440
	Yes	192	648	1536			
GPS2-C	No	0,078	0,117	0,156			
ESIGN	No	22,5	50,63	90	7,5	16,88	30

Annex C
(informative)

Numerical examples

C.1 RSA-PSS scheme

C.1.1 Message with salt

Data elements for signing/verifying — The size of each prime factor is 512 bits. The size of the modulus is 1024 bits. The verification exponent is $v = 3$ (it divides neither p_1–1 nor p_2–1).

$p_1 =$ CC109249 5D867E64 065DEE3E 7955F2EB C7D47A2D 7C995338 8F97DDDC 3E1CA19C
35CA659E DC3D6C08 F64068EA FEDBD911 27F9CB7E DC174871 1B624E30 B857CAAD

$p_2 =$ D8CD81F0 35EC57EF E8229551 49D3BFF7 0C53520D 769D6D76 646C7A79 2E16EBD8
9FE6FC5B 6060BD97 8ED64A90 59C5B039 98A0E94C 86D78B85 BA37B5AF D987505F

$s =$ 1CCDA20B CFFB8D51 7EE96668 66621B11 822C7950 D55F4BB5 BEE37989 A7D17312
E326718B E0D62CCB 11415F78 B36BE2E6 0D599D4E 41346C82 D845498A 81B2F663
2FD7D1CC EFCABF74 17350238 109EC289 D5382762 B77A1C99 96DD1D2B 71A52FAF
52ABA9DE D19F3F5D 5D71D054 73EC9C79 92D84128 0BAC72B8 7BF51EB1 CCB65C87

$n =$ ACD1CC46 DFE54FE8 F9786672 664CA269 0D0AD7E5 003BC642 7954D939 EEE8B271
52E6A947 45050CC2 67883CD4 34875164 5019AFD5 873A8B11 119FB93F 0A31C654
C3ECFF07 3233530C 79BE90E0 26E2421D D378B88B 40136C48 7D33075A 1612AB90
C5B75D33 2659A5D0 B5C19576 102D3424 31AC3BBB A8F98449 BD58BC0B 5E254633

Signature — The message is a string of 114 octets. The salt is a string of 20 octets.

$M =$ 859EEF2F D78ACA00 308BDC47 1193BF55 BF9D78DB 8F8A672B 484634F3 C9C26E64
78AE1026 0FE0DD8C 082E53A5 293AF217 3CD50C6D 5D354FEB F78B2602 1C25C027
12E78CD4 694C9F46 9777E451 E7F8E9E0 4CD3739C 6BBFEDAE 487FB556 44E9CA74
FF77A53C B729802F 6ED4A5FF A8BA1598 90FC

$E =$ E3B5D5D0 02C1BCE5 0C2B65EF 88A188D8 3BCE7E61

With SHA-1 and PSS, convert a message (114 octets), a salt (20 octets) and a trailer ('BC') into a representative (1024 bits).

$F =$ 66E4672E 836AD121 BA244BED 6576B867 D9A447C2 8A6E66A5 B87DEE7F BC7E65AF
5057F86F AE8984D9 BA7F969A D6FE02A4 D75F7445 FEFDD85B 6D3A477C 28D24BA1
E3756F79 2DD1DCE8 CA94440E CB5279EC D3183A31 1FC896DA 1CB39311 AF37EA4A
75E24BDB FD5C1DA0 DE7CECDF 1A896F9D 8BC816D9 7CD7A2C4 3BAD546F BE8CFEBC

$S = G^s \bmod n$

$S =$ 0F624406 FC3A216B 23D44ECF F430C05A 455B8218 E22FE47B 1FEA060C 5A9CB2DE
A6981717 80B5E60C 50A567A5 58EF47B5 FE28AF9B E029611C 85A93345 9B0E610A
064F45CC C1263A10 67E5BFC0 105BBFBC 9225A460 8385A417 EB80587B 470209F9
381658A7 72739BA8 2DA018E1 4AAE564C 0A749A05 D0C1E61C 93FDE777 6D8248E6

Verification — $G^* = S^v \bmod n$. $F^* = G^*$.

$F^* =$ 66E4672E 836AD121 BA244BED 6576B867 D9A447C2 8A6E66A5 B87DEE7F BC7E65AF
5057F86F AE8984D9 BA7F969A D6FE02A4 D75F7445 FEFDD85B 6D3A477C 28D24BA1
E3756F79 2DD1DCE8 CA94440E CB5279EC D3183A31 1FC896DA 1CB39311 AF37EA4A
75E24BDB FD5C1DA0 DE7CECDF 1A896F9D 8BC816D9 7CD7A2C4 3BAD546F BE8CFEBC

$HH^* =$ DF1A896F 9D8BC816 D97CD7A2 C43BAD54 6FBE8CFE

To recover the intermediate string, a mask of 864 bits (=1024–160) is built from HH^* and exclusive-ored with the leftmost bits of F^*. The recovered salt is a string of 20 octets. The recovered trailer is 'BC'.

```
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 000001E3 B5D5D002 C1BCE50C
2B65EF88 A188D83B CE7E61DF 1A896F9D 8BC816D9 7CD7A2C4 3BAD546F BE8CFEBC
```

Then, a string of 384 bits (64 bits, all zeroes, the 160 bits of $h(M)$ and the 160 bits of E^*) is hashed into HH.

E^* = `E3B5D5D0 02C1BCE5 0C2B65EF 88A188D8 3BCE7E61`

HH = `DF1A896F 9D8BC816 D97CD7A2 C43BAD54 6FBE8CFE`

C.1.2 Message without salt

Data elements for signing/verifying — The example makes use of the same data elements as C.1.1.

Signature — The message is a string of 114 octets. The salt is empty.

M =
```
859EEF2F D78ACA00 308BDC47 1193BF55 BF9D78DB 8F8A672B 484634F3 C9C26E64
78AE1026 0FE0DD8C 082E53A5 293AF217 3CD50C6D 5D354FEB F78B2602 1C25C027
12E78CD4 694C9F46 9777E451 E7F8E9E0 4CD3739C 6BBFEDAE 487FB556 44E9CA74
FF77A53C B729802F 6ED4A5FF A8BA1598 90FC
```

With SHA-1 and PSS, convert a message (114 octets), a salt (empty) and a trailer ('BC') into a representative (1024 bits).

F =
```
2DDA5328 280470C5 AFBBF866 78F0E0C6 5B473939 BF146088 B70009A3 8A8C8E25
3BDF02F3 B3DE52E9 364CACAC 3196F828 D5CDCF83 F9529F70 DB26F641 FC112E4C
11ACC6F0 15FF3C57 74C27775 96042A36 81923E5F 7A636D16 EEA8F881 3775E1A8
FB94ED45 9292E062 0AB94764 8E5FA0D7 5B53051C C87F4ECF E350AB8E 4DADABBC
```

$S = G^s \bmod n$

S =
```
81A9AA0C A1D227C5 E6FDB537 B7C897D5 D96A6B24 B8D1EAA0 A4673B05 D6D98FF6
7045161A 28BF464F B72F884B 23AB3ED0 D27F80A9 0BBF2365 2A023B00 8E997933
D08B3914 453CDF10 28566F21 F2A88C37 2A750B0E 1E962656 9571C6AF 30359BA4
F9A10764 C69CBD2F 19461CD9 4A21337E 5B6AD86F EF65FDFE 1945802D 96FF4B51
```

Verification — $G^* = S^v \bmod n$. $F^* = G^*$.

F^* =
```
2DDA5328 280470C5 AFBBF866 78F0E0C6 5B473939 BF146088 B70009A3 8A8C8E25
3BDF02F3 B3DE52E9 364CACAC 3196F828 D5CDCF83 F9529F70 DB26F641 FC112E4C
11ACC6F0 15FF3C57 74C27775 96042A36 81923E5F 7A636D16 EEA8F881 3775E1A8
FB94ED45 9292E062 0AB94764 8E5FA0D7 5B53051C C87F4ECF E350AB8E 4DADABBC
```

HH^* = `648E5FA0 D75B5305 1CC87F4E CFE350AB 8E4DADAB`

To recover the intermediate string, a mask of 864 bits (=1024–160) is built from HH^* and exclusive-ored with the leftmost bits of F^*. The recovered salt is empty. The recovered trailer is 'BC'.

```
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000164 8E5FA0D7 5B53051C C87F4ECF E350AB8E 4DADABBC
```

Then, a string of 224 bits (64 bits, all zeroes, and the 160 bits of $h(M)$) is hashed into HH.

HH = `648E5FA0 D75B5305 1CC87F4E CFE350AB 8E4DADAB`

C.1.3 Empty message without salt

Data elements for signing/verifying — The size of each prime factor is 512 bits. The size of the modulus is 1024 bits. The verification exponent is $v = 3$ (it divides neither $p_1 - 1$ nor $p_2 - 1$).

p_1 =
```
FB961451 995C82F9 527CAAAF B3FB4254 6D00A01D 8B2BDE3D 2E7B8F7D 0C9E781E
B7FABFC8 E86E9F6D ACE3435A 9D043A99 93F3E473 D93FA888 D3577906 77A94931
```

p_2 = FF0EAFCA 70585166 A8CD8E90 36E75290 2F32B863 068016B6 A89F2EA3 418882EF
6F570122 F92D2E9B EFFF7329 1818F251 BF095D6E 208F93CD CEF4767A 568AB241

s = 0A71B48C DF4A1342 5E1BAB87 9F471638 92AEB277 A9CBC369 B1CAD109 3C93FE22
33267EC0 805A7693 F6A506D0 F9723F6B 1A6F755A ECB0B7DE 1F440102 94186936
316AAC4B F39B37BF 6105DFA0 AEA60B82 C17306F2 179F2ED4 704D5A6F BCB141C0
C9380F5A 500823CE 67E8ED81 7F8A5100 59E9541B 498C91F4 1ABE8C10 6220E72B

n = FAA8ED34 EEF1CE38 D29814B6 EEAA154D C060BB37 EB1A51E8 AB0398DD ADDFD334
CB9BE20C 087B1DDF 1F78A397 62B5F20A 7A730086 30913CD2 EE60183D E249DD16
9CA4EB3A E0420E51 13D73050 4A73A926 BEFBFF32 C89858DE 5E5B3899 FEC52521
04933163 625F2963 5AB8FAA7 AA14C4F3 C0DD2470 DEFCEB39 2429110A 0149A771

Signature — With SHA-1 and PSS, convert a message (empty), a salt (empty) and a trailer (one octet, set to 'BC') into a representative (1024 bits).

F = 7CCB5422 2079C84C 343B0AB1 6307273B 36359229 BD3DFDEC A9FE8054 AD1EF319
44758A67 3B7C70C2 FACB6FE9 12690EE2 6DF58975 585A78C2 723F0C71 50535C80
8F0868F6 CA94F36C FB079FBB 9126286D 5EECA3CA ACA12593 033A0D64 136A7A72
D605080A 6CF68B6D DA0AE6A3 5D1688A6 0AC69FD5 3E44428B FD380E94 DB9176BC

$S = G^s \bmod n$.

S = F9DD9F72 FAB4AFFC ED3B0538 C5848B27 756AC50C B2890F4C BC268D96 C5E91EE8
8E3B058F 2EF6585F EF5323CA 4E2C308C C6140CF5 F5357960 5B3BF0CC 621082EB
77F4A42D 3567355E AA151FB4 652BAFFE 58A4B310 7A064669 FD4177C8 D79F5DE5
EEC562FF A2D0F5D9 C409AEA0 D5B9F8DF 493AF2F1 8F91D828 CE32C4CC 35C13113

Verification — $G^* = S^v \bmod n$.

G^* = 7CCB5422 2079C84C 343B0AB1 6307273B 36359229 BD3DFDEC A9FE8054 AD1EF319
44758A67 3B7C70C2 FACB6FE9 12690EE2 6DF58975 585A78C2 723F0C71 50535C80
8F0868F6 CA94F36C FB079FBB 9126286D 5EECA3CA ACA12593 033A0D64 136A7A72
D605080A 6CF68B6D DA0AE6A3 5D1688A6 0AC69FD5 3E44428B FD380E94 DB9176BC

HH^* = A35D1688 A60AC69F D53E4442 8BFD380E 94DB9176

To recover the intermediate string, a mask of 856 bits (= 1024 – 160 – 8) is built from HH^* and exclusive-ored with the leftmost bits of F^*. The recovered salt is empty.

00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 000001A3 5D1688A6 0AC69FD5 3E44428B FD380E94 DB9176BC

Then, a string of 224 bits (64 bits, all zeroes, and the 160 bits of $h(\varnothing)$) is hashed into HH.

HH = A35D1688 A60AC69F D53E4442 8BFD380E 94DB9176

C.2 RW-PSS scheme

C.2.1 Message with salt

Data elements for signing/verifying — The size of each prime factor is 512 bits (one prime factor is congruent to $3 \bmod 8$ and the other one to $7 \bmod 8$). The size of the modulus is 1024 bits.

p_1 = DBB3CB4C 375C0ECD 2FD300DB 4F085472 93CA004C EDD2019C E79CA08A 15EEFB25
DD3BAF98 183B0C2F 01D7F8B4 931856F6 DD3EBA17 7D763C03 E1DCEABC D803BE33

p_2 = EEAA4A53 47999FE7 6FB78760 64BBEC66 CB409A77 39EF5A59 06613DC3 7225D41D
2BEB1F9F 5EC77A85 38767A87 BB7015D6 07FF26DE 61282753 9306BA1C FFF093A7

s = 199A6985 E9B2BFF5 A2841CCC D80FC73A 28A14266 0987EB12 3DBCAEB2 B8EE546D
2356A3A5 7D9C28ED 71E455C4 466CBE30 7787DC5A 9959B747 5A189A8F 038A4741
E4B10153 BE08C26E 4401F7AB 6E7E9609 2CAF07C0 870B13B6 4F669667 3029EC2C
77AABC39 7FA528A2 45D7073C E69CC9BD CD7BEF91 599DCA48 4000C0BD 8AB0814E

$n =$
CCD34C2F 4D95FFAD 1420E666 C07E39D1 450A1330 4C3F5891 EDE57595 C772A369
1AB51D2B ECE1476B 8F22AE22 3365F183 BC3EE2D4 CACDBA3A D0C4D478 1C523A10
EFE6203D 6F3BC226 BF9A4597 27B8F122 C482D8C8 6019F9A8 69329187 096430A6
C67CB103 742BCBC6 6906AD23 836EBABB 511D5D80 AB8CB599 74E9AAC6 2D785C45

Signature — The message is a string of 114 octets. The salt is a string of 20 octets.

$M =$
859EEF2F D78ACA00 308BDC47 1193BF55 BF9D78DB 8F8A672B 484634F3 C9C26E64
78AE1026 0FE0DD8C 082E53A5 293AF217 3CD50C6D 5D354FEB F78B2602 1C25C027
12E78CD4 694C9F46 9777E451 E7F8E9E0 4CD3739C 6BBFEDAE 487FB556 44E9CA74
FF77A53C B729802F 6ED4A5FF A8BA1598 90FC

$E =$
E3B5D5D0 02C1BCE5 0C2B65EF 88A188D8 3BCE7E61

With SHA-1 and PSS, convert a message (114 octets), a salt (20 octets) and a trailer (one octet, set to 'BC') into a representative (1024 bits).

$F =$
66E4672E 836AD121 BA244BED 6576B867 D9A447C2 8A6E66A5 B87DEE7F BC7E65AF
5057F86F AE8984D9 BA7F969A D6FE02A4 D75F7445 FEFDD85B 6D3A477C 28D24BA1
E3756F79 2DD1DCE8 CA94440E CB5279EC D3183A31 1FC896DA 1CB39311 AF37EA4A
75E24BDB FD5C1DA0 DE7CECDF 1A896F9D 8BC816D9 7CD7A2C4 3BAD546F BE8CFEBC

As $(F \mid n) = -1$, $G = F / 2$, so that $(G \mid n) = +1$. Then $S = G^{s} \bmod n$

$G =$
33723397 41B56890 DD1225F6 B2BB5C33 ECD223E1 45373352 DC3EF73F DE3F32D7
A82BFC37 D744C26C DD3FCB4D 6B7F0152 6BAFBA22 FF7EEC2D B69D23BE 146925D0
F1BAB7BC 96E8EE74 654A2207 65A93CF6 698C1D18 8FE44B6D 0E59C988 D79BF525
3AF125ED FEAE0ED0 6F3E766F 8D44B7CE C5E40B6C BE6BD162 1DD6AA37 DF467F5E

$S =$
8A505E24 FCC61832 03636262 C6AD70F5 3AC1E5CE DC714F59 ED3693B1 F2332442
FD5D2FF1 2C8DBF9B 942A6A46 C6C63C1D 09C2D316 FF605081 19B19F3E 52F6A2BD
D20A6F20 F217C9AD 0F1E496B 70529DA9 1AD7879A F912FB99 ABD387EF AD6FE54C
72FF2FCD 80069BE0 2614AA1D 7C4FE2FF AC70D936 5A81F03B C7F1D82F 733B5E12

Verification — $G^* = S^2 \bmod n$

$G^* =$
33723397 41B56890 DD1225F6 B2BB5C33 ECD223E1 45373352 DC3EF73F DE3F32D7
A82BFC37 D744C26C DD3FCB4D 6B7F0152 6BAFBA22 FF7EEC2D B69D23BE 146925D0
F1BAB7BC 96E8EE74 654A2207 65A93CF6 698C1D18 8FE44B6D 0E59C988 D79BF525
3AF125ED FEAE0ED0 6F3E766F 8D44B7CE C5E40B6C BE6BD162 1DD6AA37 DF467F5E

As G^* is congruent to $6 \bmod 8$, $F^* = 2\ G^*$.

$F^* =$
66E4672E 836AD121 BA244BED 6576B867 D9A447C2 8A6E66A5 B87DEE7F BC7E65AF
5057F86F AE8984D9 BA7F969A D6FE02A4 D75F7445 FEFDD85B 6D3A477C 28D24BA1
E3756F79 2DD1DCE8 CA94440E CB5279EC D3183A31 1FC896DA 1CB39311 AF37EA4A
75E24BDB FD5C1DA0 DE7CECDF 1A896F9D 8BC816D9 7CD7A2C4 3BAD546F BE8CFEBC

$HH^* =$ DF1A896F 9D8BC816 D97CD7A2 C43BAD54 6FBE8CFE

To recover the intermediate string, a mask of 856 bits (=1024–160–8) is built from HH^* and exclusive-ored with the leftmost bits of F^*. The recovered salt is a string of 20 octets.

00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 000001E3 B5D5D002 C1BCE50C
2B65EF88 A188D83B CE7E61DF 1A896F9D 8BC816D9 7CD7A2C4 3BAD546F BE8CFEBC

Then, a string of 384 bits (64 bits, all zeroes, the 160 bits of $h(M)$ and the 160 bits of E^*) is hashed into HH.

$E^* =$ E3B5D5D0 02C1BCE5 0C2B65EF 88A188D8 3BCE7E61

$HH =$ DF1A896F 9D8BC816 D97CD7A2 C43BAD54 6FBE8CFE

C.2.2 Message without salt

Data elements for signing/verifying — The example makes use of the same data elements as C.2.1.

Signature — The message is a string of 114 octets. The salt is empty.

$M =$

```
859EEF2F D78ACA00 308BDC47 1193BF55 BF9D78DB 8F8A672B 484634F3 C9C26E64
78AE1026 0FE0DD8C 082E53A5 293AF217 3CD50C6D 5D354FEB F78B2602 1C25C027
12E78CD4 694C9F46 9777E451 E7F8E9E0 4CD3739C 6BBFEDAE 487FB556 44E9CA74
FF77A53C B729802F 6ED4A5FF A8BA1598 90FC
```

With SHA-1 and PSS, convert a message (114 octets), a salt (empty) and a trailer (one octet, set to 'BC') into a representative (1024 bits).

$F =$

```
2DDA5328 280470C5 AFBBF866 78F0E0C6 5B473939 BF146088 B70009A3 8A8C8E25
3BDF02F3 B3DE52E9 364CACAC 3196F828 D5CDCF83 F9529F70 DB26F641 FC112E4C
11ACC6F0 15FF3C57 74C27775 96042A36 81923E5F 7A636D16 EEA8F881 3775E1A8
FB94ED45 9292E062 0AB94764 8E5FA0D7 5B53051C C87F4ECF E350AB8E 4DADABBC
```

As $(F|n) = +1$, $G = F$, so that $(G|n) = +1$. Then $S = G^s \bmod n$.

$S =$

```
A110B935 D2589D74 74ADDD01 D9397699 D34DCA6F 10FF7547 A18CA4CF 16BD845A
247EEA0E CAE8E452 F4E3942A 3D729927 35645278 E51B2C84 2499B71A 93398E1A
06F91686 B4CE2883 D4227E36 E9EDDC39 FED100BA 941F22D5 336A9237 C9CA808B
85BD195D 758F7766 51B38B29 B6566F8C A6D43A20 088DE73D 3C324E7F A3B1F3AF
```

Verification — $G^* = S^2 \bmod n$.

$G^* =$

```
9EF8F907 25918EE7 6464EE00 478D590A E9C2D9F6 8D2AF809 36E56BF2 3CE61543
DED61A38 3902F482 58D60176 01CEF95A E6711350 D17B1AC9 F59DDE36 20410BC4
DE39594D 593C85CF 4AD7CE21 91B4C6EC 42F09A68 E5B68C91 7A899905 D1EE4EFD
CAE7C3BD E198EB64 5E4D65BE F50F19E3 F5CA5863 E30D66C9 9198FF37 DFCAB089
```

As G^* is congruent to $1 \bmod 8$, $F^* = n - G^*$.

$F^* =$

```
2DDA5328 280470C5 AFBBF866 78F0E0C6 5B473939 BF146088 B70009A3 8A8C8E25
3BDF02F3 B3DE52E9 364CACAC 3196F828 D5CDCF83 F9529F70 DB26F641 FC112E4C
11ACC6F0 15FF3C57 74C27775 96042A36 81923E5F 7A636D16 EEA8F881 3775E1A8
FB94ED45 9292E062 0AB94764 8E5FA0D7 5B53051C C87F4ECF E350AB8E 4DADABBC
```

$HH^* =$

```
648E5FA0 D75B5305 1CC87F4E CFE350AB 8E4DADAB
```

To recover the intermediate string, a mask of 856 bits (=1024–160–8) is built from HH^* and exclusive-ored with the leftmost bits of F^*. The recovered salt is empty.

```
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000164 8E5FA0D7 5B53051C C87F4ECF E350AB8E 4DADABBC
```

Then, a string of 224 bits (64 bits, all zeroes, and the 160 bits of $h(M)$) is hashed into HH.

$HH =$

```
648E5FA0 D75B5305 1CC87F4E CFE350AB 8E4DADAB
```

C.2.3 Empty message with salt

Data elements for signing/verifying — The size of each prime factor is 512 bits (one prime factor is congruent to $3 \bmod 8$ and the other one to $7 \bmod 8$). The size of the modulus is 1024 bits.

$p_1 =$

```
C41DB9CC D8777062 2BEA8836 1E49AFA2 B5B6CBD0 28479585 472150A1 96C65E89
C2114580 FDE60F6B E12CA9DD A370A3EA 74D33B52 8EB791A9 0FD52818 3D8F612F
```

$p_2 =$

```
F69AD66B F97E4CCC B4A76FD3 1F43871D C71100CA F9256C3D BE98CC23 BEC06324
A2282D3C CFCAF00B 0E7492C0 1FB19CE5 0F73EEFD 1A08B0AE 6756E7DF 5670D69B
```

$s =$ 029FB5FB 55F94917 7777F3DC 7FE703F7 A3ABC251 70FDB83E 6A02DB8A 2794CECE
05C19920 85BEE677 57CCB1CC 8972089A 1D120D0C FB04C8C0 D141FE23 5A42C453
F0883D5E 73742EB5 98435B52 B393B491 F053C59C A8950D48 CA990ADF 888C6DE4
085CEB5D 6B02AEAB BCC2D543 B4C9F995 3FE16572 2F4E0846 9AD92248 D8622DEA

$n =$ BCEB2EB0 2E1C8E99 99BC9603 F8F91DA6 084EA6E7 C75BD18D D0CDBEDB 21DA29F1
9E731125 9DB0D190 B1920186 A8126B58 2D13ABA6 9958763A DA8F79F1 62C7379D
6109D2C9 4AA2E041 B383A74B BF17FFCC 145760AA 8B58BE3C 00C52BA3 BD05A9D0
BE5BA503 E6721FC4 066D37A8 9BF072C9 7BABB26C F6B29633 043DB474 6F9D2175

Signature — The message is the empty string. The salt is a string of 160 bits.

$E =$ 61DF870C 4890FE85 D6E3DD87 C3DCE372 3F91DB49

With RIPEMD-160 and PSS, convert a message (empty), a salt (20 octets) and a trailer (one octet, set to 'BC') into a representative (1024 bits).

$F =$ 73FEAF13 EB12914A 43FE6350 22BB4AB8 188A8F3A BD8D8A9E 4AD6C355 EE920359
C7F237AE 36B1212F E947F676 C68FE362 247D27D1 F298CA93 02EB21F4 A64C26CE
44471EF8 C0DFE1A5 4606F0BA 8E63E87C DACA993B FA62973B 567473B4 D38FAE73
AB228600 934A9CC1 D3263E63 2E21FD52 D2B95C5F 7023DA63 DE9509C0 1F6C7BBC

As $(F \mid n) = -1$, $G = F / 2$, so that $(G \mid n) = +1$. Then $S = G^s \bmod n$.

$G =$ 39FF5789 F58948A5 21FF31A8 115DA55C 0C45479D 5EC6C54F 256B61AA F74901AC
E3F91BD7 1B589097 F4A3FB3B 6347F1B1 123E93E8 F94C6549 817590FA 53261367
22238F7C 606FF0D2 A303785D 4731F43E 6D654C9D FD314B9D AB3A39DA 69C7D739
D5914300 49A54E60 E9931F31 9710FEA9 695CAE2F B811ED31 EF4A84E0 0FB63DDE

$S =$ B6935ACC DCABB323 D7A7125A CA86B2E6 AF7937DE 4F523629 93B07BF2 895A4677
50553ECE 92570E7F 975CDB89 D3EC9487 CA626E9B 4E7FD5A4 16ED9C7A 9E619DCF
DC05A5A9 4089E593 50C9E86B 4DD10E5B DD709150 843D755B 057C99F6 71330258
E56474B9 6A7A4848 DC1F4100 1603BBAB DBA44AE7 1A6F8211 40137572 67C97D0C

Verification — $G^* = S^2 \bmod n$.

$G^* =$ 39FF5789 F58948A5 21FF31A8 115DA55C 0C45479D 5EC6C54F 256B61AA F74901AC
E3F91BD7 1B589097 F4A3FB3B 6347F1B1 123E93E8 F94C6549 817590FA 53261367
22238F7C 606FF0D2 A303785D 4731F43E 6D654C9D FD314B9D AB3A39DA 69C7D739
D5914300 49A54E60 E9931F31 9710FEA9 695CAE2F B811ED31 EF4A84E0 0FB63DDE

As G^* is congruent to $6 \bmod 8$, $F^* = 2\ G^*$.

$F^* =$ 73FEAF13 EB12914A 43FE6350 22BB4AB8 188A8F3A BD8D8A9E 4AD6C355 EE920359
C7F237AE 36B1212F E947F676 C68FE362 247D27D1 F298CA93 02EB21F4 A64C26CE
44471EF8 C0DFE1A5 4606F0BA 8E63E87C DACA993B FA62973B 567473B4 D38FAE73
AB228600 934A9CC1 D3263E63 2E21FD52 D2B95C5F 7023DA63 DE9509C0 1F6C7BBC

$HH^* =$ 632E21FD 52D2B95C 5F7023DA 63DE9509 C01F6C7B

To recover the intermediate string, a mask of 856 bits (= 1024 – 160 – 8) is built from HH^* and exclusive-ored with the leftmost bits of F^*. The recovered salt is a string of 20 octets.

00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000161 DF870C48 90FE85D6
E3DD87C3 DCE3723F 91DB4963 2E21FD52 D2B95C5F 7023DA63 DE9509C0 1F6C7BBC

Then, a string of 384 bits (64 bits, all zeroes, the 160 bits of $h(\varnothing)$ and the 160 bits of E^*) is hashed into HH.

$E^* =$ 61DF870C 4890FE85 D6E3DD87 C3DCE372 3F91DB49

$HH =$ 632E21FD 52D2B95C 5F7023DA 63DE9509 C01F6C7B

C.3 GQ1 scheme

Data elements for signing/verifying — The size of each prime factor is 512 bits. The size of the modulus is 1024 bits. The verification exponent is $v = 2^{80} + 13$ (a prime number dividing neither $p_1 - 1$ nor $p_2 - 1$).

p_1 =
D716BEA5 9AC10B1C B5CFD57D 0204C349 52240F8E 9BDD319D 4F5ADD0C D9478B7E
AF96558F 85A74A20 B6664136 DD589F35 CFF94287 1B3298BE 40ED2C86 899186E9

p_2 =
FBB4E01A A4BF2952 CE9BEDD7 0EEB1EC2 51CD63D1 0BD4332F 3A822FC4 4065FBC6
0197A2F7 0C969BCA 54BF79C6 6D9A2907 C06794F6 EF40CABB B45079DD 9BEBA6F9

n =
D37B4534 B4B788AE 23E1E471 9A395BBF F8A98EDB DCB39923 06C513AA A95E9A33
5221998C 20CD1344 CA50C591 93B84437 FFC1E91E 5EBEF958 76158751 02A7E836
24DA4F72 CAF28D1D F4296523 46D6F203 E17C6528 8790F6F6 D9783521 6B49F593
2728A967 D6D36561 621FF38D FC185DFA 5A160962 E7C8E087 CE90897B 16EA4EA1

v =
10000 00000000 0000000D

u =
08943E6A 64EE957A 4414AD43 2353F3E5 8DC47A64 207C07B2 A43F9C89 A4C36E25
EC66D68C A67B5931 07C612A2 B13C6AED 06AC073B BC625197 DCD66B0D B0B5C608
68E87A65 CDAC3207 78EE13F0 D7A3CC06 6F49C57E 0F91B31A BDFE911B DF85465C
A917203E 53625392 7711BDAA 035ACE7C 828E69B2 33FD26A9 AB107875 9D47D3D3

The sequence of identification data is the bit string representing “Alex Ample”.

Id = 416C 6578 2041 6D70 6C65

With SHA-1 and PSS, convert the identification data into a representative (1024 bits).

G =
3E641A22 D0D0747D 4ACC7188 4D3DFF2B 2ADFDC17 03B5A74E FD8333AB 8C4377BB
2A9B48E7 07F73409 ABFBCD2D ED69F52B 16A145CE 062FE6BD 712C1952 110DFB23
16C5F3F3 21922ED3 75A4DEB8 C41FA79B CAD86B0E A0D8FF02 C9D0D591 1BFF1E87
DBCF073F 71F18C08 EB944AE8 4883A1E1 3FB1DEA1 23B5B1EF EA2A9263 5BD5D88F

$Q = G^u \bmod n$

Q =
3BED38CE BB1219BC 068774E0 E2655CDE F67FE547 BCF2D9FA 9FE167B1 E63B2F10
1A1483D3 8A8F24ED E365A3E4 4F4F10AD ECEA7B30 D042C14C 162477B8 184AE6CF
AA78441B 1FDFB0B2 23ABCD52 8B61F313 D859FCF9 C26FCAF9 E4D9DA9B A83E9D2F
DA041E8C CBF90056 C31D654B 546C1A7F 6729A8DD 8E68512F 39E3B6F0 7959CE61

Signature — $W = r^v \bmod n$

r =
487CDB00 41BEED03 23FDD3DE C8542584 FA0E6CB9 90FAD587 8DB34E9B EDDC95B6
5D22790C 108E2184 07ED7F7D 686657BA B5A28EF8 1C2E2498 5B56E37D 9934E195
A38A835C C02CEE8E BA2F56C8 7663E332 976F5A37 20DACA12 0BCD3DF0 AEF6FD78
582EBFCE E6D05E06 172A871E AB0E8F5F C22DDB60 0F541B87 CF8E1473 58374406

W =
649A17DF 13BE8088 55E154B0 E6698DEC 528A26FD 447CC267 CF040FCE C262D0EE
8B9BFCF4 C1053A4B 997755B4 8A207E83 AB16C84D 7137BC60 0FC50DDD 6E12C4FA
F0E2429C AACDDE3A 2C2D15F6 6D57E9A6 9389DAC2 D96A4D1E A34C1DD9 4E067D4C
AA8D8E7D 13F71B0B 6CFED133 8A42F6E7 94A81579 FA374E21 90B318B6 21139691

The message is a string of 57 ASCII-coded characters, i.e., 448 bits.

M = abcdbcdecdefdefgefghfghighijhijkijkljklmklmnlmnomnopnopqo
= 61626364 62636465 63646566 64656667 65666768 66676869 6768696A 68696A6B
696A6B6C 6A6B6C6D 6B6C6D6E 6C6D6E6F 6D6E6F70 6E6F7071 6F

With SHA-1 used in accordance with the first hash-variant, $H = \mathrm{h}(W \| M)$ is computed.

H = 99394F1D 15924C03 74CF5DA4 85FCB2EC F5303F7F

As $v = 2^{80} + 13$, the first part of the signature is a string of 80 bits.

R = 99394F1D 15924C03 74CF

S =
80C7274C D9F23290 3A6423D9 327156F6 9743EAEF 03E1EFED FDA8474C 97F6570D
9EF53C6C E2AE2BA6 8D01FFF9 AA820682 14BCD775 B95CC297 DDC38A63 741AB316
6B58275E 0FB728D2 6DB18A2C 3F14B621 CF3863F8 648B3149 FE896348 BE73D37E
2F06E6E2 6C84C044 984C09C6 58300B58 EC2383E3 B0A1F139 0D62B772 A69F37B5

Verification — The sequence of identification data is the bit string representing “Alex Ample”.

$Id =$ 416C 6578 2041 6D70 6C65

With SHA-1 and PSS, convert the identification data into a representative (1024 bits).

$G =$ 3E641A22 D0D0747D 4ACC7188 4D3DFF2B 2ADFDC17 03B5A74E FD8333AB 8C4377BB
2A9B48E7 07F73409 ABFBCD2D ED69F52B 16A145CE 062FE6BD 712C1952 110DFB23
16C5F3F3 21922ED3 75A4DEB8 C41FA79B CAD86B0E A0D8FF02 C9D0D591 1BFF1E87
DBCF073F 71F18C08 EB944AE8 4883A1E1 3FB1DEA1 23B5B1EF EA2A9263 5BD5D88F

$W^* = S^v \times G^R \bmod n$

$W^* =$ 649A17DF 13BE8088 55E154B0 E6698DEC 528A26FD 447CC267 CF040FCE C262D0EE
8B9BFCF4 C1053A4B 997755B4 8A207E83 AB16C84D 7137BC60 0FC50DDD 6E12C4FA
F0E2429C AACDDE3A 2C2D15F6 6D57E9A6 9389DAC2 D96A4D1E A34C1DD9 4E067D4C
AA8D8E7D 13F71B0B 6CFED133 8A42F6E7 94A81579 FA374E21 90B318B6 21139691

$H^* =$ 99394F1D 15924C03 74CF5DA4 85FCB2EC F5303F7F

$R^* =$ 9939 4F1D1592 4C0374CF

C.4 GQ2 scheme

C.4.1 First example: $b > 1$ and $m = 10$

Data elements for signing/verifying — The size of each prime factor is 512 bits. The size of the modulus is 1024 bits.

$p_1 =$ EBF36016 972BFE86 E5FA0D25 21E852A8 D8D28681 973F9439 9E06DA9B AFB5B9AA
2823FD4B 6788C807 5B9581B5 2E8343F8 AC469E00 37149F01 15404132 E99EDF91

$p_2 =$ F5ACDA1A 3C03EB5D 211AB7D1 6BDC15D8 AA624EFB 1C5CAE72 78B39C6A 86811C74
B1FE14C8 5BC9B189 7D25C467 84551316 D90C92FF B0ED7312 400E0C54 87A5DDE5

$n =$ E26F3B7F 9BB6527A 98C545CC 3AACDE35 234D51B7 199F409A 102EBA25 88C9A15D
4B8937A5 BAD6A5BF 7CE79F28 C95973F4 315B2C13 78BA6783 CCCE8CFE 1A45CEEA
0129B046 9A6820D4 637A5BF3 25E80B82 AFB6F274 10F9D46C 7057066C 40AF0383
BD14EDE6 21DB0B27 EF03596E 6111DDD5 7373B2CA DCC8E18A EE50C918 B19329B5

As $b_1 = 4$ and $b_2 = 2$, the adaptation parameter is $b = 4$. The security parameter is $k = 8$.

$u_1 =$ 03F315E6 C0CDCB85 B00F7C82 541E4C8A 35891E22 61511F72 2AE62B5E C523F1B8
9A260238 681EA921 278773A8 D164507E 449A3A9B 0EEC075D 5BA41057 632B19CC

$u_2 =$ 0AB0F9AD CC449BAA 1984CDA7 D9159FE3 61CA2F37 E587F887 7348B0FA 92C27661
040EF29F 881E92FD DFB638C0 113E43C8 AA8A1015 A88F1555 F7519C81 5DB733DC

There are ten base numbers ($m = 10$), namely, $g_1 = 2$, $g_2 = 3$, $g_3 = 5$ … $g_9 = 23$, $g_{10} = 29$.

$Q_{1,1} =$ 82BBA646 0DE18D07 5DE2E587 21B39EB8 DE519421 6D708F55 AD6F4931 5C5B0855
CBC2998E EFD22770 C86C1D1E 5D86262B 993BA8C1 3B68F1C4 470AA1EC 423AC707

$Q_{1,2} =$ BE7E88FC A3C077CE 99470064 720AFBD1 85EE2F86 BE030D41 CD7963E2 3F6E8F60
AF6E27B9 DADBA151 6CF69B16 689B9B79 B6551C33 31EB9306 EE5A6941 C3510295

$Q_{2,1} =$ B14DE96C 2535745A A34B3383 1851EE0D 3FB2BE8E F35481C4 F70D2C83 9A764413
837CB60F 95C48BB7 9CDA14EB A6BCC2A0 E0534B98 EF31EF9F 2728BD4A 53BAA0AF

$Q_{2,2} =$ 1F63D720 C208381A 5018521A 7A94C3B4 C9391194 CB89A591 811985DE 8D577EA4
FCF1006A 6565450C 765FB060 BE850F6B 6591058A 2EEB4EF6 1E037196 A1F6865B

$Q_{3,1} =$ 3CCA59F0 2BF22ECA F41715C0 EB63B927 57311919 E35C111A FD30B283 0AEF9E24
4ED0258E 9C5D2D88 A3EFBFB1 C8748ADD 806477F1 2557D27A 6E57ADEE A8C852CE

$Q_{3,2} =$ DF5A0BB8 2A12B2AD 53997661 D8B3A0DA D597A0BD 2B45E6FF E1B86C85 74F3066A
A73566CB 65C57F74 2B172459 A7827489 BE751387 8F315CF8 1AD7FC58 A4A4DFE5

$Q_{4,1} =$ 91158AEF FA55FEBE AECF276F D9901100 74F047C0 600D14FC 8214307E 5F54B5E2
B932774A 7A8AD32D A99CDA71 AEA9B497 CC25F7F7 FE4EDCA3 F1E31788 EF5DDE13

$Q_{4,2}$ = 44AB0D39 BC94C14D 84094C96 DC39A55C 5C93E34B BAB404BD D3AC7CE6 20D27F2F
3E18D74A 59947BE4 44A65B15 5C34A5D5 23BEE51D 23222E47 D7DE3853 F1C28AD7

$Q_{5,1}$ = 504A973C 7D80D257 254D57A5 23C66FF5 17BA0459 2BE2905A 4470C934 895CF339
A24DE8C7 9915773B A6D5BABD C94FA867 793709C2 DBC86441 4FC5DB1D A06B98C3

$Q_{5,2}$ = 7C280A6B 5C863BBF A7067371 468F580B 12EA4E02 C7EBBFF1 06425C64 5FA1202B
215C623C A860A064 4717409C 4DE7C025 C2C54C76 115713BA EF38A13D 3519D8FC

$Q_{6,1}$ = 9E28C724 D91C36A1 E698147F C63DAB2C 1DC614B1 2AAB9815 32B5A48F 14294A70
15A02AAF D214E899 283D0C87 FEB6C22C 04A1684E 66746A18 E15E094A 33CE2916

$Q_{6,2}$ = D1B7089C 2B0DB77D A2C30284 F838D625 1BD12E80 DCBE42CF 62C17FD1 D46B67BB
A0856BDB 242CDA05 675FE38E 1D2D2B3D 7411C4DF 5D2693F2 51150984 A9D98825

$Q_{7,1}$ = 90371C09 3331C592 54A4D9F7 CD1D87F6 392D50AD 2927DF6D B0069020 C11DE222
DA979513 EE070AF2 8DEF161E 970771A7 92EBD088 BC035804 5BD90D3B 36FB43BF

$Q_{7,2}$ = 7B073391 164F2A6D 428FAC6C 7641D332 EE990071 2D3B7736 DE04A6FB BAB717B8
031E7E02 1E87A4E0 1EF97F5C 37EA95A6 8E00C9B8 75133CFE 676F0EF4 71D27C99

$Q_{8,1}$ = 1F8AC869 116F8959 65E15081 70E4F943 C4319C3A 86B39FEA 26D51C3D B45C25DC
70DA1286 18E64E76 93E56D98 52E1774E 1A211794 4EE4749B E5EFCD58 EC8E4704

$Q_{8,2}$ = D161EE9D C0955D92 45A09B09 A6296EB4 7CE3798B E799C6BE 1BA43FC4 69837579
A8EF1710 1D706BDB 533757EC 55057767 193BF413 01240301 48EBA7A1 66F94152

$Q_{9,1}$ = 7F6D3D4E EA6D838A CA90E050 0BC11F77 B7B2A7A2 9A0B2D70 FE335817 2D5C71AA
A26A78AF 0EA3C2E9 A24F4809 A7F9D816 297F99E6 4D83F965 29E3EFC5 8AC2D425

$Q_{9,2}$ = 7092A2EF 08527BAA BE0344DC A7D5DDA7 E7B09C61 115D6041 51058F5A 151A2244
972DAFC7 796186FB 7D36416C 0ECE7B65 DA96EEDF 9C086E29 BA468733 59650F97

$Q_{10,1}$ = 4833DAC9 2EFC0A52 F44C33D2 98B4604C A12C33EE 7B122FF9 D079A745 1670096E
21E65D78 DDBBE50A 39CCB146 6D807CCF 3795D5CA 7D846115 00BCFF24 B8B02457

$Q_{10,2}$ = 57AD0549 0C3CBC49 446296D2 FBB05666 72E160A0 7FD80DBF BD2A1A5A 6D6932EA
FEAA46B1 84B3E43A 869A4AAE E8A56015 18789A7D 42273083 944B52C5 20787136

Signature — $W_i = r_i^{2^{k+b}} \bmod p_i$

r_1 = 958FE0FE 77561815 FCCE3499 D2AA78C6 701CB4DF 3EAEF982 160F9254 592C63ED
D4692A99 336020DA 4427AD2A 5845CFDD 0153CEB3 6507C76A 9473DAC1 A764E4C2

r_2 = ED1F46C6 B0143F7F A70DC68C 0E8E4324 5F22CE6C BC811A7C E90D7B0C 0D828256
C479922A C1B1CD6E 52DD82F3 75B90D0C 9EA6FD45 34611F9C 2CE4EF1E DB7DB9B7

W = 202B4E86 A41BC533 50A20AB4 BAD183E4 1362321A 6EF33B89 162CA681 C993A94D
0F009CB3 4EFEBECB FB473A02 291888C8 A73D9B90 13D814BF AEFA104D 1B551E59
DFD8A626 C74F9F85 C047D5FF E580277D 14A13B84 537B421B 5E6F8F64 64334BA9
9092041F 9EADBAF1 3EA6246B 8A1E3275 31C41AE2 904FA368 BA980C56 356E4896

The message is a string of 57 ASCII-coded characters, i.e., 456 bits.

M = abcdbcdecdefdefgefghfghighijhijkijkljklmklmnlmnomnopnopqo
= 61626364 62636465 63646566 64656667 65666768 66676869 6768696A 68696A6B
696A6B6C 6A6B6C6D 6B6C6D6E 6C6D6E6F 6D6E6F70 6E6F7071 6F

With SHA-1 in accordance with the first hash-variant, $H = \mathrm{h}(W \| M)$ is computed.

H = 6AD0F7A4 1C5F7A93 29CA5B49 AE8D7105 7010A69B

The first part of the signature is a string of 80 bits, i.e., ten octets.

R_1 = 6A, R_2 = D0, R_3 = F7, R_4 = A4, R_5 = 1C, R_6 = 5F, R_7 = 7A, R_8 = 93, R_9 = 29, R_{10} = CA

S_1 = 3E356475 F5020A7B 4DD75C90 FCBB8994 CC147A08 9A0121B4 3B59119B 5AE46177
02C3672C B745A2A0 7BD11811 39771D77 FFD9ECD6 DBFED354 58E45185 9BA85073

S_2 = 097F6189 B060BE13 DF8CB226 1A72ED8B 29EDA213 90995927 019E9304 8AFC9720
1B9B2942 FECF94D1 893E176C C6DAA4ED 247EF7CD 5D19AE7D 59BCBB54 9518A45D

S = 6EADFEAF 25CFA808 B2BCEC31 6F6CD229 95E8599C 767F6A1F BAD1B2AD 86BE12FE
0CD1D5CB 1A09DB55 147E9D70 D7A13B6D 5A2BE45C 96E12695 D83328BF E0932757
A17EBC09 D0A49E92 BE539CE8 08D4F460 C588817C DD66AAAB 1A44794D 9E789943
E1A42021 AC22F0FC 56908E7E E9D0FB8C 04A6CE88 0F10F085 D72F786B DE73EE12

Verification — $W^* = S^{2^{k+b}} \times (g_1^{2^b})^{R_1} \times \ldots (g_m^{2^b})^{R_m} \mod n$

W^* = 202B4E86 A41BC533 50A20AB4 BAD183E4 1362321A 6EF33B89 162CA681 C993A94D
0F009CB3 4EFEBECB FB473A02 291888C8 A73D9B90 13D814BF AEFA104D 1B551E59
DFD8A626 C74F9F85 C047D5FF E580277D 14A13B84 537B421B 5E6F8F64 64334BA9
9092041F 9EADBAF1 3EA6246B 8A1E3275 31C41AE2 904FA368 BA980C56 356E4896

H^* = 6AD0F7A4 1C5F7A93 29CA5B49 AE8D7105 7010A69B

R^* = 6AD0 F7A41C5F 7A9329CA

C.4.2 Second example: $b = 1$ and $m = 4$

Data elements for signing/verifying — The size of each prime factor is 512 bits. The size of the modulus is 1024 bits.

p_1 = EBF36016 972BFE86 E5FA0D25 21E852A8 D8D28681 973F9439 9E06DA9B AFB5B9AA
2823FD4B 6788C807 5B9581B5 2E8343F8 AC469E00 37149F01 15404101 12ECF827

p_2 = F5ACDA1A 3C03EB5D 211AB7D1 6BDC15D8 AA624EFB 1C5CAE72 78B39C6A 86811C74
B1FE14C8 5BC9B189 7D25C467 84551316 D90C92FF B0ED7312 400E0BA5 327E1DF3

n = E26F3B7F 9BB6527A 98C545CC 3AACDE35 234D51B7 199F409A 102EBA25 88C9A15D
4B8937A5 BAD6A5BF 7CE79F28 C95973F4 315B2C13 78BA6783 CCCE8C2C AC4BB5A4
FC439166 CAE4EE3B 4C8C9A58 CC18654A 87E1DD6E 2223DF5B D728EDA2 DB46D042
25E3DB20 0BF6F035 8ACA6C79 61D12407 A768CF6F B3824000 5B1C0A66 903DF805

As $b_1 = b_2 = 1$, the adaptation parameter is $b = 1$. The security parameter is $k = 20$. Consequently, the verification exponent is $v = 2^{21} = 2\,097\,152$ (in decimal).

u_1 = 11411739 5367474A FC81C9AB 8C9E5F19 4B79E03E 9A85D9ED 690E5EF6 F67ABFCC
13732A9C 8A55D80D FE1D7137 0A3718BF B785B28B 5EAF213D 0F5A3FB7 E786B7C1

u_2 = 650DECFE AB2D5927 8AB00315 F1142953 632009EC 9344DB6A 74781226 FF34646B
941B28FA 28D58264 27A67783 D8084107 1394A798 DB25907F 7CF19802 3C092551

There are four base numbers ($m = 4$), namely, $g_1 = 2$, $g_2 = 3$, $g_3 = 5$, $g_4 = 7$.

$Q_{1,1}$ = 6B6C99CB 3C7BA9EC E455C0D9 75D97D24 BD8EFDA7 9C42B083 ED036C00 5F60D226
A458A073 1D4706AA 30C83CC9 E5B40772 4D30B963 4A82A7C6 8C3AD268 92ECD9A7

$Q_{1,2}$ = A2DBCAB3 9DC79BE3 0CEBEB77 6711016F 3F58CFE5 7511F1BE B0FE6858 C9CAA0D9
F77AD391 DE4B2848 54FDB389 A2919770 FE1BEBD6 266B2242 0113254F 3F2BF010

$Q_{2,1}$ = 2504EFC8 FF8B668D CECBA1CD 74C7385C B21F14EC A19D4169 1F6F79B7 99B67B9A
8460DACC 08D6D751 CEB4A936 F8048D43 7A5D7F53 D18551E4 EBE20773 782039A6

$Q_{2,2}$ = 2191936A E4032F92 E3D56BD1 837A50B5 26EAF3BE 8B21D9ED 9C31D966 FCA6EFA1
70BB5E6C 48947C34 276A56C9 3C3E60EE 1BCDCA60 3BF54BA8 7E06F299 9E926F66

$Q_{3,1}$ = 25E99A95 36B0A61F 611C677C 0BE33157 4CC00CCE 52E88139 EC4D8F6A 9AD417F4
9B37668D 0793EA8E DE220ADA C671A811 27599970 73869A53 632D6050 CE6534B8

$Q_{3,2}$ = 0E96C5C7 8762E342 3F1AC822 9611409B EF5AE5D7 FA68AF22 C6A94DC4 D202DFDE
0A3BDAF2 31B4C3E2 D8B4FFDC 06FD4D18 768E7829 00550B83 E75A7A8B 648E51A3

$Q_{4,1}$ = 31437E9C BB2A3FF9 32016F4B 1A3D0C77 7AD99519 085466AB 8D4010C7 32330887
C044CD80 3DEDE7B2 60321F13 CE4E0656 8C352155 3277EC9C CAF9132D 64EC3639

$Q_{4,2}$ = 7B4187E7 C0F8D801 D9CE9A35 CD2408B3 4B83AD55 1BB1A106 4AA62448 51B1861E
99EBA585 F182E835 42BD9ABE E5E40FCA 25A4395A C89001A4 E926D644 BC7163C1

Signature — $W_i = r_i^{2^{k+b}} \mod p_i$

r_1 = 958FE0FE 77561815 FCCE3499 D2AA78C6 701CB4DF 3EAEF982 160F9254 592C63ED
D4692A99 336020DA 4427AD2A 5845CFDD 0153CEB3 6507C76A 9473DAC1 A764E4C2

$r_2 =$ `ED1F46C6 B0143F7F A70DC68C 0E8E4324 5F22CE6C BC811A7C E90D7B0C 0D828256`
`C479922A C1B1CD6E 52DD82F3 75B90D0C 9EA6FD45 34611F9C 2CE4EF1E DB7DB9B7`

$W =$ `82074289 8E8E9537 437D57D4 17184A82 06FEB795 F9CA167D 60BB7314 EB8F1360`
`5882C202 467DD2C0 F7F8D14B 87A7FB41 15D68D1C D6313C14 CA24DD84 E4F293F6`
`30AF2A90 EB122FD1 E113C184 DCB976AC FBCD0CA4 35EF6CDD E5F66F4C 06947B36`
`5E1E3B03 3D766C5B 8619B164 6470A0FA 961008A7 90CAA733 8E3119B1 C10263B8`

The message M is a string of 57 ASCII-coded characters, i.e., 456 bits.

$M =$ `abcdbcdecdefdefgefghfghighijhijkijkljklmklmnlmnomnopnopqo`
$=$ `61626364 62636465 63646566 64656667 65666768 66676869 6768696A 68696A6B`
`696A6B6C 6A6B6C6D 6B6C6D6E 6C6D6E6F 6D6E6F70 6E6F7071 6F`

With SHA-1 in accordance with the first hash-variant, $H = \mathrm{h}(C \| W \| M)$ is computed.

$H =$ `DF96299E D7FFA63E 421B021D BBF1F0DF F9EFF729`

The first part of the signature is a string of 80 bits, i.e., four strings of 20 bits.

$R =$ `DF96299E D7FFA63E 421B`, i.e., $R_1 =$ `DF962`, $R_2 =$ `99ED7`, $R_3 =$ `FFA63`, $R_4 =$ `E421B`

$S_1 =$ `7D41042D 3E007FF9 A0CFD957 FB31EDDE 5C38DB80 9031311B 87678442 CCF9B760`
`73198995 77A622C7 93E442C4 E4B00CB3 2AD7C919 8284D27D 10BBF60A 3D0C8943`

$S_2 =$ `8F21A259 C79DD004 9AE9F552 B9120E27 452B639F 5E32FD7D 1CC80F3F B63F91D2`
`B39DFE65 9B1ABEF0 B77ACB0F 24438FAE 86C2D492 0993E936 AC1F3E3B 87741ADB`

$S =$ `0C0C08F9 AAF3ACC0 13D4B871 C087B78F A971456C 0E6531DB FD40476B 9572B5D2`
`DF3080D0 032350A7 96976D07 A32323C0 CA64E943 786E5324 201A79AB FBEFD412`
`51A8C425 5C7BA61F 4583FBA4 C93E9332 90E9B89F 06FE1F6C DE65A020 E092131D`
`6CAF52E9 A7E0748D A63065FA 39E97AB7 A56C587E 1AF1E781 F1DA70BD D29CD81C`

Verification — $W^* = S^{2^{k+b}} \times (g_1^{2^b})^{R_1} \times \ldots (g_m^{2^b})^{R_m} \mod n$

$W^* =$ `82074289 8E8E9537 437D57D4 17184A82 06FEB795 F9CA167D 60BB7314 EB8F1360`
`5882C202 467DD2C0 F7F8D14B 87A7FB41 15D68D1C D6313C14 CA24DD84 E4F293F6`
`30AF2A90 EB122FD1 E113C184 DCB976AC FBCD0CA4 35EF6CDD E5F66F4C 06947B36`
`5E1E3B03 3D766C5B 8619B164 6470A0FA 961008A7 90CAA733 8E3119B1 C10263B8`

$H^* =$ `DF96299E D7FFA63E 421B021D BBF1F0DF F9EFF729`

$R^* =$ `DF96299E D7FFA63E 421B`

C.4.3 Third example: $b = m = 1$

Data elements for signing/verifying — The prime factors and the modulus are the same as in C.4.2.

As $b_1 = b_2 = 1$, the adaptation parameter is $b = 1$. The security parameter is $k = 80$. Consequently, the verification exponent is $v = 2^{81}$.

$u_1 =$ `35C918A9 00582D37 8FE446C5 6C90396A 545739DE E16BFE29 5C594FFD 32A42925`
`B36E7CE9 3C29725E 16A1E4FC 2C97313D F047890A 00C5CA74 30A8986D 14BC0B69`

$u_2 =$ `0D5D9A73 FE1370BE 6C5B7AFA 29CADD59 19B472B0 1E910CC2 5F9F7D89 8F9FBD77`
`37581681 EE2223DE 437CBDF2 9E84EE09 98C38FF9 BC02AFA6 0E3085DA 83ADD03A`

There is one base number ($m = 1$), namely, $g_1 = 2$.

$Q_{1,1} =$ `5A8F0C99 74AA0B14 D3B10DFD B6E6B744 F9EF70C3 C048C232 AD051D03 01F471BC`
`CA8B82E1 C9888090 D5EE6942 70E86782 9AD31130 CC80C82F D1F61AC6 951E38F0`

$Q_{1,2} =$ `C7FDCB32 349A74E5 51200E87 245B8C1D C9E14192 68342A85 6DC5AA9F 58CE5B30`
`89F36C45 3CDDD494 1D14E696 61CE1EEA 71FED68E C8549FC1 1772AC8D 22F9322A`

Signature — $W_i = r_i^{2^{k+b}} \mod p_i$

$r_1 =$ `958FE0FE 77561815 FCCE3499 D2AA78C6 701CB4DF 3EAEF982 160F9254 592C63ED`
`D4692A99 336020DA 4427AD2A 5845CFDD 0153CEB3 6507C76A 9473DAC1 A764E4C2`

r_2 = ED1F46C6 B0143F7F A70DC68C 0E8E4324 5F22CE6C BC811A7C E90D7B0C 0D828256
C479922A C1B1CD6E 52DD82F3 75B90D0C 9EA6FD45 34611F9C 2CE4EF1E DB7DB9B7

W = D8B7FC73 E7D63980 40BD83D2 10C218E3 54E05104 A7F5F504 C504104D 45FE0EA6
829D5CFC 4FBAA8A6 291E86FE 78C5C8DE A32D58C2 D23831C4 5A3977B1 5A3AED68
2D7FCA9E 3C6AB4D9 BA502D7C B78D9BF4 FF4E1D1A 07462D0E 1E80A010 1232C74A
57481C4C ADFCBCDB DDD467D4 84829DB8 DF3D0F29 FCAB2A33 58C8EFE2 B22E541E

The message M is a string of 57 ASCII-coded characters, i.e., 456 bits.

M = abcdbcdecdefdefgefghfghighijhijkijkljklmklmnlmnomnopnopqo
= 61626364 62636465 63646566 64656667 65666768 66676869 6768696A 68696A6B
696A6B6C 6A6B6C6D 6B6C6D6E 6C6D6E6F 6D6E6F70 6E6F7071 6F

With SHA-1 in accordance with the first hash-variant, $H = \mathrm{h}(W \| M)$ is computed.

H = 6038021F 5173AD35 D0228511 1BC06E71 BE283E8C

The first part of the signature is a string of 80 bits.

R = 6038021F 5173AD35 D022

S_1 = 3F6109BE 4C85B65E 052ECF16 466E5ED4 697EB562 22D9C28B D7AFCED9 84E2D6E0
8EA6CC1B 674AAD88 8A87E393 FF614842 2B4D222A 89FDD988 5D2B7BAA 0E2790F4

S_2 = 730249AC 01308B9B EE624AA0 B461462D 29F585F9 010BE04E D1A624F5 9B6350E7
65FE0C51 B89C7D30 8CEFE2C2 2EA84C82 C68E3228 47A33FFE 11B2EE4F D34DE163

S = DCBE96EA D4E23255 6DC10F5E 6657FC84 C76A291F 7D5EEE0B 0E3A3F21 D17BA34A
FA2EB265 A4AD8E99 94100F8A B676506C 2CBE826C C2CF5591 79FE4509 0C90BAB4
E4A29553 577D0CFD FAF8D2CB D502E501 0BDC5B29 05FA1092 6DF1C571 36CC580A
62F3FF2F 02D5AA5F 2EEED0F7 4CB5A612 E8E8CA51 5E5225E9 2B5F5BDA E6FED15E

Verification — $W^* = S^{2^{k+b}} \times (g_1^{2^b})^{R_1} \bmod n$

W^* = D8B7FC73 E7D63980 40BD83D2 10C218E3 54E05104 A7F5F504 C504104D 45FE0EA6
829D5CFC 4FBAA8A6 291E86FE 78C5C8DE A32D58C2 D23831C4 5A3977B1 5A3AED68
2D7FCA9E 3C6AB4D9 BA502D7C B78D9BF4 FF4E1D1A 07462D0E 1E80A010 1232C74A
57481C4C ADFCBCDB DDD467D4 84829DB8 DF3D0F29 FCAB2A33 58C8EFE2 B22E541E

H^* = 6038021F 5173AD35 D0228511 1BC06E71 BE283E8C

R^* = 6038021F 5173AD35 D022

C.5 GPS1 scheme

Data elements for signing/verifying — The size of each prime factor is 512 bits. The size of the modulus is 1023 bits. The private bit string consists of $|H|$ = 160 bits. The base number is $g = 2$.

p_1 = 82066EF0 3AC36CEB DD1235B2 7C903D82 08A7C794 4ED7D930 99975E59 493A9975
2833661E 0F0FBDCF 1BC3CACD EBB27840 32FB6A96 F6B1D6F5 2FEC715C 299E9667

p_2 = 9A4509B9 A3103397 135BC900 D077EB70 85D2EDA0 8D147452 BE191748 9EF37B7E
5DAE8E06 EFF8A0C9 B21DC2FE 70F5CAA0 FFB32DCE 235B4FD0 D04DB70E 87778EA9

n = 4E5AEF68 ED78571B 7CC60A03 8D4D3722 645DE5DA 543558C1 6A3B2F49 818060AE
43C942A7 E083FD27 F5D2B1CE D5F0464C E9414EBE 8794E574 728F146A B30B558D
D8FE02D0 F1A481B3 222BFD2C A788A3DD 74293CEC 04355070 7E7119F8 DED7F347
9F46153F F28460B1 E57839CD 2B082897 B00D43F4 A4434F83 A6ECC997 ABFF6BFF

Q = AC7E93A5 C6D51660 62F70D5A FD5DCCD3 51D2BDE3

G = 4B841D8D 4B0B19DE 786C477B 46EAE85D B7DA4D56 E8A55A81 20C46C9C 52AE0443
88E4FC27 6B80D5BC F22F0729 DF51A23C 7A0F2F73 C10725C0 9270425A 13484DE8
174C05B2 75742C17 60C442E3 349EB3AF 1FED5696 6C2F7045 12D07CA0 F46EE0AC
D6E6470F 15AFA97A 1D234985 866F9489 80AE8B86 969E602E 0BEAC6BA 8F97E4C0

Coupon production — Every coupon (and every first part of signature) is a string of $|H| = 160$ bits; consequently, every random bit string consists of 2 $|H| + 80 = 400$ bits.

$r =$ 3392 7D49E5BB 34CA8944 5C48DD10 10C5D82D 4691D585 E837BEF0 7A85445D
349B06B4 19FEA7F4 22AFA2DB E517B674 6D6DD15B

$W =$ 1F55C1EB A498CEE5 D1C763D5 201555A7 CDF3D50C E7C1D6F5 C24FFE90 B21D03CF
FF44D97B 47CF3088 B8204359 EB16A83E DBB05A3D DE935380 6C241FB6 8C45D26E
65730578 2538CB66 A08B5BE4 95F24A27 A47A466B A727761A 27B02AA5 BA1A2F75
368A8880 0D948066 46D8C769 518F4AB6 88A3CF23 C80BC9DB 5CB49C18 07D5CB24

$T =$ 04314E85 FCC85423 BF6752E2 52E5447E 3FE5EB54

Coupon consumption — The message is a string of 48 octets.

$M =$ 5E62999E EAA94D28 293B3646 61F39467 00FBFA7D 59439574 F4918A54 2EDAD30A
6EB0BACA 51B84D67 D264DD46 E18E1D4D

As $T = \mathrm{h}(W)$), with SHA-1 in accordance with the third hash-variant, $R = \mathrm{h}(\mathrm{h}(W) \| M)$ is computed.

$R =$ F1A2D8F3 65089130 8DBF7E5A 30ADBC27 8B3EA2F2

$S =$ 3392 7D49E5BB 34CA8943 B977F9C8 C4E1AF1B 6BC37488 2B7D4C0A 7405B2F2
15908C29 4D336B75 E4517479 913E7D30 7A12AAC5

Verification — $W^* = G^R \times g^S \bmod n$

$W^* =$ 1F55C1EB A498CEE5 D1C763D5 201555A7 CDF3D50C E7C1D6F5 C24FFE90 B21D03CF
FF44D97B 47CF3088 B8204359 EB16A83E DBB05A3D DE935380 6C241FB6 8C45D26E
65730578 2538CB66 A08B5BE4 95F24A27 A47A466B A727761A 27B02AA5 BA1A2F75
368A8880 0D948066 46D8C769 518F4AB6 88A3CF23 C80BC9DB 5CB49C18 07D5CB24

$\mathrm{h}(W^*) =$ 04314E85 FCC85423 BF6752E2 52E5447E 3FE5EB54

$R^* =$ F1A2D8F3 65089130 8DBF7E5A 30ADBC27 8B3EA2F2

C.6 GPS2 scheme

Data elements for signing/verifying — The size of each prime factor is 512 bits. The size of the modulus is 1024 bits. The verification exponent is $v = 2^{160}+7$ (a prime number dividing neither $p_1 - 1$ nor $p_2 - 1$). The base number is $g = 2$.

$p_1 =$ C64CCAD5 257D396F C5C913B6 7DA871B2 93A2F18F B96DB409 10732E70 9C5B43BB
5CD2F846 080CD347 9D82CDA5 3138D667 AD1ABB51 F0969798 19D12064 C6BA2447

$p_2 =$ C8A11B13 662D4910 E6950FD5 319C8DB0 9A569353 389ED9FE FB74291D C22ABBDF
8BE79413 030E4029 190DB076 4BCB7F6B C4CF5557 63C38E41 ECB6BFB1 2D5AFFB1

$n =$ 9B68C9BB 35939E50 00D1EE1D BB8C398B DBE8EBEC 34A2DE5F C6683C06 E5C3D726
89A0D1AE AC6E2ED8 18063C75 4AE472D9 9814D17F F466CD99 49DB846E E0342555
F5259565 66E0B02F 88C01C3A 4A67B8EC 93B7CAF1 8B556218 EF87F670 9DCC0CDC
DA91CFA7 E8290D66 04EE08DC 08C20B5C 7FB9029E FD3537D8 C9766B89 0CCBCE17

$v =$ 1 00000000 00000000 00000000 00000000 00000007

$Q =$ 32872E7E E4C3DE36 4E6055DD FF82C082 F79B044B F577CE94 A88C99AE 0012EEA3
4FE81876 2A5F4791 76F23313 6FC8A4B3 89398CD8 2FBD0833 F599145D 7E3B3598
F2E016B9 649FB68D 23518763 A2F65A73 7302EFD5 90F0BEA9 DA4047FC 49A11B72
9AF6499E 56DA3DF2 A71DC422 FEE9DF17 280BC086 FCB2BCD2 15379E6E DA40D117

Coupon production — Every coupon (and every first part of signature) is a string of $|H| = 160$ bits; consequently, every random bit string consists of $\alpha + |H| + 80 = 1024 + 160 + 80 = 1264$ bits.

$r =$ 5DB7 023CC4A7 0D37412C 5FD64999 D19D86B4 42FE83BA D7263123 D8188DED
95D04097 64FDB882 9E170B9C D251C234 EC61ACA8 7439BCD3 5C62A1A6 7993AF09
913F2386 C4B77433 AFD5C0BC 2FD53E86 57FBCAF8 5E3CC341 C93AEFE5 ED4B8CB6
B31190F8 35F84100 36B15A7E 0594C11C 37639BCB F75F8C29 08248EE6 743DE34D
FE1284C0 D7745FFE A34433BC CDA50ECA 34BA2130 A6EF18D5 663DECD8 D554F2BD