BRITISH STANDARD

BS ISO/IEC 9798-5:2009

Information technology — Security techniques — Entity authentication

Part 5: Mechanisms using zero knowledge techniques

ICS 35.040

British Standards

National foreword

This British Standard is the UK implementation of ISO/IEC 9798-5:2009. It supersedes BS ISO/IEC 9798-5:1999 which is withdrawn.

The UK participation in its preparation was entrusted to Technical Committee IST/33, IT - Security techniques.

A list of organizations represented on this committee can be obtained on request to its secretary.

This publication does not purport to include all the necessary provisions of a contract. Users are responsible for its correct application.

Compliance with a British Standard cannot confer immunity from legal obligations.

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 28 February 2010.

ISBN 978 0 580 59842 5

Amendments/corrigenda issued since publication

Date	Comments

INTERNATIONAL STANDARD

ISO/IEC 9798-5

Third edition
2009-12-15

Information technology — Security techniques — Entity authentication —

Part 5: **Mechanisms using zero-knowledge techniques**

Technologies de l'information — Techniques de sécurité — Authentification d'entité —

Partie 5: Mécanismes utilisant des techniques à divulgation nulle

Reference number
ISO/IEC 9798-5:2009(E)

Contents

Page

Foreword

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

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 9798-5 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 27, *IT Security techniques*.

This third edition cancels and replaces the second edition (ISO/IEC 9798-5:2004), which has been technically revised. This edition adds a new mechanism based on elliptic curve discrete logarithm.

ISO/IEC 9798 consists of the following parts, under the general title *Information technology — Security techniques — Entity authentication*:

— *Part 1: General*

— *Part 2: Mechanisms using symmetric encipherment algorithms*

— *Part 3: Mechanisms using digital signature techniques*

— *Part 4: Mechanisms using a cryptographic check function*

— *Part 5: Mechanisms using zero-knowledge techniques*

— *Part 6: Mechanisms using manual data transfer*

Introduction

This part of ISO/IEC 9798 specifies authentication mechanisms that involve exchanges of information between a claimant and a verifier.

In accordance with the types of calculations that need to be performed by the claimant and the verifier, the mechanisms can be classified into the following four main groups (see Annex C).

— The first group (see Clauses 4 and 5) is characterized by the performance of short modular exponentiations. The challenge size needs to be optimized since it has a proportional impact on workloads.

— The second group (see Clauses 6 and 7 and 8) is characterized by the possibility of a "coupon strategy" for the claimant. A verifier can authenticate a claimant with very limited computational power. The challenge size has no practical impact on workloads.

— The third group (see 9.2) is characterized by the possibility of a coupon strategy for the verifier. A verifier with very limited computational power can authenticate a claimant. The challenge size has no impact on workloads.

— The fourth group (see 9.3) has no possibility of a coupon strategy.

ISO and IEC draw attention to the fact that it is claimed that compliance with this part of ISO/IEC 9798 may involve the use of the following patents and their counterparts in other countries.

US 4 995 082 issued 1991-02-19, Inventor: C.P. Schnorr,

US 5 140 634 issued 1992-08-18, Inventors: L.C. Guillou and J-J. Quisquater,

EP 0 311 470 issued 1992-12-16, Inventors: L.C. Guillou and J-J. Quisquater,

EP 0 666 664 issued 1995-02-02, Inventor: M. Girault,

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 applications 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 overleaf.

RSA Security Inc. Attention General Counsel 174 Middlesex Turnpike Bedford, MA 01730, USA	US 4 995 082
France Telecom R&D 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
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
France Telecom claims that Patent Applications are pending in relation to Clauses 6 (GQ2) and 8 (GPS2). The Patent numbers will be provided when available. ISO/IEC will then request the appropriate statement.	

Information technology — Security techniques — Entity authentication —

Part 5: Mechanisms using zero-knowledge techniques

1 Scope

This part of ISO/IEC 9798 specifies entity authentication mechanisms using zero-knowledge techniques:

— mechanisms based on identities and providing unilateral authentication;

— mechanisms based on integer factorization and providing unilateral authentication;

— mechanisms based on discrete logarithms with respect to numbers that are either prime or composite, and providing unilateral authentication;

— mechanisms based on asymmetric encryption systems and providing either unilateral authentication, or mutual authentication;

— mechanisms based on discrete logarithms on elliptic curves and providing unilateral authentication.

These mechanisms are constructed using the principles of zero-knowledge techniques, but they are not necessarily zero-knowledge according to the strict definition for every choice of parameters.

2 Terms and definitions

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

2.1
accreditation exponent
secret number related to the verification exponent and used in the production of private keys

2.2
adaptation parameter
public key specific to the modulus and used in the definition of public keys in the GQ2 mechanisms

2.3
asymmetric cryptographic technique
cryptographic technique that uses two related operations: a public operation defined by a public data item, and a private operation defined by a private data item (the two operations have the property that, given the public operation, it is computationally infeasible to derive the private operation)

2.4
asymmetric encryption system
system based on asymmetric cryptographic techniques whose public operation is used for encryption and whose private operation is used for decryption

2.5
asymmetric pair
two related data items where the private data item defines a private operation and the public data item defines a public operation

2.6
challenge
procedure parameter used in conjunction with secret parameters to produce a response

2.7
claimant
entity whose identity can be authenticated, including the functions and the private data necessary to engage in authentication exchanges on behalf of a principal

2.8
coupon
pair of pre-computed numbers to be used only once; one is kept secret and the other remains secret until its use by an entity

2.9
claimant parameter
public data item, number or bit string, specific to a given claimant within the domain

2.10
decryption
reversal of a corresponding encryption

NOTE Decryption [30] and decipherment [24] are equivalent terms.

2.11
domain
collection of entities operating under a single security policy

NOTE For example, public key certificates created either by a single certification authority, or by a collection of certification authorities using the same security policy.

2.12
domain parameter
public key, or function, agreed and used by all entities within the domain

2.13
encryption
reversible operation by a cryptographic algorithm converting data into ciphertext, so as to hide the information content of the data

NOTE Encryption [30] and encipherment [24] are equivalent terms.

2.14
entity authentication
corroboration that an entity is the one claimed

[ISO/IEC 9798-1:1997, definition 3.3.11]

2.15
exchange multiplicity parameter
number of exchanges of information involved in one instance of an authentication mechanism

2.16
hash-function
function that maps strings of bits to fixed-length strings of bits, satisfying the following two properties:
— for a given output, it is computationally infeasible to find an input that maps to this output;
— it is computationally infeasible to find two distinct inputs that map to the same output

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

2.17
identification data
set of public data items (an account number, an expiry date and time, a serial number, etc.) assigned to an entity and used to identify it

2.18
mutual authentication
entity authentication that provides both entities with assurance of each other's identity

[ISO/IEC 9798-1:1997, definition 3.3.14]

2.19
number
natural number, i.e. a non-negative integer

2.20
pair multiplicity parameter
number of asymmetric pairs of numbers involved in one instance of an authentication mechanism

2.21
private key
data item of an asymmetric pair, that shall be kept secret and should only be used by a claimant in accordance with an appropriate response formula, thereby establishing its identity

2.22
procedure parameter
transient public data item used in an instance of an authentication mechanism such as a witness, challenge or response

2.23
public key
data item of an asymmetric pair, that can be made public and shall be used by every verifier for establishing the claimant's identity

2.24
random number
time variant parameter whose value is unpredictable

[ISO/IEC 9798-1:1997, definition 3.3.24]

2.25
response
procedure parameter produced by the claimant, and processed by the verifier for checking the identity of the claimant

2.26
secret parameter
number or bit string that does not appear in the public domain and is only used by a claimant, e.g. a private key

2.27
token
message consisting of data fields relevant to a particular communication and which contains information that has been produced using a cryptographic technique

2.28
unilateral authentication
entity authentication that provides one entity with assurance of the other's identity but not vice versa

[ISO/IEC 9798-1:1997, definition 3.3.33]

2.29
verification exponent
public key used as exponent by the claimant and the verifier

2.30
verifier
entity including the functions necessary for engaging in authentication exchanges on behalf of an entity requiring an entity authentication

2.31
witness
procedure parameter that provides evidence of the claimant's identity to the verifier

3 Notation, symbols and abbreviated terms

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

$(a \mid n)$	Jacobi symbol of a positive integer a with respect to an odd composite integer n

NOTE 1 By definition, the Jacobi symbol of any positive integer a with respect to any odd positive composite integer n is the product of the Legendre symbols of a with respect to each prime factor of n (repeating the Legendre symbols for the repeated prime factors). The Jacobi symbol [13][16] can be efficiently computed without knowledge of the prime factors of n.

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

NOTE 2 By definition, the Legendre symbol of any positive integer a with respect to any odd positive prime integer p is equal to $a^{(p-1)/2} \bmod p$. This means that $(a \mid p)$ is zero if a is a multiple of p, and either +1 or –1 otherwise, depending on whether or not a is a square modulo p.

$\lvert A \rvert$	bit size of the number A if A is a number (i.e. the unique integer i so that $2^{i-1} \le A < 2^i$ if $A > 0$, or 0 if $A = 0$, e.g. $\lvert 65\,537 = 2^{16}+1 \rvert = 17$), or bit length of the bit string A if A is a bit string

NOTE 3 The binary representation of a number A as a string of $\lvert A \rvert$ bits is straightforward. To represent a number A as a string of α bits with $\alpha > \lvert A \rvert$, $\alpha - \lvert A \rvert$ bits set to 0 are appended to the left of the $\lvert A \rvert$ bits.

$\lfloor A \rfloor$	the greatest integer that is less than or equal to the real number A
$B \parallel C$	bit string resulting from the concatenation of data items B and C in the order specified. In cases where the result of concatenating two or more data items is input to a cryptographic algorithm as part of an authentication mechanism, this result shall be composed so that it can be uniquely resolved into its constituent data strings, i.e. so that there is no possibility of ambiguity in interpretation. This latter property could be achieved in a variety of different ways, depending on the application. For example, it could be guaranteed by

(a) fixing the length of each of the substrings throughout the domain of use of the mechanism, or

(b) encoding the sequence of concatenated strings using a method that guarantees unique decoding, e.g. using the distinguished encoding rules defined in ISO/IEC 8825-1 [23]

CRT	Chinese remainder theorem
d	challenge (procedure parameter)
D	response (procedure parameter)
f	number of prime factors
gcd(a, b)	the greatest common divisor of the two integers a and b
G, G_i	public key (domain parameter)
$G(A)$, $G_i(A)$	public key (claimant parameter)
h	hash-function
$\lvert h \rvert$	bit length of the hash-code produced by the hash-function h
H, HH	hash-codes
$Id(A)$	identification data (claimant parameter)
$Id_i(A)$	part of the identification data (claimant parameter)
j mod n	the unique integer i from {0, 1, … n–1} such that n divides $j – i$
j mod* n	the unique integer i from {0, 1, … (n–1)/2} such that n divides either $j – i$ or $j + i$
lcm(a, b)	the least common multiple of the two integers a and b
m	pair multiplicity parameter (domain parameter)
n	composite modulus (domain parameter)
$n(A)$	composite modulus (claimant parameter)
p_1, p_2 …	prime factors of the modulus in ascending order, i.e. $p_1 < p_2 <$ … (secret parameters)
Q, Q_i	private key (secret parameter)
r	fresh random number or fresh string of random bits (secret parameter)
v	verification exponent (domain parameter)
W	witness (procedure parameter)
'$X_1X_{2...}$'	number whose hexadecimal representation is X_1X_2…, where each X_i is equal to one of 0-9 and A-F
α	modulus size in bits, i.e. $2^{\alpha-1} \leq$ modulus $< 2^{\alpha}$, also denoted $\lvert$modulus$\rvert$ (domain parameter)
δ	length of fresh strings of random bits for representing challenges (domain parameter)
ρ	length of fresh strings of random bits for representing random numbers (domain parameter)
{a, b, c, …}	set containing the elements a, b, c, …

For the purposes of Clause 5 (identity-based mechanisms), the following symbols and abbreviated terms apply.

F	bit string
t	exchange multiplicity parameter (domain parameter)
u	accreditation exponent with respect to the modulus (secret parameter)
u_j	accreditation exponent with respect to the prime factor p_j (secret parameter)

For the purposes of Clause 6 (integer factorization based mechanisms), the following symbols and abbreviated terms apply.

b	adaptation parameter (specific to the modulus)
D_j	response component with respect to the prime factor p_j (secret parameter)
g_i	basic number (domain parameter)
$g_i(A)$	basic number (claimant parameter)
k	security parameter (domain parameter)
$Q_{i,j}$	private component with respect to the basic number g_i and the prime factor p_j (secret parameter)
r_j	fresh random number with respect to the prime factor p_j (secret parameter)
u_j	accreditation exponent with respect to the prime factor p_j (secret parameter)
W_j	witness component with respect to the prime factor p_j (secret parameter)

For the purposes of Clause 7 (mechanisms based on discrete logarithms with respect to prime numbers), the following symbols and abbreviated terms apply.

g	base of the discrete logarithms (domain parameter)
p	modulus (domain parameter)
q	prime number (domain parameter)

For the purposes of Clause 8 (mechanisms based on discrete logarithms with respect to composite numbers), the following symbols and abbreviated terms apply.

g	base of the discrete logarithms (domain parameter)
$g(A)$	base of the discrete logarithms (claimant parameter)
σ	number of bits for private keys in the first mode (domain parameter)

For the purposes of Clause 9 (mechanisms based on asymmetric encryption systems), the following symbols and abbreviated terms apply.

P_A	public operation, i.e. encryption (claimant parameter)
S_A	private operation, i.e. decryption (secret parameter)

For the purposes of Clause 10 (mechanisms based on discrete logarithms on elliptic curves), the following symbols and abbreviated terms apply.

$[n]P$	multiplication operation that takes a positive integer n and a point P on the curve E as input and produces as output another point Q on the curve E, where $Q = [n]P = P + P + \ldots + P$ is the sum of n occurrences of P. The operation satisfies $[0]P = 0E$ (the point at infinity), and $[-n]P = [n](-P)$

4 Mechanisms based on identities

4.1 Security requirements for the environment

These mechanisms enable a verifier to check that a claimant knows private key(s) that are related to identification data by a verification key.

NOTE These mechanisms implement schemes due either to Fiat and Shamir [4] and denoted FS, or to Guillou and Quisquater [11] and denoted GQ1.

Within a given domain, the following requirements shall be satisfied.

1) Domain parameters shall be selected, which will govern the operation of the mechanism. They include a hash-function, e.g. one of the functions specified in ISO/IEC 10118-3 [25]. The selected parameters shall be made known in a reliable manner to all entities within the domain.

2) Every claimant shall be equipped with a modulus that is either a domain parameter or a claimant parameter. Each number used as modulus is set equal to the product of two or more distinct prime factors so that knowledge of its value shall not feasibly enable any entity to deduce its prime factors, where feasibility is defined by the context of use of the mechanism.

   — If the modulus is a domain parameter, then it is denoted n. A trusted authority has selected it and only this authority can use the corresponding prime factors. The authority guarantees the identities of every claimant within the domain.

     NOTE 1 For example, a card issuer has a modulus. A delegated entity signs identification data for issuing smart cards; it uses the issuer's prime factors. In each card, the delegated entity stores appropriate identification data and private key(s). During its life, the card uses its private key(s) in accordance with an identity-based mechanism.

   — If the modulus is a claimant parameter, then it is denoted $n(A)$. A principal has selected it and the corresponding prime factors are the principal's long-term secret. For each session, the principal creates a claimant. The claimant uses private key(s) as a short-term secret.

     NOTE 2 For example, in a local area network, an authority supervises each login operation within the domain and manages a directory where every verifier can obtain a trusted copy of a modulus for each principal.

     — During each login operation, i.e. when a computer opens a session, it uses a principal's prime factors for a "single-sign-on" of session identification data including identifier, expiry date and time, rights, etc.

     — During the session, the computer cannot use the prime factors because it does not know them any more. It uses the private key(s) in accordance with an identity-based mechanism. The private keys only last for a few hours: their utility disappears after the session.

3) Every claimant shall be provided with identification data and with one or more private keys by some means. In this context, the identification data is a string of bits, not all equal, that uniquely and meaningfully identifies the claimant in accordance with an agreed convention.

   NOTE The presence of an expiry date and time in the identification data enforces their expiry; the presence of a serial number simplifies their revocation.

4) Every verifier shall obtain a trusted copy of the correct modulus of the claimant.

   NOTE The exact means by which the verifier obtains a trusted copy of the correct modulus is beyond the scope of this part of ISO/IEC 9798. his may, for example, be achieved by the use of public-key certificates or by some other environment-dependent means.

5) Every claimant and every verifier shall have the means to produce random numbers.

4.2 Key production

4.2.1 Asymmetric key pair

A verification exponent, a pair multiplicity parameter and an exchange multiplicity parameter shall be selected. Unless otherwise specified, they are domain parameters respectively denoted v, m and t.

— Certain values of v, such as the prime numbers 2, 257, $2^{16}+1$, $2^{32}+15$, $2^{36}+2^{13}+1$ and $2^{40}+15$, have some practical advantages.

— The value of m shall be at most eight if $v = 2$ and set equal to one if v is an odd prime.

— The value of $v^{-m \times t}$ fixes a mechanism security level (see C.1.4). A value from 2^{-8} to 2^{-40} is appropriate for most applications.

A number, denoted α, fixes the modulus size in bits, i.e. $2^{\alpha-1} <$ modulus $< 2^{\alpha}$, in accordance with the context of use of the mechanism (for further details, see C.1.1). It is a domain parameter.

The authority or the principal shall keep secret two or more distinct large prime factors denoted p_1, p_2 … in ascending order, the product of which is the modulus.

- If $v = 2$ (the Rabin scheme), there shall be only two prime factors (i.e. $f = 2$), both congruent to 3 mod 4, but not congruent to each other mod 8.
- If v is an odd prime (the RSA scheme), there may be more than two prime factors. For each prime factor p_j, $p_j - 1$ shall be co-prime to v.

If α is a multiple of the number of prime factors, denoted f, then the bit size of each prime factor shall be α / f (for further details, see C.1.2). The modulus is set equal to either $p_1 \times p_2$ if $v = 2$, or $p_1 \times \ldots \times p_f$ if v is odd. In accordance with the second requirement in 5.1, the modulus is either a domain parameter denoted n, or a claimant parameter denoted $n(A)$.

With respect to each prime factor p_j, an accreditation exponent, denoted u_j, is set equal to the least positive integer so that $u_j \times v + 1$ is a multiple of either $(p_j - 1)/2$ if $v = 2$, or $p_j - 1$ if v is an odd prime.

With respect to the modulus, an accreditation exponent, denoted u, is set equal to the least positive integer so that $u \times v + 1$ is a multiple of either $\mathrm{lcm}(p_1 - 1, p_2 - 1)/2$ if $v = 2$, or $\mathrm{lcm}(p_1 - 1, \ldots\, p_f - 1)$ if v is an odd prime.

4.2.2 Asymmetric pair(s) of numbers

4.2.2.1 Case where *v* = 2

The identification data $Id(A)$ shall be converted into m parts by appending sixteen bits representing the numbers 1 to m, namely '0001', '0002', and so on, in turn to the string $Id(A)$.

$$Id_x(A) = Id(A) \,||\, \text{'000X'}$$

NOTE The mechanism below derives from the first format mechanism specified in ISO/IEC 14888-2 [27], known as PSS (PSS reads Probabilistic Signature Scheme) and due to Bellare and Rogaway [1].

For converting each part, from $Id_1(A)$ to $Id_m(A)$, into a string of α bits, denoted F_1 to F_m, the following computational steps are performed.

1) The string $Id_x(A)$ shall be hashed to obtain a hash-code denoted H_x.

$$H_x = h(Id_x(A))$$

2) A string of $(64+|h|)$ bits is constructed from left to right by concatenating 8 octets set to '00' and the hash-code H_x. This string shall be hashed to obtain a hash-code denoted HH_x.

$$HH_x = h(\text{'00000000 00000000'} \,||\, H_x)$$

3) A mask comprising a string of $(\alpha - |h| - 8)$ bits is constructed from the hash-code HH_x. The procedure makes use of two variables: a bit string of variable length, denoted *String*, and a 32-bit counter, denoted *Counter*.

   a) Set *String* to the empty string.

   b) Set *Counter* to 0.

   c) Replace *String* by *String* || $h(HH_x \,\|\, Counter)$.

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

   e) If $|h| \times Counter < \alpha - |h| - 8$, then go to step c.

   $Mask_x$ equals the leftmost $(\alpha - |h| - 8)$ bits of *String* where the leftmost bit has been forced to 0.

4) A string denoted F_x is constructed from left to right by concatenating the $(\alpha - |h| - 8)$ bits of the mask where the rightmost bit has been reversed, the $|h|$ bits of the hash-code HH_x and one octet set to 'BC'.

$$F_x = \text{Format}(Id_x(A)) = (Mask_x \oplus (000 \ldots 000 \,\|\, 1)) \,\|\, HH_x \,\|\, \text{'BC'}$$

A public key denoted $G_x(A)$ is derived from the number represented by the bit string F_x (also denoted F_x, this number is even, non-zero and less than the modulus), as follows.

— If the Jacobi symbol $(F_x \mid n)$ is +1, then $G_x(A) = F_x$.

— If the Jacobi symbol $(F_x \mid n)$ is –1, then $G_x(A) = F_x / 2$.

The authority or the principal shall provide claimant A with m private keys denoted Q_1 to Q_m. The private key denoted Q_x is set equal to the u-th modular power of the public key $G_x(A)$.

$$Q_x = G_x(A)^{\,u} \text{ (mod* either } n \text{ or } n(A))$$

NOTE 1 The CRT technique (see C.2.3) may be used for converting each public key into a private key.

— For each prime factor p_j, a component Z_j is set equal to $G_x(A)^{\,uj} \bmod p_j$.

— A CRT composition converts the set of components $\{Z_1, Z_2 \ldots\}$ into a number Z.

$$Q_x = Z \text{ (mod* either } n \text{ or } n(A))$$

NOTE 2 Each asymmetric pair of numbers verifies a relationship governed by the verification key.

$$G_x(A) \times Q_x^{\,2} \equiv 1 \text{ (mod* either } n \text{ or } n(A))$$

NOTE 3 Consequently, any number $G_x(A)$ or Q_x may be replaced by the modulus minus the number.

4.2.2.2 Case where *v* is an odd prime

NOTE The mechanism below derives from the first format mechanism specified in ISO/IEC 14888-2 [27], known as PSS (PSS reads Probabilistic Signature Scheme) and due to Bellare and Rogaway [1].

For converting the identification data $Id(A)$ into a string of α bits, denoted F, the following computational steps are performed.

1) The string $Id(A)$ shall be hashed to obtain a hash-code denoted H.

$$H = h(Id(A))$$

2) A string of $(64+|h|)$ bits is constructed from left to right by concatenating 8 octets set to '00' and the hash-code H. This string shall be hashed to obtain a hash-code denoted HH.

$$HH = h(\text{'00000000 00000000'} \,\|\, H)$$

3) A mask comprising a string of $(\alpha - |h|)$ bits is constructed from the hash-code HH. The procedure makes use of two variables: a bit string of variable length, denoted *String*, and a 32-bit counter, denoted *Counter*.

   a) Set *String* to the empty string.

   b) Set *Counter* to 0.

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

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

e) If $|h| \times Counter < \alpha - |h|$, then go to step c.

The mask equals the leftmost ($\alpha - |h|$) bits of *String* where the leftmost bit has been forced to 0.

4) A string denoted *F* is constructed from left to right by concatenating the ($\alpha - |h|$) bits of the mask where the rightmost bit has been reversed and the $|h|$ bits of the hash-code *HH*.

$$F = \text{Format}(Id(A)) = (Mask \oplus (000 \ldots 000 \,||\, 1)) \,||\, HH$$

A public key, denoted *G*(*A*), is set equal to the number represented by the bit string *F* (also denoted *F*, this number is non-zero and less than the modulus).

$$G(A) = F$$

The authority or the principal shall provide claimant *A* with a private key, denoted *Q*, set equal to the *u*-th modular power of the public key *G*(*A*).

$$Q = G(A)^{u} \text{ (mod either } n \text{ or } n(A))$$

NOTE 1 The CRT technique (see C.2.3) may be used for converting the public key into the private key.

— For each prime factor p_j, a component Q_j is set equal to $G(A)^{uj}$ mod p_j.

— A CRT composition converts the set of components $\{Q_1, Q_2 \ldots\}$ into the number *Q*.

NOTE 2 The asymmetric pair of numbers (the private key is the modular inverse of the RSA signature, see ISO/IEC 14888-2 [27]) verifies a relationship governed by the verification key.

$$G(A) \times Q^{v} \equiv 1 \text{ (mod either } n \text{ or } n(A))$$

4.3 Unilateral authentication exchange

The bracketed numbers in Figure 1 correspond to the steps of the mechanism, including the exchanges of information, described in detail below. The claimant is denoted *A*. The verifier is denoted *B*.

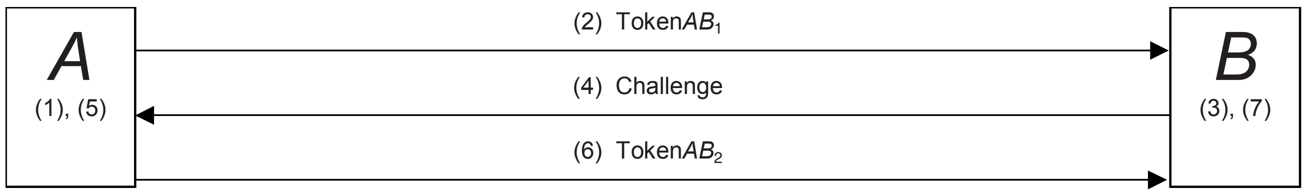

Figure 1 — Identity-based mechanism

In addition to identification data *Id*(*A*), a verification exponent *v* (a prime number), a pair multiplicity parameter *m* and an exchange multiplicity parameter *t*, the claimant shall store a modulus *n* or *n*(*A*) and either

- *m* private keys Q_1 to Q_m if $v = 2$, or
- a single private key *Q* if *v* is an odd prime.

In addition to identification data *Id*(*A*), a verification exponent *v* (a prime number), a pair multiplicity parameter *m* and an exchange multiplicity parameter *t*, the verifier shall be provided with a trusted copy of a modulus *n* or *n*(*A*). If not already known by *B*, a copy of *Id*(*A*), *v*, *m* and *t* shall be sent along with TokenAB_1; however, such a copy needs not be trusted.

For each application of the mechanism, the following procedure shall be performed *t* times. The verifier *B* shall only accept the claimant *A* as valid if all *t* iterations of the procedure complete successfully.

1) For each iteration of the procedure, a fresh number shall be uniformly selected at random, so that it is non-zero and less than the modulus. Denoted *r*, it shall be kept secret.

The fresh random number r shall be converted into a witness, denoted W, as the v-th modular power.

- Witness formula if $v = 2$: $W = r^2$ (mod* either n or $n(A)$)
- Witness formula if v is an odd prime: $W = r^v$ (mod either n or $n(A)$)

The number W is represented by a string of α bits, also denoted W.

2) A sends TokenAB_1 to B. TokenAB_1 is either witness W or a hash-code of W and *Text*, one of the following four hash variants.

The four hash variants are $h(W \,||\, Text)$, $h(W \,||\, h(Text))$, $h(h(W) \,||\, Text)$, and $h(h(W) \,||\, h(Text))$, where h is a hash-function and *Text* is an optional text field (it may be empty). If the text field is non-empty, then B shall have the means to recover the value of *Text*; this may require A to send all or part of the text field at this point. The text field is available for use in applications outside the scope of this part of ISO/IEC 9798. Annex A of ISO/IEC 9798-1 [24] gives information on the use of text fields. The hash variant is a domain parameter.

3) On receipt of TokenAB_1, the following computational steps are performed.

a) If the value of $v^{m \times t}$ is less than 2^{40} and/or if $m > 8$ when $v = 2$, and/or if $m > 1$ when v is an odd prime, then the procedure fails.

b) If the identification data $Id(A)$ is invalid (e.g. expired or revoked), then the procedure fails.

c) A fresh string of δ bits shall be uniformly selected at random.

- If $v = 2$, then $\delta = m$ and the string consists of m bits, denoted d_1 to d_m.
- If v is an odd prime, then $\delta = |v| - 1$ and the string represents a number less than v, possibly zero, denoted d.

NOTE The total number of possible challenges per iteration of the procedure should be limited to 2^{40}. If this recommendation is not followed, then special care should be taken to prevent the verifier using the claimant as a signing oracle.

4) B sends the fresh string as a challenge to A.

NOTE Optimizations may induce constraints on the Hamming weight of the challenges, with an impact on the total number of possible challenges and on the mechanism security level.

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

a) If the challenge is not a string of δ bits, then the procedure fails.

b) A response denoted D shall be computed from the random number r and

- the m private keys $Q_1, Q_2, \ldots Q_m$ and the m challenge bits $d_1, d_2, \ldots d_m$ if $v = 2$.

  Response formula if $v = 2$: $D = r \times \prod_{i=1}^{m} Q_i^{d_i}$ (mod* either n or $n(A)$)

- the single private key Q and the challenge number d if v is an odd prime.

  Response formula if v is an odd prime: $D = r \times Q^d$ (mod either n or $n(A)$)

6) A sends TokenAB_2 to B. TokenAB_2 is the response D computed from step 5)b).

7) On receipt of TokenAB_2, the following computational steps are performed.

a) If the response D is **zero** or equal to or more than the modulus, then the procedure fails.

b) The identification data $Id(A)$ shall be converted into

- m public keys (see 5.2.2.1), denoted $G_1(A), G_2(A), \ldots G_m(A)$, if $v = 2$.
- a single public key (see 5.2.2.2), denoted $G(A)$, if v is an odd prime.

c) Denoted W^*, a witness shall be computed.

- Verification formula if $v = 2$: $W^* = D^2 \times \prod_{i=1}^{m} G_i(A)^{d_i}$ (mod* either n or $n(A)$)
- Verification formula if v is an odd prime: $W^* = D^v \times G(A)^d$ (mod either n or $n(A)$)

d) If either witness W^* or a hash-code of W^* and *Text*, one of the four hash variants, is identical to TokenAB_1 received in step (2), then the iteration of the procedure is successful. Otherwise the procedure fails.

NOTE 1 Other information may be sent with any exchange of the procedure. *B* might use such information to help compute the value of the optional text field.

NOTE 2 *B* can compute the public key(s) for *A* at any stage, i.e. *B* need not wait until the receipt of response *D* before computing them. If *B* verifies *A* frequently, then *B* may cache the public key(s).

NOTE 3 The *t* iterations of the procedure can be performed in parallel, i.e. in the first step, *A* may choose *t* random numbers r_1, r_2, … r_t, compute *t* witnesses W_1, W_2, … W_t, send them simultaneously to *B*, and so on. If this parallel implementation is adopted, the total number of message exchanges will be equal to three, regardless of the value of *t*.

NOTE 4 The use of a hash-code instead of witness *W* in the first exchange of the procedure can achieve efficiency gains by reducing the number of bits in TokenAB_1.

5 Mechanisms based on integer factorization

5.1 Security requirements for the environment

These mechanisms enable a verifier to check that a claimant knows a decomposition of a claimed modulus.

NOTE These mechanisms implement schemes due to Guillou and Quisquater [12] and denoted GQ2.

Within a given domain, the following requirements shall be satisfied.

1) Domain parameters shall be selected, which will govern the operation of the mechanism. The selected parameters shall be made known in a reliable manner to all entities within the domain.

2) Every claimant shall be equipped with distinct prime factors so that knowledge of their product, i.e. the modulus (a claimant parameter), shall not feasibly enable any entity to deduce them, where feasibility is defined by the context of use of the mechanism.

   NOTE When opening a session (see 5.1), a computer may randomly select two prime factors to be used during the session (a few hours). Using the principal's long-term secret in a "single-sign-on" of session identification data, the computer signs an "ephemeral" certificate covering an "ephemeral" modulus, product of the "ephemeral" prime factors.

3) Every verifier shall obtain a trusted copy of the modulus specific to the claimant.

   NOTE The exact means by which the verifier obtains a trusted copy of the modulus specific to the claimant is beyond the scope of this part of ISO/IEC 9798. This may, for example, be achieved by the use of public-key certificates or by some other environment-dependent means.

4) Every claimant and every verifier shall have the means to produce random numbers.

5) If the mechanism makes use of a hash-function, then all entities within the domain shall agree on a hash-function, e.g. one of the functions specified in ISO/IEC 10118-3 [25].

5.2 Key production

A number, denoted α, fixes the modulus size in bits, i.e. $2^{\alpha-1}$ < modulus < 2^{α}, in accordance with the context of use of the mechanism (for further details, see C.1.1). It is a domain parameter.

A security parameter and a pair multiplicity parameter, denoted *k* and *m*, together fix a mechanism security level set to the value of $2^{-k \times m}$ in accordance with the needs of the application (see C.1.4). They are domain parameters. A value of $k \times m$ from 8 to 40 is appropriate for most applications.

NOTE 1 The total number of possible challenges should be limited to 2^{40}. If this recommendation is not followed, then special care should be taken to prevent the verifier using the claimant as a signing oracle.

Claimant *A* shall keep secret two or more distinct large prime factors denoted p_1, p_2 … in ascending order. If α is a multiple of the number of prime factors, denoted *f,* then the bit size of each prime factor shall be α / f (for further details, see C.1.2).

Each prime factor p_j determines a number, denoted b_j, so that $p_j - 1$ is divisible by 2^{bj}, but not by 2^{bj+1}, i.e. the b_j+1 least significant bits of $p_j - 1$ are one bit set to 1 followed by b_j bits set to 0 and $(p_j - 1)/2^{bj}$ is an odd number.

NOTE 2 The number b_j is set equal to one if $p_j \equiv 3 \bmod 4$, and to two or more if $p_j \equiv 1 \bmod 4$.

For the equivalence with a decomposition of the modulus, the first 54 prime numbers, namely {2, 3, 5, 7, 11, … 251}, i.e. of bit size equal to eight or less, are searched for an appropriate number *g*.

- The Legendre symbol of a candidate number *g* is evaluated with respect to each prime factor from p_1 to p_f. The candidate number *g* is appropriate if there are two prime factors p_j and p_i as follows.
  - If $b_j = b_i$, the Legendre symbols are different, i.e. $(g \mid p_j) = -(g \mid p_i)$.
  - If $b_j > b_i$, the Legendre symbol with respect to p_j is –1, i.e. $(g \mid p_j) = -1$.

NOTE 3 In average, each candidate number has one chance out of 2^{f-1} of being appropriate. Consequently the probability is negligible of not finding an appropriate number *g* within the first 54 prime numbers.

The *m* basic numbers are the number *g*, completed by as many numbers as needed from the first 54 prime numbers. They are either domain parameters, denoted g_1 to g_m in ascending order if they are the first *m* prime numbers, or claimant parameters, denoted $g_1(A)$ to $g_m(A)$ in ascending order otherwise.

NOTE 4 If the *m* basic numbers are systematically the first *m* prime numbers without checking the Legendre symbols, then for *f* large prime factors randomly generated, the probability that the knowledge of the set of the private keys does not imply the knowledge of a decomposition of the modulus is in average less than $2^{-m \times (f-1)}$.

An adaptation parameter denoted *b* is set equal to max(b_1 to b_f). It is a claimant parameter. For each basic number g_i or $g_i(A)$, a public key denoted G_i is set equal to the *b*-th square of the basic number.

$$G_i = \text{ Either } g_i^{2^b} \text{ or } g_i(A)^{2^b}$$

A verification exponent denoted *v* is set equal to 2^{k+b}. With respect to each prime factor p_j, an accreditation exponent, denoted u_j, is set equal to the least positive integer so that $v \times u_j + 1$ is a multiple of $(p_j - 1)/2^{bj}$.

For each basic number g_i or $g_i(A)$ and each prime factor p_j, a private component denoted $Q_{i,j}$ is set equal to the u_j -th modular power of the public key G_i.

$$Q_{i,j} = G_i^{u_j} \bmod p_j$$

The modulus is set equal to the product of the large prime factors, i.e. $p_1 \times ... \times p_f$. It is a claimant parameter denoted *n*(*A*).

NOTE 5 The same modulus may be used for the GQ2 mechanisms and for the RSA mechanisms.

5.3 Unilateral authentication exchange

The bracketed numbers in Figure 2 correspond to the steps of the mechanism, including the exchanges of information, described in detail below. The claimant is denoted *A*. The verifier is denoted *B*.

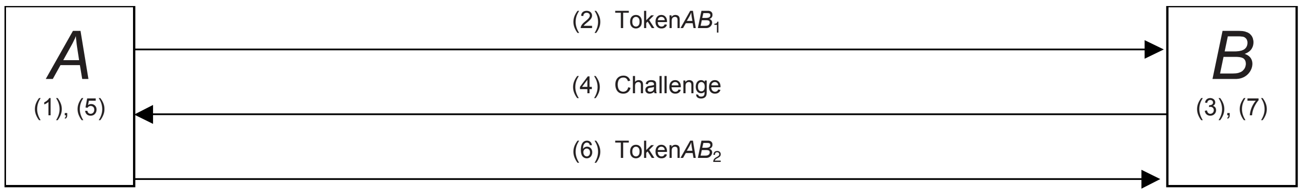

Figure 2 — Mechanism based on the factorization of a modulus

In addition to parameters b, k and m, and m basic numbers g_1 to g_m or $g_1(A)$ to $g_m(A)$, the claimant shall store either

— a modulus n(A) and m private keys Q_1 to Q_m, or

— f prime factors p_1 to p_f, $f \times m$ private components $Q_{1,1}$ to $Q_{m,f}$ and (f–1) CRT coefficients (see C.2.3).

In addition to parameters b, k and m, and m basic numbers g_1 to g_m or $g_1(A)$ to $g_m(A)$, the verifier shall be provided with a trusted copy of the claimant's modulus $n(A)$. If not already known by B, a copy of b, k, m and $g_1(A)$ to $g_m(A)$ shall be sent along with TokenAB_1; however, such a copy needs not be trusted.

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

1) For each iteration of the procedure, for each prime factor p_j, a fresh number shall be uniformly selected at random, non-zero and less than p_j. Denoted r_j, it shall be kept secret.

   Each fresh random r_j number shall be converted into a witness component, denoted W_j.

   Witness component formula: $W_j = r_j^{\,v} \bmod p_j$

   Involving the set of prime factors and CRT coefficient(s), a CRT composition (see C.2.3) shall convert the set of witness components {W_1, W_2, …} into a witness denoted W. The number W is represented by a string of α bits, also denoted W.

2) A sends TokenAB_1 to B. TokenAB_1 is either witness W or a hash-code of W and *Text*, one of the following four hash variants.

   The four hash variants are $h(W \,||\, Text)$, $h(W \,||\, h(Text))$, $h(h(W) \,||\, Text)$, and $h(h(W) \,||\, h(Text))$, where h is a hash-function and *Text* is an optional text field (it may be empty). If the text field is non-empty, then B shall have the means to recover the value of *Text*; this may require A to send all or part of the text field at this point. The text field is available for use in applications outside the scope of this part of ISO/IEC 9798. Annex A of ISO/IEC 9798-1 [24] gives information on the use of text fields. The hash variant is a domain parameter.

3) On receipt of TokenAB_1, the following computational steps are performed.

   a) If the product $k \times m$ is more than 40, then the procedure fails.

   b) If the basic numbers are not distinct prime numbers less than 256, then the procedure fails.

   c) A fresh string of $k \times m$ bits shall be uniformly selected at random and denoted $d_{1,1}$ to $d_{m,k}$.

4) B sends the fresh string as a challenge to A.

   NOTE Optimizations may limit the Hamming weight of the challenges, with an impact on the total number of possible challenges and on the mechanism security level.

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

   a) If the challenge is not a string of $k \times m$ bits, then the procedure fails.

   b) For each prime factor p_j, a component D_j shall be computed from the challenge denoted $d_{1,1}$ to $d_{m,k}$, the m private components $Q_{1,j}$ to $Q_{m,j}$ and the random number r_j.

      Starting from a number set equal to one, k sequences of zero to m modular multiplications are interleaved with k–1 modular squares. The ii-th sequence is as follows: for i from 1 to m, the bit $d_{i,ii}$ indicates whether the current number shall be modularly multiplied by the private component $Q_{i,j}$ (bit set to 1) or not (bit set to 0). A last modular multiplication by the random number r_j produces a final number, namely a response component denoted D_j.

      Consequently, considering that, from bit $d_{i,1}$ as the most significant bit up to bit $d_{i,k}$ as the least significant bit, each string of k bits represents a number less than 2^k, possibly zero, denoted d_i, the response component formula reads as follows.

      $$D_j = r_j \times \prod_{i=1}^{m} Q_{i,j}^{d_i} \bmod p_j$$

      Involving the set of prime factors and the CRT coefficient(s), a CRT composition (see C.2.3) shall convert the set of response components {D_1, D_2, …} into a response denoted D.

6) A sends TokenAB_2 to B. TokenAB_2 is the response D computed from step 5)b).

7) On receipt of TokenAB_2, the following computational steps are performed.

   a) If the response D is **zero** or equal to or more than $n(A)$, then the procedure fails.

   b) The response D shall be converted into a witness denoted W^*.

      Starting from a number set equal to D, $(b + k)$ modular squares are interleaved with k elementary operations. The *ii*-th elementary operation occurs between the *ii*-th and the $(ii +1)$-th modular squares. The *ii*-th elementary operation is as follows: for i from 1 to m, the bit $d_{i,ii}$ states whether the current number shall be modularly multiplied by the basic number g_i (bit set to 1) or not (bit set to 0).

      Consequently, considering that, from bit $d_{i,1}$ as the most significant bit up to bit $d_{i,k}$ as the least significant bit, each string of k bits represents a number less than 2^k, possibly zero, denoted d_i, the verification formula reads as follows.

$$W^* = D^v \times \prod_{i=1}^{m} G_i^{d_i} \mod n(A)$$

   c) If either witness W^* or a hash-code of W^* and *Text*, one of the four hash variants, is identical to TokenAB_1 received in step (2), then the procedure is successful. Otherwise the procedure fails.

NOTE 1 Other information may be sent with any exchange of the procedure. B may use such information to help compute the value of the optional text field. For example, A may send information such as certificates with TokenAB_1.

NOTE 2 For computing the witness and the response, the CRT technique (see C.2.3) is optional.

NOTE 3 The use of a hash-code instead of witness W in the first exchange of the procedure can achieve efficiency gains by reducing the number of bits in TokenAB_1. Moreover, this deters fault inductions when using the CRT technique in portable devices, e.g. in smart cards.

6 Mechanisms based on discrete logarithms with respect to prime numbers

6.1 Security requirements for the environment

These mechanisms enable a verifier to check that a claimant knows the discrete logarithm of a claimed public key with respect to a prime number.

NOTE These mechanisms implement schemes due to Schnorr [21] and denoted SC.

Within a given domain, the following requirements shall be satisfied.

1) Domain parameters shall be selected, which will govern the operation of the mechanism. The selected parameters shall be made known in a reliable manner to all entities within the domain.

2) The number used as the base of discrete logarithms shall be so that, for any arbitrary number j, non-zero and less than the modulus, finding a number k (if one exists), so that the k-th modular power of the base is j, shall be computationally infeasible, where feasibility is defined by the context of use of the mechanism.

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

4) Every verifier shall obtain a trusted copy of the public key specific to the claimant.

   NOTE The exact means by which the verifier obtains a trusted copy of the public key specific to the claimant is beyond the scope of this part of ISO/IEC 9798. This may, for example, be achieved by the use of public-key certificates or by some other environment-dependent means.

5) Every claimant and every verifier shall have the means to produce random numbers.

6) If the mechanism makes use of a hash-function, then all entities within the domain shall agree on a hash-function, e.g. one of the functions specified in ISO/IEC 10118-3 [25].

6.2 Key production

Three numbers, denoted p, q and g, shall be selected in accordance with the context of use of the mechanism.

— The modulus p shall be a prime number. The bit size of the number p is denoted $|p|$.

— The number q shall be a prime factor of p–1. Unless otherwise specified, the bit size of the number q is 160, i.e. $|q| = 160$.

— The base of the discrete logarithms, denoted g, shall be of order q modulo p, i.e. a number greater than 1 so that $g^q \bmod p = 1$. The base g is conveniently represented as a string of $|p|$ bits.

NOTE 1 The prime number p can be selected so that a copy of the binary representation of q is embedded within the binary representation of p. Such an approach for choosing p and q may be useful in situations where storage space and/or communications bandwidth is at a premium. See an example in D.5.1.

NOTE 2 If there is an odd factor less than q dividing p–1, then the private key may be compromised by an attack of the type described by Lim and Lee [15]. To prevent such an attack, p and q should be selected so that $(p–1)/(2\times q)$ has no prime factor less than q. Ideally, $(p–1)/(2\times q)$ should be prime.

Each claimant A shall be provided with a fresh number uniformly selected at random, non-zero and less than q, representing a private key denoted Q. It is represented by a string of $|q|$ bits.

Denoted $G(A)$, the public key for claimant A is set equal to the Q-th modular power of the base g. It is represented by a string of $|p|$ bits.

$$G(A) = g^{Q} \bmod p$$

A number, denoted δ, fixes the number of bits for representing challenges. A value of δ from 8 to 40 is appropriate for most applications. Unless otherwise specified, the value of δ is set equal to 40.

NOTE The total number of possible challenges should be limited to 2^{40}. If this recommendation is not followed, then special care should be taken to prevent the verifier using the claimant as a signing oracle.

6.3 Unilateral authentication exchange

The bracketed numbers in Figure 3 correspond to the steps of the mechanism, including the exchanges of information, described in detail below. The claimant is denoted A. The verifier is denoted B.

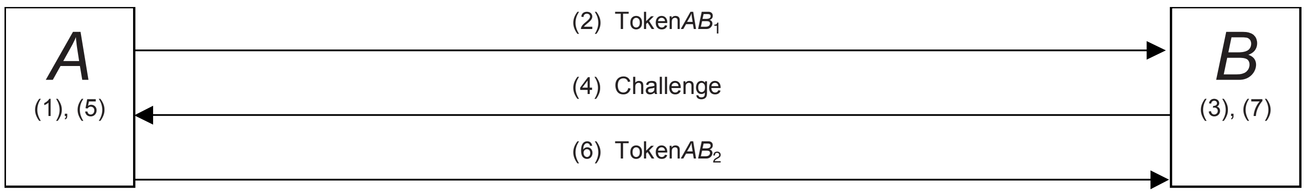

Figure 3 — Mechanism using a discrete logarithm with respect to a prime number

In addition to prime numbers p and q, a number δ and a base g, the claimant shall store a private key Q.

In the case of a coupon strategy, the claimant shall store a private key Q, a number δ and a set of coupons. To be used only once, each coupon consists of a $|q|$-bit number (that needs not be stored if it can be reproduced by a pseudo-random function) and an α-bit witness (or preferably, its hash-code).

In addition to prime numbers p and q, a number δ and a base g, the verifier shall be provided with a trusted copy of a claimed public key $G(A)$.

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

1) For each authentication, a fresh number shall be uniformly selected at random, non-zero and less than q. Denoted r, it shall be kept secret. The fresh random number r shall be converted into a witness, denoted W. The number W is represented by a string of α bits, also denoted W.

   Witness formula: $W = g^{r} \bmod p$

2) A sends TokenAB_1 to B. TokenAB_1 is either witness W or a hash-code of W and *Text*, one of the following four hash variants.

The four hash variants are $h(W \| Text)$, $h(W \| h(Text))$, $h(h(W) \| Text)$, and $h(h(W) \| h(Text))$, where h is a hash-function and *Text* is an optional text field (it may be empty). If the text field is non-empty, then *B* shall have the means to recover the value of *Text*; this may require *A* to send all or part of the text field at this point. The text field is available for use in applications outside the scope of this part of ISO/IEC 9798. Annex A of ISO/IEC 9798-1 [24] gives information on the use of text fields. The hash variant is a domain parameter.

3) On receipt of TokenAB_1, a fresh string of δ bits shall be uniformly selected at random.

4) *B* sends the fresh string as a challenge to *A*. The fresh string represents a number denoted *d*.

5) On receipt of the challenge, the following computational steps are performed.
   a) If the challenge is not a string of δ bits, then the procedure fails.
   b) A response *D* shall be computed from the random number *r* and the private key *Q*.

      Response formula: $D = r - d \times Q \bmod q$

6) *A* sends TokenAB_2 to *B*. TokenAB_2 is the response *D* computed from step 5)b).

7) On receipt of TokenAB_2, the following computational steps are performed.
   a) If the response *D* is **zero** or equal to or more than *q*, then the procedure fails.
   b) Denoted W^*, a witness shall be computed using the public key $G(A)$.

      Verification formula: $W^* = G(A)^d \times g^D \bmod p$
   c) If either witness W^* or a hash-code of W^* and *Text*, one of the four hash variants, is identical to TokenAB_1 received in step (2), then the procedure is successful. Otherwise the procedure fails.

NOTE 1 Other information may be sent with any exchange of the procedure. *B* may use such information to help compute the value of the optional text field. For example, *A* may send information such as certificates with TokenAB_1.

NOTE 2 The use of a hash-code instead of witness *W* in TokenAB_1 can achieve efficiency gains by reducing the number of bits in TokenAB_1.

7 Mechanisms based on discrete logarithms with respect to composite numbers

7.1 Security requirements for the environment

These mechanisms enable a verifier to check that a claimant knows the discrete logarithm of a public key with respect to a composite number. The public key and / or the composite number are claimed.

NOTE These mechanisms implement schemes due to Girault, Poupard and Stern [5][19] for GPS1, and to Girault and Paillès [8] for GPS2.

Within a given domain, the following requirements shall be satisfied.

1) Domain parameters shall be selected, which will govern the operation of the mechanism. These domain parameters include one of the two modes of use specified hereafter. The selected parameters shall be made known in a reliable manner to all entities within the domain.

2) Every claimant shall be equipped with a modulus that is either a domain parameter or a claimant parameter. Each number used as modulus shall be so that knowledge of its value shall not feasibly enable any entity to deduce its prime factors, where feasibility is defined by the context of use of the mechanism.

3) Each number used as the base of discrete logarithms shall be so that, for any arbitrary number *j*, non-zero and less than the modulus, finding a number *k* (if one exists), so that the *k*-th modular power of the base is *j*, shall be computationally infeasible, where feasibility is defined by the context of use of the mechanism.

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

5) Every verifier shall obtain a trusted copy of the public key(s) specific to the claimant.

NOTE The exact means by which the verifier obtains a trusted copy of the public key(s) specific to the claimant is beyond the scope of this part of ISO/IEC 9798. This may, for example, be achieved by the use of public-key certificates or by some other environment-dependent means.

6) Every claimant and every verifier shall have the means to produce fresh strings of random bits.

7) If the mechanism makes use of a hash-function, then all entities within the domain shall agree on a hash-function, e.g. one of the functions specified in ISO/IEC 10118-3 [25].

7.2 Key production

7.2.1 General

A number, denoted α, fixes the modulus size in bits, i.e. $2^{\alpha-1}$ < modulus < 2^{α}, in accordance with the context of use of the mechanism (for further details, see C.1.1). It is a domain parameter.

A number, denoted δ, fixes the number of bits for representing challenges. A value from 8 to 40 is appropriate for most applications. Unless otherwise specified, the value of δ is set equal to 40. It is a domain parameter.

NOTE The total number of possible challenges should be limited to 2^{40}. If this recommendation is not followed, then special care should be taken to prevent the verifier using the claimant as a signing oracle.

Within the domain, a mode of use shall be selected from the two modes specified hereafter.

7.2.2 First mode of use (GPS1)

A number, denoted σ, fixes the number of bits for representing private keys. Unless otherwise specified, the value of σ is set equal to 160. It is a domain parameter.

For claimant *A*, a fresh string of σ bits shall be uniformly selected at random. The string represents the private key, denoted Q.

Denoted *g*, the base of the discrete logarithms is a domain parameter. The value *g* = 2 has some practical advantages.

The modulus is either a domain parameter denoted *n*, or a claimant parameter denoted *n*(*A*). In both cases, the factorization of the modulus, i.e. the large prime factors (for further details, see C.1.2), may be unknown.

Denoted *G*(*A*), the public key for claimant *A* is set equal to the *Q*-th modular power of the base *g*. It is represented by a string of α bits.

$$G(A) = g^{Q} \text{ (mod either } n \text{ or } n(A))$$

7.2.3 Second mode of use (GPS2)

Denoted *v*, the verification exponent is a domain parameter. It shall be prime and greater than 2^{δ}. As the value of δ is set equal to 40, unless otherwise specified, the value of *v* is set equal to $2^{40}+15$ (a prime number).

Claimant *A* shall keep secret two or more distinct large prime factors, denoted p_1, p_2 … in ascending order. If α is a multiple of the number of prime factors, denoted *f*, then the bit size of each prime factor shall be α / f (for further details, see C.1.2). For each prime factor p_j, p_j–1 shall be co-prime to *v*.

The modulus is set equal to the product of the prime factors, i.e. $p_1 \times ... \times p_f$. It is a claimant parameter denoted *n*(*A*).

NOTE 1 The verification exponent *v* and the modulus *n*(*A*) together form a public RSA key.

Denoted *Q*, the private key for claimant *A* is the least positive integer so that $v \times Q - 1$ is a multiple of lcm(p_1–1, … p_f–1). The number *Q* is represented by a string of α bits.

NOTE 2 The private key *Q* and the modulus *n*(*A*) together form a private RSA key.

Denoted G, the public key is a domain parameter. The value $G = 2$ has some practical advantages.

NOTE 3 The number playing the role of the base is the v-th modular power of G, i.e. $g(A) = G^v \bmod n(A)$. It is used neither by the claimant, nor by the verifier.

7.3 Unilateral authentication exchange

The bracketed numbers in Figure 4 correspond to the steps of the mechanism, including the exchanges of information, described in detail below. The claimant is denoted A. The verifier is denoted B.

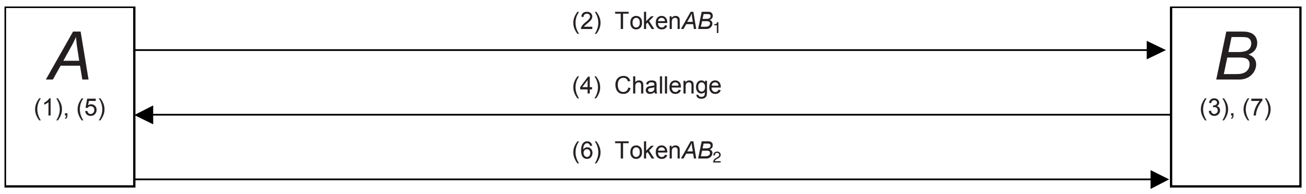

Figure 4 — Mechanism using a discrete logarithm with respect to a composite number

— In the first mode, the claimant shall store a number δ, a base g, a private key Q (as a string of σ bits) and a modulus n or $n(A)$. Unless otherwise specified, $\delta = 40$, $g = 2$, $\sigma = 160$.

— In the second mode, the claimant shall store a number δ, a public key G, a verification exponent v, a private key Q (as a string of α bits) and a modulus $n(A)$. Unless otherwise specified, $\delta = 40$, $G = 2$, $v = 2^{40}+15$.

In the case of a coupon strategy, in addition to a number δ and a private key Q, the claimant shall only store a set of coupons. To be used only once, each coupon consists of a ρ-bit string (that needs not be stored if it can be reproduced by a pseudo-random function) and an α-bit witness (or preferably, its hash-code).

— In the first mode, in addition to a number δ, a base g and a number σ, the verifier shall be provided with a trusted copy of a public key $G(A)$ and a trusted copy of a modulus n or $n(A)$.

— In the second mode, in addition to a number δ, a public key G and a verification exponent v, the verifier shall be provided with a trusted copy of a modulus $n(A)$.

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

1) For each authentication, a fresh string of ρ bits shall be uniformly selected at random. It shall be kept secret.

In the first mode,	$\rho = \sigma + \delta + 80$.
In the second mode,	$\rho = \alpha + \delta + 80$.

NOTE 1 If the fresh string of ρ bits is selected at random, then the probability that the leftmost 80 bits are all equal is negligible.

Denoted r, the number represented by the fresh string shall be converted into a witness, denoted W. The number W is represented by a string of α bits, also denoted W.

Witness formula in the first mode:	$W = g^r$ (mod either n or $n(A)$)
Witness formula in the second mode:	$W = G^{r \times v} \bmod n(A)$

NOTE 2 If the prime factors are available, then the witness computation (performed in advance in the case of a coupon strategy) may make use of the CRT technique (see C.2.3).

2) *A* sends TokenAB_1 to *B*. TokenAB_1 is either witness *W* or a hash-code of *W* and *Text*, one of the following four hash variants.

   The four hash variants are $h(W \,||\, Text)$, $h(W \,||\, h(Text))$, $h(h(W) \,||\, Text)$, and $h(h(W) \,||\, h(Text))$, where *h* is a hash-function and *Text* is an optional text field (it may be empty). If the text field is non-empty, then *B* shall have the means to recover the value of *Text*; this may require *A* to send all or part of the text field at this point. The text field is available for use in applications outside the scope of this part of ISO/IEC 9798. Annex A of ISO/IEC 9798-1 [24] gives information on the use of text fields. The hash variant is a domain parameter.

3) On receipt of TokenAB_1, a fresh string of δ bits shall be uniformly selected at random.

4) *B* sends the fresh string as a challenge to *A*. The fresh string represents a number denoted *d*.

5) On receipt of the challenge, the following computational steps are performed.
   a) If the challenge is not a string of δ bits, then the procedure fails.
   b) A response *D* shall be computed from the random number *r* and the private key *Q*.

      Response formula: $D = r - d \times Q$

6) *A* sends TokenAB_2 to *B*. TokenAB_2 is the response *D* computed from step 5)b).

7) On receipt of TokenAB_2, the following computational steps are performed.
   a) If the response *D* is not a string of ρ bits and/or if the leftmost 80 bits of *D* are all equal, then the procedure fails.
   b) Denoted W^*, a witness shall be computed.

      Verification formula in the first mode: $W^* = G(A)^{d} \times g^{D}$ (mod either n or $n(A)$)

      Verification formula in the second mode: $W^* = G^{d + v \times D} \bmod n(A)$
   c) If either witness W^* or a hash-code of W^* and *Text*, one of the four hash variants, is identical to TokenAB_1 received in step (2), then the procedure is successful. Otherwise the procedure fails.

NOTE 1 Other information may be sent with any exchange of the procedure. *B* may use such information to help compute the value of the optional text field. For example, *A* may send information such as certificates with TokenAB_1.

NOTE 2 The use of a hash-code instead of witness W in TokenAB1 can achieve efficiency gains by reducing the number of bits in TokenAB1.

8 Mechanisms based on asymmetric encryption systems

8.1 Security requirements for the environment

These mechanisms enable a verifier to check that a claimant knows the decryption key corresponding to a claimed encryption key.

NOTE These mechanisms derive from schemes due to Brandt, Damgård, Landrock and Pedersen [2][16]. The second mechanism also derives from the key transport mechanism 6 from ISO/IEC 11770-3 [26], and Mitchell and Yeun [17].

Within a given domain, the following requirements shall be satisfied.

1) All entities within the domain shall agree on the use of two cryptographic functions: a hash-function, e.g. one of the functions specified in ISO/IEC 10118-3 [25], and an asymmetric encryption system, e.g. one of the systems specified in ISO/IEC 18033-2 [31].

2) Every claimant shall be equipped with an asymmetric key pair for use with the asymmetric encryption system.

3) Every verifier shall obtain a trusted copy of the public key specific to the claimant.

   NOTE The exact means by which the verifier obtains a trusted copy of the public key specific to the claimant is beyond the scope of this part of ISO/IEC 9798. This may, for example, be achieved by the use of public-key certificates or by some other environment-dependent means.

4) Every verifier shall have the means to produce fresh strings of random bits.

8.2 Unilateral authentication exchange

The bracketed numbers in Figure 5 correspond to the steps of the mechanism, including the exchanges of information, described in detail below. The claimant is denoted *A*. The verifier is denoted *B*.

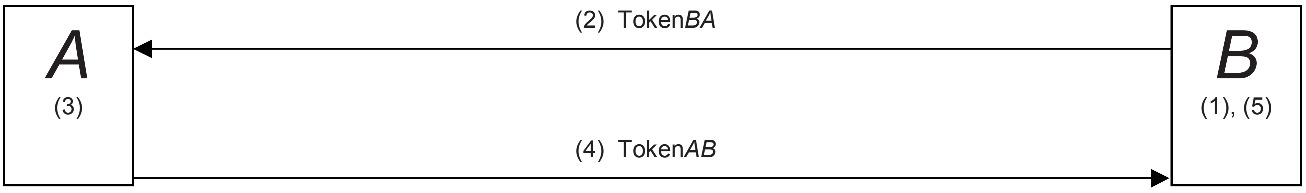

Figure 5 — Mechanism using an asymmetric key pair for encryption

The claimant shall store the private portion of an asymmetric key pair, defining a private operation denoted S_A.

The verifier shall be provided with a trusted copy of the public portion of an asymmetric key pair, defining a public operation denoted P_A.

If a coupon strategy is being used, the verifier shall store a set of coupons. To be used only once, each coupon is dedicated to a given claimant; it consists of a ρ-bit string (that needs not be stored if it can be reproduced by a pseudo-random function) and an α-bit challenge.

The bit length of the fresh strings of random bits, a number denoted ρ, shall be selected. The value of ρ shall be at least $2 \times |h|$, but less than $|n(A)| - |h|$, so that the concatenation of a fresh string with a hash-code lies within the domain of definition of P_A.

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

1) The following computational steps are performed.
   a) For each authentication, a fresh string of ρ bits shall be uniformly selected at random. Denoted *r*, it shall be kept secret.

      The value of ρ shall be at least $2 \times |h|$, but less than $|n(A)| - |h|$, so that the concatenation of a fresh string with a hash-code lies within the domain of definition of P_A.
   b) A hash-code *H* shall be computed from the fresh string *r*.
   $$H = h(r)$$
   c) A number *d* shall be computed using P_A.
   $$d = P_A(r \,||\, H)$$

2) *B* sends Token*BA* to A. Token*BA* is the number *d* computed from step 1)c).

3) On receipt of Token*BA*, the following computational steps are performed.
   a) Two strings denoted r^* and H^* shall be recovered using S_A.
   $$r^* \,||\, H^* = S_A(d)$$
   b) If the string H^* and the hash-code $h(r^*)$ are different, then the procedure fails.

4) *A* sends Token*AB* to *B*. Token*AB* is the string *r** recovered from step 3)a).

5) On receipt of Token*AB*, the string *r** is compared with the string *r*. If they are identical, then the procedure is successful; otherwise the procedure fails.

NOTE 1 If the encryption system in use provides the property of non-malleability (see ISO/IEC 18033-2 [31]), then the hash-code may be omitted from Token*BA*. In such a case, step 3.b is replaced by a check that the decryption process completes correctly. However special care should then be taken to prevent the verifier using the claimant as a decryption oracle.

NOTE 2 Other information may be sent with either of the exchanges of the mechanism.

8.3 Mutual authentication exchange

The bracketed numbers in Figure 6 correspond to the steps of the mechanism, including the exchanges of information, described in detail below. Each entity, *A* as *B*, is a claimant and a verifier.

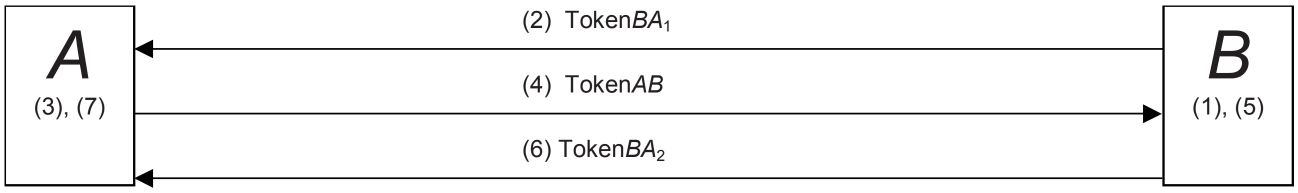

Figure 6— Mechanism using two asymmetric key pairs for encryption

Each entity shall store the private portion of their asymmetric key pair, defining a private operation denoted either S_A, or S_B and be provided with a trusted copy of the public portion of the asymmetric key pair of the other entity, defining a public operation denoted either P_B, or P_A. It shall also be provided with its own identification data, denoted either *Id*(*A*) or *Id*(*B*), and the identification data of the other entity, denoted either *Id*(*B*) or *Id*(*A*).

The bit length of the fresh strings of random bits, a number denoted ρ, shall be selected. The value of ρ shall be at least $2\times|h|$, but less than $\min(|n(A)|-|h|-|Id(B)|, (|n(B)|-|h|-|Id(A)|)/2)$, so that

— the concatenation of *Id*(*B*) and a fresh string with a hash-code lies within the domain of definition of P_A.

— the concatenation of *Id*(*A*) and two fresh strings with a hash-code lies within the domain of definition of P_B.

For each application of the mechanism, the following procedure shall be performed. The two entities *A* and *B* shall only accept each other as valid if the procedure completes successfully.

1) The following computational steps are performed.
   a) For each authentication, a fresh string of ρ bits shall be uniformly selected at random. Denoted r_B, it shall be kept secret.
   b) A hash-code H_B shall be computed from the identification data *Id*(*B*) and the fresh string r_B.
   $$H_B = h(Id(B) \,||\, r_B)$$
   c) A number d_B shall be computed using P_A.
   $$d_B = P_A(Id(B) \,||\, r_B \,||\, H_B)$$

2) *B* sends TokenBA_1 to *A*. TokenBA_1 is the number d_B computed from step 1)c).

3) On receipt of TokenBA_1, the following computational steps are performed.
   a) Three strings denoted Id_B^*, r_B^* and H_B^* shall be recovered using S_A.
   $$Id_B^* \,||\, r_B^* \,||\, H_B^* = S_A(d_B)$$
   b) If the string H_B^* and the hash-code $h(Id_B^* \,||\, r_B^*)$ are different, then the procedure fails.
   c) If the string Id_B^* and the identification data *Id*(*B*) are different, then the procedure fails.

d) The following computational steps are performed.
   i. For each authentication, a fresh string of ρ bits shall be uniformly selected at random. Denoted r_A, it shall be kept secret.
   ii. A hash-code H_A shall be computed from the identification data $Id(A)$, the string $r_B{}^*$ and the fresh string r_A.
   $$H_A = h(Id(A) \mid\mid r_B{}^* \mid\mid r_A)$$
   iii. A number d_A shall be computed using P_B.
   $$d_A = P_B(Id(A) \mid\mid r_B{}^* \mid\mid r_A \mid\mid H_A)$$

4) *A* sends Token*AB* to *B*. Token*AB* is the number d_A.

5) On receipt of Token*AB*, the following computational steps are performed.
   a) Four strings denoted $Id_A{}^*$, $r_B{}^{**}$, $r_A{}^*$ and $H_A{}^*$ shall be recovered using S_B.
   $$Id_A{}^* \mid\mid r_B{}^{**} \mid\mid r_A{}^* \mid\mid H_A{}^* = S_B(d_A)$$
   b) If the string $H_A{}^*$ and the hash-code $h(Id_A{}^* \mid\mid r_B{}^{**} \mid\mid r_A{}^*)$ are different, then the procedure fails.
   c) If the string $Id_A{}^*$ and the identification data $Id(A)$ are different, then the procedure fails.
   d) If the string $r_B{}^{**}$ and the string r_B produced at step (1) are different, then the procedure fails.

6) *B* sends TokenBA_2 to *A*. TokenBA_2 is the string $r_A{}^*$.

7) On receipt of TokenBA_2, the string $r_A{}^*$ is compared with the string r_A produced at step (3). If they are identical, then the procedure is successful; otherwise the procedure fails.

NOTE 1 If the encryption system in use provides the property of non-malleability (see ISO/IEC 18033-2 [31]), then the hash-codes may be omitted from TokenBA_1 and Token*AB*. In such a case, steps 3.b and 5.b are replaced by checks that the decryption process completes correctly. However special care should then be taken to prevent the verifier using the claimant as a decryption oracle.

NOTE 2 Other information may be sent with any of the exchanges of the mechanism.

9 Mechanism based on discrete logarithms with respect to elliptic curves

9.1 Security requirements for the environment

This mechanism enables a verifier to check that a claimant knows the elliptic curve discrete logarithm of a claimed public point with respect to a base point. A general framework for cryptographic techniques based on elliptic curves is given in ISO/IEC 15946-1 [28].

NOTE 1 This mechanism implements the elliptic curve variant [6] of the GPS [9] scheme due to Girault, Poupard and Stern. It allows use of the so-called LHW (Low Hamming Weight) variant [7], particularly suitable for environments where the resources of the claimant are very low.

Within a given domain, the following requirements shall be satisfied.

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

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

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

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

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

NOTE 2 The exact means by which the verifier obtains a trusted copy of the public point specific to the claimant is beyond the scope of this part of ISO/IEC 9798. This may, for example, be achieved by the use of public-key certificates or by some other environment-dependent means.

6) Every claimant and every verifier shall have the means to produce fresh strings of random bits.

7) If the mechanism makes use of a hash-function, then all entities within the domain shall agree on a hash-function, e.g. one of the functions specified in ISO/IEC 10118-3 [25].

9.2 Key production

For claimant *A*, a fresh string shall be uniformly selected at random from the set [2, *n* – 2]. The string represents the private key, denoted *Q*.

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

Denoted *G*(A), the public point for claimant A is set equal to the multiplication of number Q by the base point *P*.

$G(A) = (x_G, y_G) = [Q]P$

The challenges are selected from a set of integers S of cardinality Δ, where $2^{\delta - 1} < \Delta \leqq 2^{\delta}$. The length in bits of the greatest possible challenge is denoted by *β*. A value of δ from 8 to 40 is appropriate for most applications. Unless otherwise specified, the value of δ is set equal to 40. It is a domain parameter.

NOTE 1 The total number of possible challenges should be limited to 2^{40}. If this recommendation is not followed, then special care should be taken to prevent the verifier using the claimant as a signing oracle.

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

NOTE 3 A challenge is said to be LHW (Low Hamming Weight) if there are at least $\sigma - 1$ zero bits between any two consecutive one bits in its binary representation.

9.3 Unilateral authentication exchange

The bracketed numbers in Figure 7 correspond to the steps of the mechanism, including the exchanges of information, described in detail below. The claimant is denoted by *A*. The verifier is denoted by *B*.

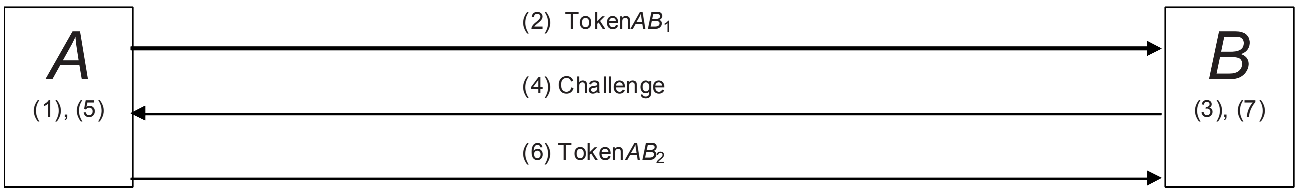

Figure 7 — Mechanism using a discrete logarithm with respect to elliptic curves

The claimant shall store a number *δ*, a base *P*, and a private key Q (as a string of *σ* bits). Unless otherwise specified, *δ* = 40.

In the case of a coupon strategy, in addition to a number *δ* and a private key *Q*, the claimant shall only store a set of coupons. To be used only once, each coupon consists of a *ρ*-bit string (that need not be stored if it can be reproduced by a pseudo-random function) and a witness.

In addition to a number *δ* and a number *σ*, the verifier shall be provided with a trusted copy of a public point *G*(*A*) and a trusted copy of the curve *E*, the base point *P* and the parameters *q* and *n*.

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

1) For each authentication, a fresh string of ρ bits shall be uniformly selected at random. It shall be kept secret.

$$\rho = \sigma + \beta + 80$$

NOTE 1 If the fresh string of ρ bits is selected at random, then the probability that the leftmost 80 bits are all equal is negligible.

Denoted *r*, the number represented by the fresh string shall be converted into a witness, denoted *W*.

Witness formula: $W = P2OS([r]P)$

NOTE 2 *P2OS* is the function used to convert a point to an octet string.

2) *A* sends TokenAB_1 to *B*. TokenAB_1 can be either witness *W* or a hash-code of *W* and *Text*, one of the following four hash variants, to *B*.

The four hash variants are $h(W \,||\, Text)$, $h(W \,||\, \mathrm{h}(Text))$, $h(h(W) \,||\, Text)$, and $h(h(W) \,||\, h(Text))$, where *h* is a hash-function and *Text* is an optional text field (it may be empty). If the text field is non-empty, then B shall have the means to recover the value of *Text*; this may require that A sends all or part of the text field with the token. How the text field is made available for use in applications is outside the scope of this part of ISO/IEC 9798. Annex A of ISO/IEC 9798-1 [24] gives information on the use of text fields. The hash variant is a domain parameter.

3) On receipt of TokenAB_1, a fresh string shall be uniformly selected at random from the set *S*.
4) *B* sends the fresh string as a challenge to *A*. The fresh string represents a number denoted *d*.

NOTE 3 If an LHW challenge is used, it can be transmitted in a compressed form to *A* who must have the means to retrieve the original challenge before step 5a

5) On receipt of the challenge, the following computational steps are performed.
   a) If the challenge is not an element of *S*, then the procedure fails.
   b) A response *D* shall be computed from the random number *r* and the private key *Q*.

Response formula: $D = r - d \times Q$

NOTE 4 If the challenge received is an LHW challenge, the computation of D is reduced to a serial addition of r with a concatenation of copies of Q, separated by zero bits.

6) *A* sends TokenAB_2 to *B*. TokenAB_2 is the response *D* computed from step 5)b).

7) On receipt of TokenAB_2, the following computational steps are performed.
   a) If the response *D* is not a string of *ρ* bits and/or if the leftmost 80 bits of *D* are all equal, then the procedure fails.
   b) Denoted *W**, a witness shall be computed.

Verification formula: $W^* = \mathrm{P2OS}([d]G(A) + [D]P)$

   c) If either witness W* or a hash-code of W* and Text (one of four hash variants) is identical to TokenAB_1 received in step (2), then the procedure is successful. Otherwise the procedure fails.

NOTE 5 Other information may be sent with any exchange of the procedure. *B* may use such information to help compute the value of the optional Text field. For example, *A* may send information such as certificates with TokenAB_1.

Annex A
(normative)

Object identifiers

A.1 Formal definition

```
EntityAuthenticationMechanisms-9 {
   iso(1) standard(0) e-auth-mechanisms(9798)
      part(5) asn1-module(0) object-identifiers(0) }
   DEFINITIONS EXPLICIT TAGS ::= BEGIN

-- EXPORTS All; --

-- IMPORTS None; --


OID ::= OBJECT IDENTIFIER -- alias

-- Synonyms --

is9798-5 OID ::= { iso(1) standard(0) e-auth-mechanisms(9798) part(5) }

mechanism OID ::= { is9798-5 mechanisms(1) }

-- Unilateral and mutual entity authentication mechanisms --

ua-identity-based-FS OID ::= { mechanism 1 }
ua-identity-based-GQ1 OID ::= { mechanism 2 }
ua-integer-factorization-GQ2 OID ::= { mechanism 3 }
ua-discrete-logarithms-prime-number-SC OID ::= { mechanism 4 }
ua-discrete-logarithms-composite-number-GPS1 OID ::= { mechanism 5 }
ua-discrete-logarithms-composite-number-GPS2 OID ::= { mechanism 6 }
ua-asymmetric-encryption OID ::= { mechanism 7 }
ma-asymmetric-encryption OID ::= { mechanism 8 }
ua-discrete-logarithms-ecc-GPS OID ::= { mechanism 9 }


END -- EntityAuthenticationMechanisms-9 --
```

A.2 Use of subsequent object identifiers

If a mechanism specified in this part of ISO/IEC 9798 uses a hash-function, then just after an object identifier identifying the mechanism, another object identifier may follow for identifying a hash-function (e.g. one of the dedicated hash-functions specified in ISO/IEC 10118-3 [25]).

For the last two mechanisms, another object identifier may follow for referring to an encryption system (e.g. one of the mechanisms specified in ISO/IEC 18033-2 [31]). In the absence of such a subsequent object identifier, an RSA permutation is used.

A.3 Coding examples in accordance with the basic encoding rules of ASN.1

In accordance with ISO/IEC 8825-1[23], an object identifier consists of one or more series of octets. Each series codes a number.

— Bit 8 (the most significant bit) is set equal to zero in the last octet of a series and to one in the previous octets, if there is more than one octet.

— The concatenation of bits 7 to 1 of the octets of a series codes a number. Each number shall be encoded on the fewest possible octets, that is, the octet '80' is invalid in the first position of a series.

— The first number is the number of the standard; the second number, if present, is the part in a multi-part standard.

An object identifier may refer to any mechanism defined in this part of ISO/IEC 9798.

— For identifying an ISO standard, the first octet is set equal to '28', i.e. 40 in decimal (see ISO/IEC 8825-1).

— The next two octets are set equal to 'CC46'. 9798 is equal to '2646' in hexadecimal, i.e. 0010 0110 0100 0110, i.e. two blocks of seven bits: 1001100 1000110. After insertion of the appropriate value of bit 8 in each octet, the coding of the series is therefore 11001100 01000110, i.e. 'CC46'.

— The next octet is set equal to '05' for identifying part 5.

— The next octet identifies an authentication mechanism.

  — '01' identifies the unilateral authentication mechanism using FS.

  — '02' identifies the unilateral authentication mechanism using GQ1.

  — '03' identifies the unilateral authentication mechanism using the factorization of a modulus, i.e. GQ2.

  — '04' identifies the unilateral authentication mechanism using a discrete logarithm with respect to a prime number, i.e. SC.

  — '05' identifies the unilateral authentication mechanism using a discrete logarithm with respect to a composite number in the first mode of use, i.e. GPS1.

  — '06' identifies the unilateral authentication mechanism using a discrete logarithm with respect to a composite number in the second mode of use, i.e. GPS2.

  — '07' identifies the unilateral authentication mechanism using an asymmetric encryption system.

  — '08' identifies the mutual authentication mechanism using an asymmetric encryption system.

For example, the data element '28 CC 46 05 03' reads {iso standard 9798 5 3}, i.e. the third mechanism in ISO/IEC 9798-5, i.e. GQ2. The data element may be conveyed in the following BER-TLV data object (see the basic encoding rules of ASN.1, ISO/IEC 8825-1, universal class tag '06') where the dashes and the curly brackets inserted for clarity are not significant.

Data object = {'06'-'05'-'28 CC 46 05 03'}

Annex B
(informative)

Principles of zero-knowledge techniques

B.1 Introduction

In the context of the use of asymmetric cryptographic techniques, a potential weakness of an authentication exchange is that the verifier may abuse the mechanism to compromise the private key. When asymmetric cryptography is being used, the claimant uses the private key of his asymmetric pair to compute a response to a verifier's challenge. The verifier may then, by choosing the challenge wisely, gain information about the private key of the claimant that could not have been obtained just from knowledge of the public key of the claimant.

This type of abuse of an exchange of cryptographic messages is known as using the claimant as an oracle, in that the claimant provides information about his private key at the behest of the verifier. The idea behind a zero-knowledge authentication mechanism is simply to remove this particular potential threat by careful design of the messages in such a way that the verifier cannot use the claimant as an oracle.

B.2 Need for zero-knowledge mechanisms

In applications involving modern computer networks, the need for security services such as authentication, non-repudiation, etc., is widely recognized and steadily growing. In order to be able to use such services, it is necessary for a user to have access to private information, specific to that user. Examples are passwords, signature keys, private keys of asymmetric pairs, etc.

It is of course mandatory for the security of the system that the private information stays private, i.e. does not leak to other potentially hostile parties. On the other hand, the private information shall be used as input to the software or hardware modules that compute and send messages on behalf of the user. If the information is not properly used, the secrecy of the private information may be damaged, or even destroyed completely. An obvious example is when users identify themselves to a host by sending a password in cleartext. This reveals totally the private information with the immediate result that anyone eavesdropping on the line can impersonate all users whose passwords have been intercepted.

This is an example where too much information is being communicated. To illustrate this, note that from the point of view of the host, there are only two possibilities: either the user possesses the correct password or he does not. In information theoretic terms, this means that only one bit of information really needs to be communicated. By sending the entire password, we therefore communicate much more than is needed, and this is the theoretical background for the practical problem of eavesdropping.

It is natural to ask: "— Can one design protocols for use of private information which communicate exactly the information they are meant to communicate, and nothing more?" Informally, this is precisely the property that a zero-knowledge mechanism has. Consider for example a situation where user A is assigned an asymmetric pair of keys or numbers for an asymmetric cryptographic system (P_A, S_A), so that P_A is public while S_A is private to A. Then using a zero-knowledge mechanism, A can convince B that A possesses the private key corresponding to P_A without revealing anything other than this fact. Since A is characterized as the only user with access to S_A, this protocol can be used for authentication. In this case, the zero-knowledge property guarantees that B will learn nothing that could help him to later falsely impersonate A.

The zero-knowledge property is achieved by designing a dialogue that can be simulated by the verifier alone. This intuitively proves that the verifier will learn nothing from the claimant in terms of properties of the private key, which the verifier could not have obtained from the corresponding public key.

It also means that an observer to the exchange of messages making up the mechanism will be unable to decide if the claimant really was involved, or the verifier simulated the exchange.

Zero-knowledge mechanisms by nature require the use of asymmetric cryptographic techniques. Given the strict definition of a zero-knowledge mechanism, it is actually not possible to implement one. In fact, a much better description of the mechanisms in this part of ISO/IEC 9798 would be secrecy-preserving mechanisms. However, the concept of zero-knowledge mechanism is part of a well-known and established theory in cryptography, for which reason the terminology is used here.

B.3 Definitions

Going a little closer to a formal definition, a zero-knowledge mechanism takes place between two parties, a claimant A and a verifier B. The claimant tries to convince the verifier that a certain statement is true. For example, this statement could be "I know the private key corresponding to P_A". To convince B, the claimant and verifier exchange messages for a while, after which B decides to accept or reject A's proof.

Three essential properties are needed for such a mechanism.

Completeness. If A's statement is true, then B should accept it with overwhelming probability.

Soundness. If A's statement is false, then no matter how A behaves, B should reject it with overwhelming probability.

Zero-knowledge. No matter how B behaves, he receives only the information that A's statement is true. A little more precisely: whatever B receives when talking to a truthful claimant, B could just as easily compute himself without talking to A at all. What this means is that B can simulate the conversation by himself, producing a conversation that looks exactly as if it had been produced by talking to A.

B.4 Example

Consider the following example, which is a simplified version of an FS mechanism [4]. Here, we are given a modulus n and a number modulo n, named G. In this case, A's statement is "I know a modular square root of G". Note that Q is a modular square root of G, if and only if $Q^2 \bmod n \equiv G$.

The conversation between A and B goes as follows.

- A chooses a fresh random number r, non-zero and less than n, squares it modulo n and sends the modular square W to B.
- B chooses a fresh random bit d, i.e. either 0 or 1, and sends it to A as a challenge.
- If d is equal to zero, then the response is $D = r$. If d is equal to one, the response is $D = r \times Q \bmod n$. A sends the response D to B.
- B first checks that D is a non-zero number less than n; if D is zero, n or more, then B rejects A and aborts the procedure.
- If d is equal to zero, then B checks that the modular square of D is identical to W. If d is equal to one, then B checks that the modular square of D is identical to $W \times G \bmod n$.
- If the check is correct, then continue the procedure, else B rejects A and aborts the procedure.

The procedure completes successfully after t consecutive successful iterations.

It is not too difficult to see that if both A and B follow this procedure, then B will never reject A; squaring D means squaring either r or $r \times Q \bmod n$, which will give the result W or $W \times G \bmod n$.

On the other hand, if in any of the *t* iterations, *A* is able to give a correct answer to both $d = 0$ and $d = 1$, this means that *A* can provide both D_0 and D_1. As a matter of fact, $D_1 / D_0 \bmod n$ is a modular square root of *G* and therefore the statement that "*A* knows a modular square root of *G*" is true. But conversely, if *A* is cheating and does not know a modular square root of *G*, he shall be unable to answer at least one value of *d* correctly in each of the *t* iterations. Therefore the probability that a cheating claimant convinces the verifier is at most 2^{-t}. For example, by doing 20 iterations, we reduce this chance to about 1 in a million. Such a value is named "mechanism security level" (see also C.1.4). Thus the soundness property is also satisfied.

As for zero-knowledge, note that, after the conversation is over, the verifier is left with two numbers *D* and *W*, so that $D^2 \bmod n$ is equal to either *W* or $G\,W \bmod n$. But this is indeed something that the verifier could make himself without talking to *A*. To do this, *B* just chooses a random number *D* and defines *W* either as D^2 or as $D^2 / G \bmod n$. The fact that *W* and *D* are, in this case, computed in a way different from the way the claimant would compute them is insignificant; they are distributed in exactly the same way, i.e. it is impossible to tell the difference. Therefore, *B* learns nothing he could not compute himself, except for the fact that *A* knows a modular square root of *G*.

Let us anticipate here a frequently asked question. If the verifier can make good looking conversations himself, without knowing a root of *G*, why should he be convinced when the claimant produces a similar conversation? The answer is that when *B* simulates the protocol, he is free to produce the numbers in a backwards direction, i.e. to first choose *D* and then compute a *W* that fits. In a real protocol execution, *A* does not have this opportunity. The verifier expects to see *W* before *d* is selected, and then the claimant shall find a correct *D*.

Although we have glossed over a couple of technical difficulties here, these are the essentials of the argument why a mechanism has the zero-knowledge property.

B.5 Basic design principles

The example from the previous section covers one of two basic design ideas that underlie almost all known zero-knowledge mechanisms, namely:

- The claimant *A* sends a witness to the verifier *B*. Then *B* asks *A* one out of some set of questions. If *A* is cheating, he cannot answer all possible questions, so we have some chance of catching him. On the other hand, *A* never answers more than one question, and this one answer alone reveals nothing to the verifier.

This design idea forms the basis of the mechanisms specified in clauses 5, 6, 7 and 8.

The other design idea, and one which forms the basis of the mechanism specified in clause 9, is based on the following:

- The verifier asks the claimant a question, for which the verifier already knows the answer. The protocol shall ensure that this really is the case. If *A* is honest, he can easily compute the right answer, but if he is cheating, he can do no better than guess at random, and will be incorrect most of the time.

- On the other hand, when *B* receives the answer, he already knows what *A* will say, and therefore the mechanism has the zero-knowledge property.

One easy example of this is when *A* shall prove possession of a private key in a public-key system. The verifier can encipher a random message under *A*'s public key, and ask *A* to return the deciphered message. Only the user knowing the correct private key can do this. To get the zero-knowledge property, we shall ensure that *B* really knows the message in advance. This part of ISO/IEC 9798 contains an example of one way to do this, namely *B* can be asked to reveal some information (the witness) related to the message.

The bibliography indicates a comprehensive approach of zero-knowledge protocols [20] and a formal basis for a rigorous understanding of zero-knowledge protocols [3][10].

Annex C
(informative)

Guidance on parameter choice and comparison of the mechanisms

C.1 Guidance on parameter choice

C.1.1 Modulus sizes

In this part of ISO/IEC 9798, every authentication mechanism makes use of a modulus that is either prime (for the SC mechanism) or composite (for any other mechanism).

In 1995, Odlyzko [18] estimated the future of integer factorization and discrete logarithms. "*With the present state of knowledge, discrete logarithms are slightly more difficult to compute modulo an appropriately chosen prime than it is to factor a hard integer of the same size, but the difference is not large. Therefore, to be on the safe side in designing cryptosystems, one should assume that all the projections about sizes of integers that it will be possible to factor will also apply to sizes of primes modulo which one can compute discrete logarithms*."

As a conclusion at the end of the quoted article [18], Kaliski stressed the importance of variable key sizes in the implementations and provided recommendations on modulus sizes.

— Short term security: 768 bits.

— Medium term security: 1024 bits.

— Long term security: 2048 bits.

For a comprehensive analysis of key lengths, see also Silverman [22], and Lenstra and Verheul [14].

C.1.2 Composite modulus and prime factors

Throughout the standard, the distinct large prime factors are denoted p_1, p_2 ... in ascending order, the modulus is set equal to the product of the prime factors, i.e. $n = p_1 \times p_2 \times \ldots$ and α denotes the bit size of the modulus, i.e. $2^{\alpha} / 2 < n < 2^{\alpha}$. Moreover, the standard states that, if α is a multiple of the number of prime factors, denoted f, then the bit size of every prime factor shall be α / f, i.e. $2^{\alpha / f} / 2 < p_1 \ldots < p_f < 2^{\alpha / f}$.

NOTE 1 ISO/IEC 18032 [29] specifies how to select large prime numbers.

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

— The first prime factor is selected within the interval from $2^{\alpha / f} / 2$ to $2^{\alpha / f}$. The initial value of z is set equal to the first prime factor.

— This step is repeated f–1 times. A new prime factor is selected within the interval from $(2^{|z|}/z) \times 2^{\alpha / f} / 2$ to $2^{\alpha / f}$. 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 the modulus n is set equal to the final value of z.

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

— Every prime factor is selected within the interval from $\beta \times (2^{\alpha / f})$ to $2^{\alpha / f}$ where β denotes the f-th root of 1/2.

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

C.1.3 Lengths of fresh strings of random bits for representing random numbers

In the mechanisms specified in Clauses 5 to 8, the claimant converts any random number r into a witness W in accordance with a witness formula and then produces a response D to any challenge d in accordance with a response formula. The procedure parameters W, d and D together form a zero-knowledge proof, i.e. a triple denoted $\{W, d, D\}$ satisfying a verification formula. The set of proofs forms a family of d permutations of the set of, or a subset of, the integers with respect to the modulus; this set of integers is either a field, or a ring.

As any third party can use the verification formula for computing a witness W from any challenge d and response D selected at random, i.e. for producing triples at random, it is important that the set of triples is so large that the advantage obtained by producing in advance as many triples as possible remains negligible.

It is important that the claimant chooses random numbers in such a way that the probabilities of guessing them and the same number being selected twice within the claimant's lifetime are negligible. If, for example, a claimant uses twice the same random number, then he will produce an "interlocked" pair of triples, i.e. responses to two challenges for the same non-zero witness, denoted $\{W, d_1, D_1\}$ and $\{W, d_2, D_2\}$.

— In the FS mechanisms, as any interlocked pair of triples provides a modular multiplicative combination of private keys, any third party will improve its performances for impersonating the claimant.

— In the GQ1, SC and GPS mechanisms, as any interlocked pair of triples provides the private key. With the private key, any third party is able to impersonate the claimant.

— In the GQ2 mechanisms, the key production ensures that, for any values of m and k, more than one half of all the interlocked pairs of triples reveals a non-trivial modular square root of 1. The knowledge of such a number induces the knowledge of a decomposition of the modulus, i.e. the factorization if there are two factors. With the factorization, any third party is able to impersonate the claimant.

On receipt of TokenAB_1, i.e. either a witness W, or a hash-code of W and *Text*, the verifier produces a challenge d at random. It is important that all the possible challenges are equally probable and hence the challenge is unpredictable. Any cheater can succeed in a masquerade by guessing the challenge in advance. If 2^{δ} challenges are equally probable, then the probability of success of a cheater is one chance out of 2^{δ}.

In the mechanisms specified in Clause 9, there is no witness. The verifier constructs a number d from a random parameter r that is the response D. The numbers d and D together form a proof denoted $\{d, D\}$. Such a proof is of "zero-knowledge type". The set of proofs is a very small fraction of the RSA permutation, much smaller than the sets of proofs used in the mechanisms specified in Clauses 5 to 8. It is important that the verifier chooses random parameters in such a way that the probabilities of guessing them and re-using them are negligible. Any third party can use the public operation for producing pairs at random. It is important that the set of pairs is so large that the advantage obtained by producing in advance as many pairs as possible remains negligible.

As a conclusion, the length of the fresh strings of random bits for representing random numbers is set equal to

— α bits in FS, GQ1 and GQ2.

— $|q|$ bits in SC (typically, $|q| = 160$).

— $\sigma + \delta + 80$ bits in GPS1 (typically, $\sigma = 160$).

— $\alpha + \delta + 80$ bits in GPS2.

— at least $2 \times |h|$, but less than $\alpha - |h|$ bits in RSA_{UA} (typically, $|h| = 160$).

— at least $2 \times |h|$, but less than $0{,}5 \times (\alpha - |h| - |ID|)$ bits in RSA_{MA} (typically, $|h| = 160$ and $|ID| = 40$).

C.1.4 Strategies for the use of the various mechanisms

This clause considers four groups of mechanisms in accordance with the analysis of the formula complexities.

a) FS, GQ1 and GQ2, i.e. the mechanisms specified in Clauses 5 and 6;

b) SC, GPS1 and GPS2, i.e. the mechanisms specified in Clauses 7 and 8;

c) RSA_{UA}, i.e. the mechanisms specified in Clause 9.2.

d) RSA_{MA}, i.e. the mechanisms specified in Clause 9.3.

NOTE Consider a portable device so that power analysis distinguishes squaring and multiplying. In order to keep the exponents secret, countermeasures are needed for implementing the SC and GPS witness formulae and the private RSA operation. But as the exponents are public in the FS and GQ witness and response formulae, the implementation is straightforward.

In the FS, GQ1 and GQ2 mechanisms, the witness is the modular v-th power of a random number r. The verification exponent v is short (up to 40 bits). The witness formula is a **short modular exponentiation**. The response formula is also a short, possibly combined, modular exponentiation; it allows trade-offs between computational complexity and storage requirement. Nevertheless, the response formula and the witness formula have a similar complexity. The verification formula is a short combined modular exponentiation; it induces a verifier workload similar to the claimant workload.

➢ The FS, GQ1 and GQ2 mechanisms are attractive in systems where the claimant and the verifier have similar performances. For example, if the claimant is a smart card, then, as the verifier workload and the claimant workload are similar, the computational power of the smart card is sufficient for a verifier. Consequently, a payment card and a merchant card may authenticate each other, either locally through a payment terminal, or even remotely through the Internet.

In the FS, GQ1 and GQ2 mechanisms, the challenge size has to be optimized: the least the challenge, the shortest the modular exponentiations. For example, there are $2^{k \times m}$ possible GQ2 challenges.

— One chance out of 2^{36} may be an adequate security level in a high security environment, e.g. either k = 18 and two basic numbers, or k = 12 and three basic numbers, or k = 6 and six basic numbers.

— One chance out of 2^{24} may be an adequate security level through the Internet, e.g. either k = 12 and two basic numbers, or k = 8 and three basic numbers, or k = 4 and six basic numbers.

— One chance out of 65 536 may be an adequate security level for deterring "yes cards" on automated paying machines seizing rejected cards, e.g. either k = 8 and two basic numbers, or k = 4 and four basic numbers, or k = 2 and eight basic numbers.

— One chance out of 4 may be an adequate security level for deterring pirate cards periodically (every few seconds) on "official" pay TV decoders, e.g. k = 1 and two basic numbers.

In the SC and GPS mechanisms, the witness is the modular r-th power of a base g. The random number r is medium (e.g. 160 bits for the SC mechanisms, 248 to 280 bits for the GPS1 mechanisms, α +88 to α +120 bits for the GPS2 mechanisms). The witness formula is a **medium or long modular exponentiation**.

In the SC and GPS mechanisms, the complexity of the response formula is negligible in comparison with that of the witness formula. As the computation of TokenAB_1 needs no interaction with a verifier, a set of coupons (r, TokenAB_1) can be computed in advance and stored in the claimant. Additionally, if r is pseudo-randomly produced, r needs not be stored as it can be reproduced. The verification formula is a medium double modular exponentiation or a long modular exponentiation; it induces a verifier workload similar to the claimant workload. The challenge size may be optimized, but without any practical impact on the complexity of the witness and verification formulae.

➢ The SC and GPS mechanisms are attractive in systems where "coupons" can be prepared in advance for the claimant and where the interaction with a powerful verifier has to be as quick as possible. For example, a device without computational power (e.g. a tag) can quickly answer.

In the RSA_{UA} mechanisms, the verifier computes the challenge by a **short modular exponentiation** and then, the claimant computes the response by a **long modular exponentiation**. As the challenge has to be large, there is no room at all for optimization in relation with the challenge size. As the computation of TokenBA needs no interaction with the claimant, a set of coupons (r, TokenBA) can be computed in advance and stored in the verifier. Additionally, if r is pseudo-randomly produced, r needs not be stored as it can be reproduced.

➢ The RSA_{UA} mechanisms are attractive in systems where "coupons" can be "securely" prepared in advance for verifiers interacting with a powerful claimant. For example, a device without computational power (e.g. a tag) can authenticate a powerful computer.

In the RSA_{MA} mechanisms, both entities have to compute a short modular exponentiation and a long modular exponentiation. There is no possibility of a "coupon" strategy for such mechanisms.

C.2 Comparison of the authentication mechanisms

C.2.1 Symbols and abbreviated terms

The comparison uses the following measures: the storage required in the claimant, the complexity of the computations carried out by the claimant, the complexity of the computations carried out by the verifier, and the communications required between the claimant and the verifier.

NOTE If the claimant is a portable device (e.g. a smart card), then the complexity of computation and the required communication and storage may be crucial, since the processing and storage capacities of smart 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.

ChC	computational complexity of a CRT composition
ChD	computational complexity of a CRT decomposition
CM	communication required between the claimant and the verifier (CM_h when using a hash-function)
CPC	complexity of the computations carried out by the claimant
CPV	complexity of the computations carried out by the verifier
Cr	CRT coefficient
CS	storage required in the claimant
$HW(v)$	number of bits set to 1 in the binary representation of number v, e.g. $HW(65\,537 = 2^{16}+1) = 2$
M_α	computational complexity of a modular multiplication (α is the bit size of the modulus)
X_α	computational complexity of a modular square (α is the bit size of the modulus)

C.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 according with the experience, the workload due to a multiplication is approximately equal to the workload due to a reduction.

— When A and B have the same size as C, a multiplication provides a result twice longer than C.

— A 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 number n is f times longer than number p, i.e. if n and p^f have the same size, i.e. $|n| = f \times |p|$, then the ratio between a multiplication modulo n and a multiplication modulo p is approximately f^2 ($M_{|n|} \approx f^2 \times M_{|p|}$). Consequently, the value of $M_{|C|}$ is proportional to $|C|^2$.

For example, if n is twice longer than p, i.e. $|n| = 2 \times |p|$, then $M_{|n|} \approx 4 \times M_{|p|}$.

The **modular square** is defined as A^2 mod 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 twice longer than C. According to Menezes, van Oorschot and Vanstone [16], 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 this operation is as above.

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

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

The **modular exponentiation** is defined as A^B mod C. It may be performed as the right to left version of the square and multiply algorithm [13][16], i.e. $|B|-1$ modular squares and $HW(B)-1$ modular multiplications by A.

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

— If A_i is small (i.e. $|A_i| \leq 8$), then the modular multiplications due to B_i are negligible in comparison with the modular squares.

— Depending upon whether the bit size of the exponent, i.e. $\max\{|B_1|, \ldots |B_x|\}$, is either small, i.e. up to 40, or medium, i.e. {160, 240 to 280}, or large, i.e. $\{|C|, |C|+80$ to $|C|+120\}$, the modular exponentiation is either short, or medium, or long.

C.2.3 CRT technique

This clause defines the CRT technique, i.e. the use of the Chinese Remainder Theorem.

Consider two numbers $x_1 < x_2$, co-prime, but not necessarily prime, and their product denoted x.

NOTE The CRT technique accommodates any number of prime factors. Consider two distinct prime factors $p_1 < p_2$ and their product $p_1 \times p_2$, and then three distinct prime factors $p_3 < (p_1 \times p_2)$ and their product $(p_1 \times p_2) \times p_3$, and so on.

By definition, the CRT coefficient is the positive integer Cr less than x_1 so that x_1 divides $Cr \times x_2 -1$.

By definition, the CRT composition converts any pair of components, namely X_1 from $\{0, 1, \ldots x_1-1\}$ and X_2 from $\{0, 1, \ldots x_2-1\}$, into the corresponding unique number X from $\{0, 1, \ldots x-1\}$. It makes use of the two numbers x_1 and x_2 and the CRT coefficient Cr as follows.

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

The CRT composition consists of a modular multiplication modulo a factor and one multiplication of two numbers with the same size as a factor, resulting in a number with the same size as the modulus. When the two factors have the same size, e.g. $|p_1| = |p_2| = |n| / 2$, the composition complexity is denoted ChC.

$$ChC \approx 1{,}5 \times M_{|p|} \approx (3/8) \times M_{|n|}$$

Any number X from $\{0, 1, \ldots x-1\}$ is decomposed into a pair of components, namely X_1 from $\{0, 1, \ldots x_1-1\}$ and X_2 from $\{0, 1, \ldots x_2-1\}$, as follows. Decomposition reverses composition and vice versa.

$$X_1 = X \text{ mod } x_1 \quad \text{and} \quad X_2 = X \text{ mod } x_2$$

The decomposition consists of two reductions modulo a factor. When the two factors have the same size, e.g. $|p_1| = |p_2| = |n| / 2$, the decomposition complexity is denoted ChD.

$$ChD \approx M_{|p|} \approx 0{,}25 \times M_{|n|}$$

For example, the CRT technique reduces the complexity of a private RSA operation from a long modular exponentiation mod n (i.e. $(5/4) \times |n| \times M_{|n|}$) to one ChD plus two long modular exponentiations mod p_i (with exponents reduced mod p_i-1) plus one ChC (i.e. $(1+2{,}5 \times |p|+1{,}5) \times M_{|p|} = 2{,}5 \times (|p|+1) \times M_{|p|}$). As $|n| = 2 \times |p|$, $M_{|n|} \approx 4 \times M_{|p|}$ and the reduced complexity is $\approx (5/16) \times |n| \times M_{|n|}$.

C.2.4 Complexity analysis

C.2.4.1 FS

The claimant stores n and Q_1 to Q_m.		CS (bits) = $(m+1)\times\lvert n\rvert = (m+1)\times\alpha$
Witness formula	$W = r^2 \bmod^* n$	i.e. $X_{\lvert n\rvert}$
Response formula	$D = r\times\prod_{i=1}^{m} Q_i^{di} \bmod^* n$	i.e. $HW(d)\times M_{\lvert n\rvert}$
For t iterations, as $HW(d) \approx m / 2$ in average,		$CPC\ (M_\alpha) \approx t\times(2\times m+3)/4$
Verification formula	$W^* = D^2\times\prod_{i=1}^{m} G_i^{di} \bmod^* n$	i.e. $X_{\lvert n\rvert} + HW(d)\times M_{\lvert n\rvert}$
For t iterations, as $HW(d) \approx m / 2$ in average,		$CPV\ (M_\alpha) \approx t\times(2\times m+3)/4$
TokenAB_1 = either W as $\lvert n\rvert$ bits or a hash-code as $\lvert h\rvert$ bits; d as m bits; TokenAB_2 = D as $\lvert n\rvert$ bits.		
For t exchanges,	CM (bits) $\approx t\times(2\times\alpha+m)$	CM_h (bits) $\approx t\times(\alpha+\lvert h\rvert+m)$

C.2.4.2 GQ1

The claimant stores n, v and Q.		CS (bits) = $2\times\lvert n\rvert+\lvert v\rvert = 2\times\alpha+\lvert v\rvert$
Witness formula	$W = r^v \bmod n$	i.e. $(\lvert v\rvert-1)\times X_{\lvert n\rvert} + (HW(v)-1)\times M_{\lvert n\rvert}$
Response formula	$D = r\times Q^d \bmod n$	i.e. $M_{\lvert n\rvert}+(\lvert d\rvert-1)\times X_{\lvert n\rvert}+(HW(d)-1)\times M_{\lvert n\rvert}$
As d is from $\{0, 1, \ldots v-1\}$, i.e. $\lvert d\rvert = \lvert v\rvert$ and $HW(d) = \lvert v\rvert/2$,		$CPC\ (M_\alpha) \approx 2\times\lvert v\rvert+HW(v)-2{,}5$
Verification formula	$W^* = D^v\times G^d \bmod n$	i.e. $(\lvert v\rvert-1)\times X_{\lvert n\rvert}+(HW(d)+HW(v)-1)\times M_{\lvert n\rvert}$
As $HW(d) = \lvert v\rvert/2$,		$CPV\ (M_\alpha) \approx 1{,}25\times\lvert v\rvert+HW(v)-1{,}75$
TokenAB_1 = W as $\lvert n\rvert$ bits or a hash-code as $\lvert h\rvert$ bits; d as $\lvert v\rvert$ bits; TokenAB_2 = D as $\lvert n\rvert$ bits.		
	CM (bits) $\approx 2\times\alpha+\lvert v\rvert$	CM_h (bits) $\approx \alpha+\lvert h\rvert+\lvert v\rvert$

C.2.4.3 GQ2

The claimant stores p_1, p_2, Cr and $Q_{1,1}$ to $Q_{m,2}$.		CS (bits) = $(m+1{,}5)\times\lvert n\rvert = (m+1{,}5)\times\alpha$
Witness formula	$W_j = r_j^{2^{k+b}} \bmod p_j$	i.e. $2\times(k+b)\times X_{\lvert p\rvert} + ChC$
Response formula	$D_j = r_j\times\prod_{i=1}^{m} Q_{i,j}^{d_i} \bmod p_j$	i.e. $2\times((k-1)\times X_{\lvert p\rvert}+0{,}5\times k\times m\times M_{\lvert p\rvert}) + ChC$
As $ChC \approx 1{,}5\ M_{\lvert p\rvert}$,		$\approx (3+(k+(b-1)/2)\times(m+3))\times M_{\lvert p\rvert}$
As $M_{\lvert p\rvert} \approx M_{\lvert n\rvert} / 4$,		$CPC\ (M_\alpha) \approx (k+(b-1)/2)\times(m+3)/4+0{,}75$
Verification formula	$W^* = D^{2^{k+b}}\times\prod_{i=1}^{m} G_i^{d_i} \bmod n$	i.e. $(k+b)\times X_{\lvert n\rvert}$
As the basic numbers are small,		$CPV\ (M_\alpha) \approx 0{,}75\times(k+b)$
TokenAB_1 = either W as $\lvert n\rvert$ bits or a hash-code as $\lvert h\rvert$ bits; d as $k\times m$ bits; TokenAB_2 = D as $\lvert n\rvert$ bits.		
	CM (bits) $\approx 2\times\alpha+k\times m$	CM_h (bits) $\approx \alpha+\lvert h\rvert+k\times m$

C.2.4.4 SC

The claimant stores p, q, g and Q.		CS (bits) = $2\times(\lvert p\rvert+\lvert q\rvert) = 2\times(\alpha+\lvert q\rvert)$
Witness formula	$W = g^r \bmod p$	i.e. $(\lvert r\rvert-1)\times X_{\lvert p\rvert} + (HW(r)-1)\times M_{\lvert p\rvert}$
Response formula	$D = r-d\times Q \bmod q$	negligible

As $\lvert r \rvert = \lvert q \rvert$ and $HW(r) = \lvert q \rvert /2$,		$CPC\ (M_\alpha) \approx 1{,}25 \times \lvert q \rvert$
Verification formula	$W^* = g^D \times G^d \bmod p$	i.e. $(\lvert D \rvert -1) \times X_{\lvert p \rvert} + (HW(D) + HW(d) - 1) \times M_{\lvert p \rvert}$
As $HW(d) = \delta / 2$, $\lvert D \rvert = \lvert q \rvert$ and $HW(D) = \lvert q \rvert / 2$,		$CPV\ (M_\alpha) \approx 1{,}25 \times \lvert q \rvert + 0{,}5 \times \delta$

TokenAB_1 = either W as $\lvert p \rvert$ bits or a hash-code as $\lvert h \rvert$ bits; d as δ bits; TokenAB_2 = D as $\lvert q \rvert$ bits.

CM (bits) $\approx \alpha + \lvert q \rvert + \delta$ CM_h (bits) $\approx \lvert h \rvert + \lvert q \rvert + \delta$

C.2.4.5 GPS1

Without CRT, the claimant stores n and Q.		CS (bits) = $\lvert n \rvert + \lvert Q \rvert = \alpha + \sigma$
Witness formula	$W = g^r \bmod n$ where $g = 2$,	i.e. $(\lvert r \rvert -1) \times X_{\lvert n \rvert}$
Response formula	$D = r - d \times Q$	negligible
As $\lvert r \rvert = \rho = \sigma + \delta + 80$,		$CPC\ (M_\alpha) \approx (3/4) \times (\sigma + \delta) + 60$
With CRT, the claimant stores p_1, p_2, Cr and Q.		CS (bits) = $1{,}5 \times \lvert n \rvert + \lvert Q \rvert = 1{,}5 \times \alpha + \sigma$
Witness formula	$W_j = g^r \bmod p_j$ where $g = 2$,	i.e. $2 \times (\lvert r \rvert -1) \times X_{\lvert p \rvert} + ChC$
Response formula	$D = r - d \times Q$	negligible
As $\lvert r \rvert = \rho$ and $< 0{,}5 \times \lvert n \rvert$ and $ChC \approx 1{,}5 \times M_{\lvert p \rvert}$		$\approx 1{,}5 \times \rho \times M_{\lvert p \rvert}$
As $M_{\lvert p \rvert} \approx M_{\lvert n \rvert} / 4$ and $\rho = \sigma + \delta + 80$,		$CPC\ (M_\alpha) \approx (3/8) \times (\sigma + \delta) + 30$
Verification formula	$W^* = g^D \times G^d \bmod n$	i.e. $(\lvert D \rvert -1) \times X_{\lvert n \rvert} + (HW(d) - 1) \times M_{\lvert n \rvert}$
As $HW(d) = \delta / 2$ and $\lvert D \rvert = \rho$,		$CPV\ (M_\alpha) \approx 0{,}75 \times \sigma + 1{,}25 \times \delta + 60$

TokenAB_1 = either W as $\lvert n \rvert$ bits or a hash-code as $\lvert h \rvert$ bits; d as δ bits; TokenAB_2 = D as $\rho = \sigma + \delta + 80$ bits.

CM (bits) $\approx \alpha + \sigma + 2 \times \delta + 80$ CM_h (bits) $\approx \lvert h \rvert + \sigma + 2 \times \delta + 80$

C.2.4.6 GPS2

Without CRT, the claimant stores n and Q.		CS (bits) = $2 \times \lvert n \rvert = 2 \times \alpha$
Witness formula	$W_j = G^{r \times v} \bmod n$ where $G = 2$,	i.e. $(\lvert r \rvert + \lvert v \rvert) \times X_{\lvert n \rvert}$
Response formula	$D = r - d \times Q$	negligible
		$\approx (\lvert n \rvert + 2 \times \delta + 80) \times 0{,}75 \times M_{\lvert n \rvert}$
		$CPC\ (M_\alpha) \approx (3 / 4) \times (\alpha + 2 \times \delta + 80)$
With CRT, the claimant stores p_1, p_2, Cr and Q.		CS (bits) = $2{,}5 \times \lvert n \rvert = 2{,}5 \times \alpha$
Witness formula	$W_j = G^{r \times v \bmod p_j - 1} \bmod p_j$ where $G = 2$,	i.e. $2 \times (\lvert p \rvert -1) \times X_{\lvert p \rvert} + ChC$
Response formula	$D = r - d \times Q$	negligible
As $2 \times \lvert p \rvert = \lvert n \rvert$ and $ChC \approx 1{,}5 \times M_{\lvert p \rvert}$,		$\approx \lvert n \rvert \times 0{,}75 \times M_{\lvert p \rvert}$
As $M_{\lvert p \rvert} \approx M_{\lvert n \rvert} / 4$,		$CPC\ (M_\alpha) \approx (3 / 16) \times \alpha$
Verification formula	$W^* = G^{d + v \times D} \bmod n$	i.e. $(\lvert D \times v \rvert -1) \times X_{\lvert n \rvert} + (HW(D \times v) - 1) \times M_{\lvert n \rvert}$
As $\lvert D \times v \rvert = \rho$, $HW(D \times v) = \rho / 2$ and $\rho = \alpha + \delta + 80$,		$CPV\ (M_\alpha) \approx 1{,}25 \times (\alpha + \delta) + 100$

TokenAB_1 = either W as $\lvert n \rvert$ bits or a hash-code as $\lvert h \rvert$ bits; d as δ bits; TokenAB_2 = D as $\rho = \alpha + \delta + 80$ bits.

CM (bits) $\approx 2 \times \alpha + 2 \times \delta + 80$ CM_h (bits) $\approx \alpha + \lvert h \rvert + 2 \times \delta + 80$

C.2.4.7 RSA$_{UA}$

The claimant retains p_1, p_2, s_1, s_2 and *Cr*. CS (bits) = $2,5 \times |n| = 2,5 \times \alpha$

Public RSA operation: $P_X(m) = m^v \bmod n$ i.e. $(|v|-1) \times X_{|n|} + (HW(v)-1) \times M_{|n|}$

$CPV(M_\alpha) \approx 0,75 \times |v| + HW(v) - 1,75$

For example, if v is set equal to $2^{16}+1$, $CPV(M_\alpha) \approx 13$

Private RSA operation using the CRT technique $CPC(M_\alpha) \approx (5/16) \times \alpha$

Token*BA* = d as $|n|$ bits; Token*AB* = r^* as $|n|-|h|$ bits. CM (bits) $\approx 2 \times \alpha - |h|$

C.2.4.8 RSA$_{MA}$

Each entity retains p_1, p_2, s_1, s_2 and *Cr*. CS (bits) = $2,5 \times |n| = 2,5 \times \alpha$

Each entity performs a private RSA operation and a public RSA operation.

$CPC(M_\alpha) \approx CPV(M_\alpha) \approx 13 + (5/16) \times \alpha$

TokenBA_1 = d as $|n|$ bits; Token*AB* = d^* as $|n|$ bits; TokenAB_2 = rr^{**} as $0,5 \times (|n|-|h|-|ID|)$ bits.

CM (bits) $\approx 2,5 \times \alpha - 0,5 \times |h| - 0,5 \times |ID|$

C.2.4.9 Summary of the evaluations

Table C.1 summarizes the evaluations detailed in C.2.4.1 to C.2.4.8. In the FS, GQ, SC and GPS mechanisms, the required communication is either *CM* or CM_h. Such a distinction is not relevant in the RSA mechanisms. In the GPS mechanisms, the use of the CRT technique by the claimant is evaluated for *CS* and *CPC*. Table C.1 is used in C.2.5 for α = 1024, $|h|$ = 160 (e.g. RIPEMD-160 and SHA-1) and $|ID|$ = 40 with different values of the security level.

Table C.1 — Summary of the evaluations

	CS (bits)	*CPC* (M_α)	*CPV* (M_α)	*CM* (bits)	CM_h (bits)
FS	$(m+1) \times \alpha$	$t \times (2 \times m + 3)/4$	$t \times (2 \times m + 3)/4$	$t \times (2 \times \alpha + m)$	$t \times (\alpha + |h| + m)$
GQ1	$2 \times \alpha + |v|$	$2 \times |v| + HW(v) - 2,5$	$1,25 \times |v| + HW(v) - 1,75$	$2 \times \alpha + |v|$	$\alpha + |h| + |v|$
GQ2	$(m + 1,5) \times \alpha$	$(k + (b-1)/2) \times (m+3)/4 + 0,75$	$0,75 \times (k + b)$	$2 \times \alpha + k \times m$	$\alpha + |h| + k \times m$
SC	$2 \times (\alpha + |q|)$	$1,25 \times |q|$	$1,25 \times |q| + 0,5 \times \delta$	$\alpha + \delta + |q|$	$\delta + |h| + |q|$
GPS1	$\alpha + \sigma$	$0,75 \times (\sigma + \delta) + 60$	$0,75 \times \sigma + 1,25 \times \delta + 60$	$\alpha + \sigma + 2 \times \delta + 80$	$\sigma + |h| + 2 \times \delta + 80$
with CRT	$1,5 \times \alpha + \sigma$	$0,375 \times (\sigma + \delta) + 30$			
GPS2	$2 \times \alpha$	$0,75 \times \alpha + 1,5 \times \delta + 60$	$1,25 \times (\alpha + \delta) + 100$	$2 \times \alpha + 2 \times \delta + 80$	$\alpha + |h| + 2 \times \delta + 80$
with CRT	$2,5 \times \alpha$	$0,1875 \times \alpha$			
RSA$_{UA}$	$2,5 \times \alpha$	$0,3125 \times \alpha$	$0,75 \times |v| + HW(v) - 1,75$	$2 \times \alpha - |h|$	
RSA$_{MA}$	$2,5 \times \alpha$	$0,3125 \times \alpha + 0,75 \times |v| + HW(v)$		$2,5 \times \alpha - 0,5 \times |h| - 0,5 \times |ID|$	

C.2.5 Comparison for α = 1024 with different values of the security level

C.2.5.1 Comparison for α = 1024 with 2^{-8} as security level

Table C.2 compares the mechanisms for α = 1024 (medium-term security) with 2^{-8} as security level.

FS: $m = 2$ and $t = 4$

GQ1: $v = 257 = 2^8+1$, i.e. $|v| = 9$ and $HW(v) = 2$

GQ2: $b = 1$, $k = 4$ ($v = 32$) and $m = 2$

SC: $|q| = 160$ and $\delta = 8$

GPS1: $\delta = 8$, $|Q| = \sigma = 160$ ($\rho = \sigma + \delta + 80 = 248$) and $g = 2$

GPS2: $v = 257 = 2^{8}+1$, $\delta = 8$, $|Q| = \alpha = 1024$ ($\rho = \alpha + \delta + 80 = 1112$) and $G = 2$

RSA: $v = 65537 = 2^{16}+1$

Table C.2 — Comparison for $\alpha = 1024$ with 2^{-8} as security level

	CS (kbits)	*CPC* (M_{1024})	*CPV* (M_{1024})	*CM* (kbits)	CM_h (kbits)
FS	3,00	7,00	7,00	8,01	4,63
GQ1	2,01	17,50	11,50	2,01	1,17
GQ2	3,50	5,75	3,75	2,01	1,16
SC	2,31	200,00	204,00	1,16	0,32
GPS1	1,16	186,00	190,00	1,25	0,41
with CRT	1,66	93,00			
GPS2	2,00	840,00	1390,00	2,09	1,25
with CRT	2,50	192,00			
RSA$_{UA}$	2,50	320,00	13,00	1,84	
RSA$_{MA}$	2,50	334,75	334,75	2,41	

C.2.5.2 Comparison for $\alpha = 1024$ with 2^{-16} as security level

Table C.3 compares the mechanisms for $\alpha = 1024$ (medium-term security) with 2^{-16} as security level.

FS: $m = 4$ and $t = 4$

GQ1: $v = 65\,537 = 2^{16}+1$, i.e. $|v| = 17$ and $HW(v) = 2$

GQ2: $b = 1$, $k = 4$ ($v = 32$) and $m = 4$

SC: $|q| = 160$ and $\delta = 16$

GPS1: $\delta = 16$, $|Q| = \sigma = 160$ ($\rho = \sigma + \delta + 80 = 256$) and $g = 2$

GPS2: $v = 65\,537 = 2^{16}+1$, $\delta = 16$, $|Q| = \alpha = 1024$ ($\rho = \alpha + \delta + 80 = 1120$) and $G = 2$

RSA: $v = 65\,537 = 2^{16}+1$

Table C.3 — Comparison for $\alpha = 1024$ with 2^{-16} as security level

	CS (kbits)	*CPC* (M_{1024})	*CPV* (M_{1024})	*CM* (kbits)	CM_h (kbits)
FS	5,00	11,00	11,00	8,02	4,64
GQ1	2,02	33,50	21,50	2,02	1,17
GQ2	5,50	7,75	3,75	2,02	1,17
SC	2,31	200,00	208,00	1,17	0,33
GPS1	1,16	192,00	200,00	1,27	0,42
with CRT	1,66	96,00			
GPS2	2,00	852,00	1400,00	2,11	1,27
with CRT	2,50	192,00			
RSA$_{UA}$	2,50	320,00	13,00	1,84	
RSA$_{MA}$	2,50	334,75	334,75	2,41	

C.2.5.3 Comparison for $\alpha = 1024$ with 2^{-36} as security level

Table C.4 compares the mechanisms for $\alpha = 1024$ (medium-term security) with 2^{-36} as security level.

FS: $m = 6$ and $t = 6$

GQ1: $v = 2^{36}+2^{13}+1$, i.e. $|v| = 37$ and $HW(v) = 3$

GQ2: $b = 1$, $k = 6$ ($v = 128$) and $m = 6$

SC: $|q| = 160$ and $\delta = 36$

GPS1: $\delta = 36$, $|Q| = \sigma = 160$ ($\rho = \sigma + \delta + 80 = 276$) and $g = 2$

GPS2: $v = 2^{36} + 2^{13} + 1$, $\delta = 36$, $|Q| = \alpha = 1024$ ($\rho = \alpha + \delta + 80 = 1140$) and $G = 2$

RSA: $v = 65\,537 = 2^{16} + 1$

Table C.4 — Comparison for $\alpha = 1024$ with 2^{-36} as security level

	CS (kbits)	*CPC* (M_{1024})	*CPV* (M_{1024})	*CM* (kbits)	CM_h (kbits)
FS	7,00	22,50	22,50	12,04	6,97
GQ1	2,04	74,50	47,50	2,04	1,19
GQ2	7,50	14,25	5,25	2,04	1,19
SC	2,31	200,00	218,00	1,19	0,35
GPS1	1,16	207,00	225,00	1,30	0,46
with CRT	1,66	103,50			
GPS2	2,00	882,00	1425,00	2,15	1,30
with CRT	2,50	192,00			
RSA_{UA}	2,50	320,00	13,00	1,84	
RSA_{MA}	2,50	334,75	334,75	2,41	

Annex D
(informative)

Numerical examples

D.1 FS mechanism

D.1.1 Key production

D.1.1.1 Asymmetric key pair (v = 2, the Rabin scheme)

The bit size is 512 for each prime factor and α = 1024 for the modulus. As the verification exponent is v = 2, one prime factor is congruent to 3 mod 8 and the other one to 7 mod 8.

p_1 =
```
A220780E 0E0717BE D41CD957 418C6215 D25CAE16 E4F6013F 7EFC69EF AB025A1E
42848EB6 9E0983C5 389B4037 CB7B6A2C EEF2134D CBA06201 376C39EA 33D297CB
```

p_2 =
```
D4610C36 12718EF3 EAC804E2 6C2751A0 EA8A8FB2 522499DA 44105CFC 19C7A94F
06784168 DEF906A9 7AEBD153 6E3E32A4 61933F30 33006D50 F5A7B799 4FAD11FF
```

n =
```
86805974 E5195F47 C8DD033B 658151DE EF39BF57 969645CD A5610766 64D121ED
6C08EC5F 7E6DC1DF C97CD4C8 B154D5FD 21CC06FF DC2C9E44 6789AF0F 916B2B28
D75263E4 47D7FD58 8E46AFE8 99F6A36D 60DFDDA9 48066026 BE7982D8 17777F5B
30EE8A40 0C3B7508 278FD600 E7770A51 43C7DB91 CC16CE01 9DB51535 D408AE35
```

u =
```
10D00B2E 9CA32BE8 F91BA067 6CB02A3B DDE737EA F2D2C8B9 B4AC20EC CC9A243D
AD811D8B EFCDB83B F92F9A99 162A9ABF A43980DF FB8593C8 8CF135E1 F22D6564
EC1A1BF4 04EBEAD4 B9EC3A35 DD885DF6 D47F13FC 021D78A1 9F6D977D 8A55AF7D
BCFE3744 11E71D53 2E81188E B5B7ADAF FE685122 79AEBFD5 EE142476 4A11208D
```

D.1.1.2 Identification data and asymmetric pairs of numbers

The pair multiplicity parameter is m = 8. Each part of the identification data results from appending a 16-bit suffix to the bit string representing "Alex Ample".

Id_1 = `416C 6578 2041 6D70 6C65 0001` Id_2 = `416C 6578 2041 6D70 6C65 0002`
Id_3 = `416C 6578 2041 6D70 6C65 0003` Id_4 = `416C 6578 2041 6D70 6C65 0004`
Id_5 = `416C 6578 2041 6D70 6C65 0005` Id_6 = `416C 6578 2041 6D70 6C65 0006`
Id_7 = `416C 6578 2041 6D70 6C65 0007` Id_8 = `416C 6578 2041 6D70 6C65 0008`

The format mechanism makes use of SHA-1, i.e. the third hash-function specified in ISO/IEC 10118-3[25].

G_1 =
```
0004C24F 6F5F4A75 C3787AF2 8F50FF3B 5E3404D2 0DF52FF4 E86E132B CBF9AD8B
E5BE0CF3 C42FCD80 3AA602D3 22E1BFE3 3F08737A A47CB9AC 65870280 59E2B467
C4CED23F 7EE67A52 DB93E947 60E71AC0 1EE93894 A6B7E592 456534D6 CCD2FE2D
1AB9AA07 CDEB74FE FB12C73B 3D67898F 3F33803F C0A81C1C C64312DF 05ECF8DE
```

G_2 =
```
56BA5901 0415F74E 81B6C97F 04645BF9 6A35F1B1 C97AB20B 80EF22D7 E5DE2639
F36408DE 6C54B4EB B2B6AA41 4F18F869 4E7BFCE1 EAD07953 D3CC123C D0F15C30
64A7FEA2 93A5E2C9 3643242E D87B8E24 A8A85B84 A7D8B33A D325D60C 8B017C3A
F618DD78 8B51A8D4 AAD001BF 06D760AD DFA2663B 4DB850E7 321662CE 8F6049BC
```

G_3 =
```
12C93D02 41469023 ED09FDCC D558AA55 16055238 07DCF856 0D33A12E 0987359B
36053658 DF870009 3E0FEE03 1CCA1D25 454D62B3 3E2F00C6 51209F8C 02CD5F91
0D7D5872 3B912DE9 F26C8535 8872E424 880089EA A73EF73C 98B72346 F0794B3B
6ADFD119 D5201751 7827BB0C 6430D6A8 5D80B05E D0B28058 C8A98BDA 7F733A5E
```

G_4 =
```
5DE7CCDC AAD76847 603D036A 08B5FF85 B1138616 5AB8C615 918F5193 8F85A03F
C7E08EB7 01C1C8C9 986E8018 80BCB6B4 725380C6 962B780B 90A2AD09 9105C87A
2EE04035 ACA54A4C 764F0534 90ACCED1 3409B81B 74AD6906 45800ADA 56626EB8
C288BFC8 9D6A950A C45887D3 612B271C 80A5D6BA 3EB71986 27CCCFBB 14B257BC
```

G_5 =
```
07F5EA50 3C9022AF B22701A6 2E649D06 008AFE93 8EA136D7 1AFD6FBC 90B8EF18
FA8FE507 CD81B4BD DEC57637 C2C24DEC BB22A71F D7FE9229 7C807EDB 5A53FC35
61E40492 A24C4C9B 583CCEC0 ED475CCD 1E533241 BA93BB5A 8B1FA011 7A75F777
07B824BB B93FF810 77481989 2D248603 53891E9F 1466258E 6F7D6F51 E2F285BC
```

G_6 =
```
4CB08F35 99AC2CAB DCFF28C7 BBB42166 3FED4CB8 ADCC5B6E 48805AF2 33254C81
709677D7 64710108 A4446CF6 A8749A4D 61A7DB69 DED3074B E7B5B3B6 10CA526C
8556C54B 5E5E4751 8477C889 0D9F39F8 06B0FAD2 00AAC774 F3872D82 14BB6E26
1AFD4DBF F21C6165 49046374 7FE1AA53 A4DAFF81 1DB510A7 5AD7BAC2 64F23DBC
```

G_7 =
```
06CC5160 0D68CE69 1630AB55 17A73EF3 D1D5A685 86B3519B 34AD1D8C 5DE5AA65
C07986E1 DE78F4B9 DB6D2FFD B99381E0 3B9FC118 E5A6BA2E 332D2DB3 904A0382
0EAE7D0C 6255E089 7B060CD8 52FF8758 C98FB46F C6ECE83E 67469EB5 62A4D44C
4029744A DC0B813A 50D8CBD1 CFE51490 FB0BB736 E69D8CD2 A3C02B4B 724DC9BC
```

G_8 =
```
0ED43F5B 6872EF9B B42FFD7C 90282C3E 7EA28C45 67ABA2D3 6DBCC16A 2A572AB7
596FA852 8FCB4324 D2BAB32D 8ECB5E8E 43CCFEA0 C3824AA1 EB8D0064 07B7F980
428CDF44 F8A4B00E DB74A5D6 E46ADB80 D5C699BF DCAED10F CC7F0233 F6A4E815
5359D003 7007600F 91082261 D0090802 AA0D06BB 800ADCF9 7BE287A3 4CB1C55E
```

The private keys are as follows.

Q_1 =
```
1ED15C26 52F61C4C 37D4B558 C1DAB730 B248783C 6F7AF27E 55637614 A95CAF77
BEB2B52F 52B62791 446F8400 16100B21 2BCDF5A9 AFEF74FA 83188DD3 1032721B
8ACD3DD1 702C716F 38153298 20B66048 B828C0B8 3A2D15B5 D6D276B5 41B540AC
FD41FD5C 655C3A74 67B73DB9 94DBD0AD 30D4DB7E 51D64091 F859AD28 AC98E8AA
```

Q_2 =
```
009D94EA 30D5F13A 7E5917F9 21CCC91C DA18A2F8 CB368627 16E456F0 128AAECD
749394EE 79E0623B D4027C6B F4B51D3E D0DB8804 77CA7FA9 05180ED0 8B15CFDC
71756866 8642019A 10C11009 5917E043 808307B3 8D2E9BCA 41D89D21 B7125C15
E8AA839D 10B6D84C 03F31842 B174086D FE65E984 E2A924EB 1756C4CB FD49B342
```

Q_3 =
```
147C1279 C01B355F 6B295CF1 300D20D7 8381939B 1FE54B27 7356E748 A60CC211
FDAF8E92 38EC0C3C 0B13B47C 124F217B 220C5025 F5D5BC09 92A575A5 DDBE23F1
E060A199 4AE8875A 45C81CE0 B325B800 530A0433 569689FA 66CEA72D 5B42F099
BC5ED4F2 798C847D E00603DC 379619E5 28FE742E E334AFFE F8F9F433 A2B9E86B
```

Q_4 =
```
03B6941D 904B00AF 1614F88D DC3D5879 A4402420 48855251 98761996 7B3A681D
F8393CF4 9180C8E1 9C2B115F 31DE83AB 84741615 DE1CF7B1 C32BC0E5 838DCEC3
30CDF868 FB570D6C 022F8539 14FB078F 2C069A4D 7F2B6E67 25A74AB3 112CB146
4C5C12FD F51F296E 502C3399 86148FE7 69951D21 9AAEED23 6940F665 5E821794
```

Q_5 =
```
27BA1193 CA623C79 7CCF0560 184BDBEA 57DC069C 441E0B46 9B647419 87E5AA36
57619FAD B8F176E5 2D6A1D4F 26A0904D FCFF99D4 3453EB0A F3CEEA61 45B7C087
EEF9DC15 4B9933D3 98B0829E 77F8F55C 17F2EC82 0931E239 FB4D246C 84689D7D
A5614867 E66E0754 0A26818E B52A1F24 103CCF90 E87B7E50 0C36716A AA1F9EF6
```

Q_6 =
```
194DBD80 0BB6FF60 FA77CE90 E9BD233E CD99EDE7 042E414D E9EB4E22 0B4B0046
51C28CD0 78243340 87376670 5A8CB70B 6CB4A214 01B43D37 12A5CE3B A0B45B15
076D2A53 2C6B449C 1ACFADDD E6A92279 67D2519C 81351D1B 9E8C4286 DBB60650
20B5C202 8CF306E3 72138968 7C5B01B1 2137C0F7 5C02C696 0715BB3D E07F14BC
```

Q_7 =
```
07F513BA 8A0A3280 0AFC00AB 850BCFF8 FA532993 018A6608 4301BB69 FEAEC7FC
F7AE869A F9236F6D 152FCA38 CB97291C 2D2BE82A A760E978 273DF66F 6E57D012
20BE8C90 9AF83ABD A40347A3 7C6EC83C 6B1A40A6 24BE324F 1432EB7E 22897214
5C7370FC 59A2AB1F A7554C85 CCCAEF9D 5707F4B1 0DF2C349 2E10726B 5107C051
```

Q_8 =
```
2785555C C6FDCB2C 2CA944A1 4179F7C2 B2BBD59D 1903AB62 B7ED8AB8 A8D49589
F9A644AE B1A755E1 16CEDBC0 6931D163 31EB16DF EFCFA46B DE8AABA9 9BB994FF
B77AD756 7292B51B C08526B8 F32FCE66 F2D7D1BA 55F7850B 4DD6355A 9CB6C88D
17999B0B 01BDE24F C7461F58 08E4F9F3 F1567870 15322712 33B49F97 695A582E
```

D.1.2 Unilateral authentication exchange

As m = 8, the bit length is δ = 8 for the challenges. The exchange multiplicity parameter is t = 3.

Iteration 1

Step 1

```
r =   46730924 DDAE318D 6D1060BF BC5508A4 1E52C997 C3A752E1 0B511436 EF884689
      60AB25AF D8A75D74 E4B0DADD 1F5A9AFB 26556C5F 9EA22A95 87BF849C 462738AA
      D1C144E8 61293533 5914F5C5 2A8D2323 6716C336 A4E06AE3 3DDE5A34 DC8AA982
      74498C4A 6F7F6E89 83D7A2BA D51BCAF1 4629891F 6113F7DE A08E4BF2 60EDAF55

W =   1ADED7E0 6F4DE303 1E04694E 7045363D 1D62A241 4925D5BD 6A54D352 43B1C9CE
      A9ADC1BC 8968D4F7 034531F1 5C717E16 4F7F9F9F 779A439F A23EA1C2 7A831B93
      439DB041 C6AEFE7E 031B2FA1 FB2390E8 89EAE68F 699D5D27 4505EAB7 95D1FFB9
      BC7DC6CA 6C38BCBB 4651CECD 90778FA4 E91C9D65 42BFD336 108EFE8D 6AB8FA0B
```

Step 3

$d_1, d_2, \ldots, d_8$ = 0, 0, 0, 0, 1, 1, 0, 0

Step 5

```
D =   37D87F34 EA4CD0E2 A825E891 1EAC4F15 C7969E59 2C6741E9 A9142922 2817650E
      21D13151 7D768A55 7AC7A8CA BE50D66B D0BA0A09 7338B3F0 A1CD1236 1B9F9945
      951BD90C D9CB314A D3CC8F65 ACD232FA F7152A4B 68B97B7A 7C230A7C 8099E938
      62A3435E AD1F4BA6 9A6C00C3 919B5342 45E0F06F 604D6112 C7EABE7C 3D2C6D39
```

Iteration 2

Step 1

```
r =   546E4A31 5718EA7E 00779BBA DB667B34 7DC1C1B4 992AD37C 2B687927 5283389F
      B6AC25F9 55E5CB70 647EBCB4 0F9D86BF EABF7308 DB6F3B12 DBE1C73F AA5EDC9A
      988F6DE8 BCE672D2 1CA00EED 53E76E72 15805F9D 52BF401C 8B6B28BA CA10FEF3
      498118AB B89390E3 1A685343 4F99D136 EB3016E5 7C86FEAE 58A83068 033C508C

W =   3565606D 94F1FEEC A61DC570 D99193B8 01506F0F 8E1EFF0D 8A6F488E 2E1434CD
      B3D91345 F3A5D51A ED1479BA 04D2DBCC 064AFF94 058D4E07 65E4327F 2C1EB0DE
      13C6DA80 D47A6DB5 27BA686C 010A93BB 426CEAAA 6A73CF42 1F78572B 5CE999AF
      9D170BDA B008F088 CD379265 6F013A98 290788E3 ABD9A171 FCC9E01A 3D304E49
```

Step 3

$d_1, d_2, \ldots, d_8$ = 1, 0, 0, 0, 1, 1, 0, 1

Step 5

```
D =   1FA64318 C842715B 5A1404E2 445767E4 55EB9344 6EC9F311 A770B965 F34047CB
      A69F7D42 E95CC9F2 AE54716F B97B765B 7CE69B8B 05795C62 EBCF6A5D AA80323F
      7E1880BB B7154F60 BB6F5E2A F064D759 41458EED 951BE96C BA9E1E0E F07ACD22
      7B311649 8E98C7C1 EE6AFFF5 5887C1C7 37CCCADF 37DDBCAF 5B59555E FA1D35DF
```

Iteration 3

Step 1

```
r =   2D667AD3 3F6615A2 26647FB1 889EAE85 203792B8 68DFA869 2DA3B9AA 87B14D9E
      52BF5637 0065BE27 775E37E0 9896FF8F 0FB8F162 ACD7599A 18F8893A 23386E0D
      E22357B2 C1A455AE 1A809F8C 1B33A9DF CE8A4E48 2C7B2A1C A96F9F0C AC33EC1E
      27FB4368 04264F76 E1B68C3C BF37CB99 A865B9E1 23E3AA7D AE73540E 5DB834FA

W =   41068CBD 2F2CCA28 95E935BB 3D3F228A 3D43B2F1 61B1DA7D A62EE180 B0B3D930
      C87E1F5C 88F8CEA5 F6A81C5A A2A25689 AA7D2C50 505B8689 49F41FF4 A71377C8
      81E01CC4 9CCA612E 0E43BD07 D5622238 7494A0A6 3CCD433D 5782636B AB7DBB36
      394F3FB5 30FEF9DE FDC72B2C D1AE4179 6B6C7AFD 2AA114A2 966E7BAB 127A458E
```

Step 3

$d_1, d_2, \dots, d_8$ = `0, 0, 1, 0, 0, 0, 0, 0`

Step 5

```
D =   0FDD219F 57F5FCAB 2DAC9364 3EA429D7 ECEE6833 6BD1793B 0F72FBD0 8DA133F6
      5F0B46A0 F9A1CE1E 79F4F103 F37B19D8 9FF68054 3DFBCF33 01A5CB00 234FEE71
      7352006E 9555977D 1F724218 74E264B4 9179435E 4DF6CA94 48EE9529 15900FB9
      D94D132E 833103A0 50A22A5B ACBA97FE 00D21B99 BC171CFB 06523911 B3835D20
```

D.2 GQ1 mechanism

D.2.1 Key production

D.2.1.1 Asymmetric key pair (v is an odd prime, the RSA scheme)

The bit size is 512 for each prime factor and α = 1024 for the modulus. The verification exponent is $v = 2^{16}+1$ (a prime number = 65537 in decimal = '10001' in hexadecimal); it divides neither p_1-1 nor p_2-1.

```
p1 =  D716BEA5 9AC10B1C B5CFD57D 0204C349 52240F8E 9BDD319D 4F5ADD0C D9478B7E
      AF96558F 85A74A20 B6664136 DD589F35 CFF94287 1B3298BE 40ED2C86 899186E9

p2 =  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

u =   18384CCC 9C9A4CE6 61B06616 EF1A5CD4 436C9AD2 7A081D14 8E7ACD55 ED240B1D
      AFCD2E8E 4676EA1B 259F02C3 79831FD7 F87BEB20 79EA1DF9 283BEEB5 83CBFA4B
      5CAEF744 597550EB F85AE3D0 4CFC6F9F 26E035F0 E317D21F F3A241C7 92132BEC
      633560E2 C9B5A3E5 88104BE0 61535C3E E4EC7220 838B3E01 53277B9F C5EA5137
```

D.2.1.2 Identification data and asymmetric pair of numbers

The identification data consists of the bit string representing "Alex Ample".

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

The format mechanism makes use of SHA-1, i.e. the third hash-function specified in ISO/IEC 10118-3 [25].

```
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 =   24B9559A 80BD4D89 B9802A14 36DA3BDF 8DDF8DC3 993DEB1F A7EE0B4D B9F2EFFC
      3003722C 9217CE8F BFEB962A 39B32DED F02C25CF 02702195 7A103024 15A7D59A
      133A2B06 840B1DCA 10445287 FF875EAD DFEAFC8B 12B7C7E3 E05375C5 4D2369B7
      9DFCEC0F 9235ADB3 16427D66 70D9422D 39C4F32B E1A406B5 E26736E1 F68E3682
```

D.2.2 Unilateral authentication exchange

As $v = 2^{16}+1$, the bit length is $\delta = 16$ for the challenges.

Step 1

```
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 =

```
      411F7E73 D995AC63 BACAE1F2 F1BF8D03 4886E36C 5825BC31 BDB761E8 567B6762
      9947B41C 56A2EC07 8D02B880 76451F4F 991892D2 2F291949 F6F462B5 9098D627
      F473111C FD260FFD 4428DD0C 3D270B82 F09E51C3 CF9065BD 744F708C 5D5C08B8
      39336472 208415CC 72EBF75D 5A339134 C21E68AD 7AE057AB 8B25B776 CFCE18D1
```

Step 3

d = `D783`

Step 5

D =

```
      3A2B6A07 3CAEEF40 1E1792E7 D67B5F76 CB7B900B 592B344E E7D8E641 FD78FA21
      3DD31D25 FF772479 037A53E8 D82A357E 43D02FEA B768685F 03E4654B 46CC2610
      B7710A9F A4E6DE30 24F65AE6 54BB445B ABEE957B AAB861EF CF74F05C E577F407
      8DA447CC 387ECD96 A67B53E9 11D411F0 32782455 081F5AF0 AB7D6777 B3841E0C
```

D.3 GQ2 mechanism (first example: $b > 1$)

D.3.1 Key production

The bit size is 512 for each prime factor and $\alpha = 1024$ for the modulus.

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
```

Cr =

```
      66516ED5 D013D71D E282A841 0EF960FA F7D7F41A 57B60742 92BD1146 4E508BD4
      5747413F 8E92110A 958220B7 37555D9A C474DF74 00830563 89685EEB CF94C8E5
```

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
```

NOTE The above set of numbers is also used in D.8.1 (RSA_{UA}).

The security parameter is $k = 8$. As $b_1 = 4$ and $b_2 = 2$, the adaptation parameter is $b = 4$. Consequently, the verification exponent is $v = 2^{12} = 4096$.

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 two basic numbers ($m = 2$), namely, $g_1 = 2$ and $g_2 = 3$.

$Q_{1,1}$ =

```
      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
```

D.3.2 Unilateral authentication exchange

As k = 8 and m = 2, the bit length is δ = 16 for the challenges.

Step 1

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
```

Step 3

d = `948C`, i.e. d_1 = `94`, i.e. `10010100` and d_2 = `8C`, i.e. `10001100`

Step 5

D =
```
28FCBD3D 0BACC08E 614A7AB7 F4913472 D3CD8716 0961639A 94A06ED1 A5B3289F
1C635101 5ED72C6E C2B653F5 CB09E93C 88478733 FAABFF35 D2D05E35 A895EA37
6B5998EF 1E24B090 9A45E0E8 3BC01302 CBAD5D0F 26E21179 29B15DD2 E14F8EC5
18E201FF B03FFE05 9D53B5DA 5CAC04BC DE446981 E4995C3A 75E831BD 8D86D325
```

D.4 GQ2 mechanism (second example: b = 1)

D.4.1 Key production

The bit size is 512 for each prime factor and α = 1024 for the modulus. As b = 1, each prime factor is congruent to 3 mod 4.

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
```

Cr =
```
D09B24CA 87A42315 E6EBA6BE E6AD15D3 A3F45344 5D5D0824 FDDCAAEE F2544B7F
89316E3B 9E532F26 C3723E00 C911A4C2 E4D03F6C ECE82FA3 B9929B16 4FFE0970
```

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
```

The security parameter is k = 8. As $b_1 = b_2 = 1$, the adaptation parameter is b = 1. Consequently, the verification exponent is $v = 2^9 = 512$.

u_1 =
```
0638AAC8 987C68F6 0E9057D8 BAA4E02D F3B78D0B EABCED28 84EAAE43 9AE20AA5
3C8EF2ED BCFADB46 31AA312B 86F9F60A CE8ADCAA 8173CB31 474F71B6 C73FBF8B
```

u_2 =
```
4EEEC912 EDC8425E ABE2D58F 08E77604 DCBE15E0 2DDCC70C 4747B501 39B6FB64
7E2FE22D 5F7D8D4A 6C75625A 42445562 173C4A3A A6984A38 9D14833D D379051F
```

There are two basic numbers (m = 2), namely, g_1 = 2 and g_2 = 3.

$Q_{1,1}$ =
```
DEFF24F3 D063F874 51C7A580 FAEB9C6E 44C9A3E9 2819B83D A90A40EF C598853A
4FE073F4 BC348AA0 99EB45AF D7799C55 D28B01CE F74AA99C 4F64333F 0D92E928
```

$Q_{1,2}$ =
```
75801C91 DA2595A8 C790692B E5406F07 0DC6902B 431EF20D D464FBAC 4E11F8F8
21D5A934 DBAD2E4A D9A3F4E1 2CB5E0EC 0A5DD49E 04BD19DB E1838D23 F37DD3DF
```

$Q_{2,1}$ = 8DF2CC77 8B17D817 D02B9CC8 9802D8F1 04DC12C8 8089A937 3B82D665 EE3B8E14
6C964F32 41B43A20 3DBFC264 1E1F45FE 2172A0DE 2EA8875F 8EC5A514 89472CA4

$Q_{2,2}$ = 31B059C5 56422DEE 1CEDBE9E 9A5B82C0 26DC8586 47F8ECF7 FA3032B9 28389B33
1A9825CF CC280CC1 8B671507 4F2EE897 0F8C692C 6E62796F 369DA6A7 A0188A85

D.4.2 Unilateral authentication exchange

As k = 8 and m = 2, the bit length is 16 for the challenges.

Step 1

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 = 3B9C8250 E07F13D6 4A8E8F5D 8315B2F2 3368300D 54B7EC4D 66F5948C 96DE6AF9
8C2C6F7D 05F3B3D4 9E9255A2 339C2E9D 29A04F68 B007D234 483B14EA 8BF6F6FC
0FCC96C7 DFAEE6EE FC718DF8 228526F5 D8575717 EA9D726E DE91310D 2E372838
7B533EF3 667AD83F 910F153C 5CD69D89 90A3F5F2 2C532C48 F6C3D682 7C755B49

Step 3

d = 948C, i.e. d_1 = 94, i.e. 10010100 and d_2 = 8C, i.e. 10001100

Step 5

D = D94EC0C1 8D456808 2BCC6F3B B0DED48F 24466A98 4F6F90B9 9C54763F DC1774E9
56F8EDF2 F68825D1 19A2B442 5F310582 CDA7EAEE 6D782E7C 9D45711C D67509E4
46651E15 22D61A16 564EF2B8 DA46A1E7 88FF64BD ACC31045 FE98DDE9 2F56AD5B
68F56F4B 3286A34E 26ED710D 1142408C E67C4578 29C3A9DF D8F72CA1 379385AD

D.5 SC mechanism

D.5.1 Key production

The bit size is α = 1024 for the modulus p (a prime number) and 160 for the prime number q (a prime factor of p–1, selected so that a copy of q is embedded within p).

q = CB0EBC3A CCB15C36 896F67F0 703E7C69 AFC4C24B

p = EA9B8F92 26D7B2F6 729122EF 53CE81E2 567ACF40 A7DB660E BA5E4DAF CB0EBC3A
CCB15C36 896F67F0 703E7C69 AFC4C24B 221A8968 5CDCFB3E 086D8F95 702CBFC5
8E4170A2 E10DF7B5 2BF8F015 C5A689CA 48DF291B E796C443 F5E7AD19 8C159F0A
BA9D962E 60D34840 77B5993E 48BBC3ED FEF5F54C ACCDE46E 69A3F1F6 1AE08AF9

g = 26324F69 934E6733 C66367A5 AF5A08D8 455A5125 29882857 B20083E8 F72420A9
1F16A377 6DC612FF E652A2DD 05D51441 5F52C591 E8AA3127 8309CE2B CA9E5B73
5E8CC526 0DC1608D 91F32A8D 31265ADC F2F2FF5F A4A786EF 25086BDB 061355CD
96EA33F6 429AEF56 BC0C0ABA DB1EC3E0 B1140687 D60678C6 205C7F6D 6A236F87

Q = 87146299 068B4B13 017364B7 E7DDA29E CDA5547E

G = 819B36E6 62DDC4AF 146DCF3A F888D61B 560EA5EA 8BB368F7 0E822E95 EF5E45C6
68B98732 725D29DC 21BF1394 29D95DE2 98A6D595 9A7188C3 AB4B5D6D 20CA1D9E
D6BC4D7A D23A4E3B 48CBE4AC DA28D927 922C85FF DB7E1F59 71A17DD5 DC68725C
32CF50F0 BE5D8A73 F93BF113 1C55BF51 35B314BE 5067FD31 9867041D 4C96E5CF

D.5.2 Unilateral authentication exchange

The bit length is δ = 40 for the challenges.

Step 1

r =
```
87146299 068B4B13 017364B7 E7DDA29E CDA5547A
```

W =
```
397AD6F9 B435B01B 4C43A2D1 008DDADE 1A086C2F 0EA25134 FF5A8653 A374DFBF
47F1A543 FBB58232 0357CCE1 33AEB861 6AEBD4B7 65DEA271 0DFF3A09 7C40602B
7E719499 0E9C7717 0CE73286 930E9E27 F8053B28 D2C80FD2 EC529839 27F34F46
BB9842B0 BD9C6405 1B2C58D8 C5CDCC50 69C4A430 D0F93CD0 6F2F75F3 298684F6
```

Step 3

d =
```
      A2 CDA554A6
```

Step 5

D =
```
354BF25C 5F0E8CCA F2AEA2B9 7716A2D5 CB8CEB7E
```

D.6 GPS1 mechanism

D.6.1 Key production

The bit size is α = 1024 for the modulus and 160 for the private key (σ = 160). The base of the discrete logarithm is g = 2.

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
```

Q =
```
8944FE65 F644C82D 2F60D423 AD3B3C21 AE3013BA
```

G =
```
84475410 6462493C D64828E4 91D70FCC 687A0A09 C7CEF778 B968DF15 4BF34A03
388D3D74 D6931CB3 072E4D6B 21D343BE 995FB060 6114BB9E A6C0E32C 54EDD73F
92F1129B 8C4BEE86 3CFAC094 83ACFEA1 81083C9B 624E9A50 7D2778E4 B651ED85
34F1730C 2A52E5D2 345C9E09 49CE84C2 A08C2A22 6FA73ABF 92EE3CA0 FEE2A7AA
```

D.6.2 Authentication exchange

The bit length is δ = 40 for the challenges and consequently, ρ = 160 + 80 + 40 = 280 for the fresh strings of random bits.

Step 1

r =
```
  FBE252 8A24B873 01E132D0 346C29E8 8552F568 DC6FA49A 44232FF4 05F0DC65
318FFFF9
```

W =
```
7D0081A7 5C9BF2FC 78679919 EB94A740 2573FC8B 06BD1944 3FD54077 398F5252
0F3D0107 32AB7537 456354A3 97A8AAE1 011C5EF3 9B722369 3C2AF56F B7B8EFD1
5186FD48 10435B62 30765083 1AFA4782 6B57A2FA D2299D1A 79D64B3F 32730174
EFFB5F45 D33BD8CC E56EB4AB 6B223728 0F3E4069 043F9CAF 93C71632 CDFE23B5
```

Step 3

d =
```
      C0 6AF0CD17
```

Step 5

D =
```
  FBE252 8A24B873 01E13269 0755AD72 8542F6C8 BF5BDC3A E4F58756 10B57C24
89124843
```

D.7 GPS2 mechanism

D.7.1 Key production

The bit size is 512 for each prime factor and α = 1024 for the modulus. The verification exponent is $v = 2^{16}+1$ (a prime number). The private key is the RSA signature exponent. The public key is G = 2.

p_1 =
```
AD521B6A B4DF5E3F F9C3614B 7083CE55 DFA50D94 3F1260C5 82C72270 1A164BC2
F3B8952D 2D5442B4 497D27DB 235533F4 8751CC88 B9D7C534 BB9F2CEF B5C68125
```

p_2 =
```
EB4369AD 61C9161B 8DAF355F 8CDAC18D 41288DB0 8798E949 AE6D2B89 BD52ACEB
D2E0E873 2685DB36 DCADFF53 65EEA0A6 FE44C5C6 D03965AC 346F5C69 FB2E94ED
```

n =
```
9F480334 30E5EE18 E8E13560 5A91A61E 4DCC54EC C9E4F8D5 460F1828 A220DE18
4A0AD8BD E132CFC3 473A7528 9EDCDA3D 475FE45C 437DF74E A16B79F6 4F7CB0F9
E10ED6B9 30D89B76 DE10AB56 C683D315 DBC0061A BD4DBE88 A19ED2FC A442D792
296C3BF1 8BBD2FBD 40D4E085 222126DF 5994BFDC 870DEAC1 3A82BD79 8714F341
```

Q =
```
063B1F32 F6B9BBF1 7A04BB5D 905573C9 EA31B4DD C97D1D55 DC868123 AFC9F8DE
3AE1473E D0553846 F39DA011 2D7BC6C6 D068BA2A 78D26FEF 01C60E50 9A25EED7
5BA07156 CBCB37F0 6C184587 3DD86913 4E386701 0543B02A 43014BC0 430ABFDB
25D5C82F ADAFC295 F5488FF1 490C1968 815DC762 E54FAEE2 E38EBDAE 44265D75
```

D.7.2 Authentication exchange

The bit length is δ = 16 for the challenges and consequently, ρ = 1024 + 80 + 16 = 1120 for the fresh strings of random bits.

Step 1

r =
```
13FB6725 909D85BD 36810665 3E3A45E9 1163523C 33897DD8 56DE0E74 5628E712
5B7FF356 BE8B1138 750C7A66 47F892C4 D6789B19 A72D10DE 324C43A4 F8F63439
DD40A3B3 897F69C3 28747B9F AFB8D4FE 8B7AE2D2 5827ECD7 0EC2A9E0 F2C5D7EB
AE705661 3B2157E9 CAD1FD5B E80504C3 66239446 BDC0055C 11A98907 555B4FE8
183027AA 20B48E86 CF27645B
```

W =
```
9A66D066 D2A97254 6B57AA77 7EDC33A7 0D35312C 8B3F0A89 955131DB C29408EE
1C4416DB 17C69105 82325953 C89B1DD8 310D7351 A4487E02 FA870E59 FCA7E71A
639891B2 04EF8373 E901B7F8 0FB40C32 840574EF 09FAECF2 A947DA82 C53BAA14
6FDFA3E8 824D15BD E5110456 7400464A 4E34F1FF B42878A2 D0236491 94ABBB9F
```

Step 3

d =
```
6B26
```

Step 5

D =
```
13FB6725 909D85BD 368103C9 9B695738 D0437E64 1C8592A1 32C97998 CC2F5AEF
27AAB56B CA99FB72 F20CA6CD 61C0BE5E 4B4C98BE 9BE0204E B5D7A906 439AD16B
F93B207B 7E1D995E 6BFF045C 0688BAEE 5AEF17F4 277E13EA 6CB7FF51 C3592091
9BC28619 BE46BD5A 5908EA03 EF9D6017 27B8047F 00D0CE03 2B3F1571 EF196161
C0435CDF B03C902C A8659DFD
```

D.8 RSA$_{UA}$ mechanism

D.8.1 Key production

This example makes use of the prime factors (512 bits) and the modulus (α = 1024) of D.3.1, GQ2.

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
```

The public exponent is $v = 2^{16}+1$ (a prime number = 65537 in decimal = '10001' in hexadecimal); it divides neither p_1–1, nor p_2–1. The public operation is "raising a positive integer, less than n, to the v-th power modulo n".

x =
```
34DD74D5 EE600302 21EC3EC7 3774B81D 4E5C6525 5B3B49CD 1E55967A A064B8C2
8C19E16E D53FFB6B 09499768 6197FBA1 29ABC84E D47B2AA3 441BFE21 6E37599E
EA8C0090 85964B87 52FF19E0 E7D13C23 90456167 643A2E06 EB3508EA B12B8D27
966FCE60 1F5781AC 3554C703 E52A039B EEE45281 34A43CF8 467624AC CD077231
```

D.8.2 Unilateral authentication exchange

The bit length is ρ = 384 for the fresh strings of random bits. This example makes use of SHA-1, i.e. the third hash-function specified in ISO/IEC 10118-3 [25].

Step 1

r =
```
EBAA9CF6 EA04B882 D312697E DC65E40E 845C85AE 0318F8CE 75A5B650 37488370
3E85216F 69C614DD CEF89D68 22BE09BE
```

H =
```
FE4D80CD 009AFD8A 7A40B1EE D1CFC0D1 0D29E74E
```

d =
```
70C81D29 7A526847 26F67D41 A6EE4384 C8383E9A A1D5F30D FF2A0C20 82D5C335
424C37EA F2729019 91E88A87 0A3D89CC F51B15D0 84435C76 1C609D50 411DB058
83769C2B 542640AA C5EAE64B ACECC70F 4466C331 D52BAD21 C48BA126 A35B06F4
AE0D9B35 8CFB2167 8EEF8FD2 52CBE352 B0057E17 47AEB0B3 B5F3FAEA 49FC2FD0
```

Step 3

$S_A(d)$ =
```
EBAA9CF6 EA04B882 D312697E DC65E40E 845C85AE 0318F8CE 75A5B650 37488370
3E85216F 69C614DD CEF89D68 22BE09BE FE4D80CD 009AFD8A 7A40B1EE D1CFC0D1
0D29E74E
```

D.9 RSA$_{MA}$ mechanism

D.9.1 Key production

This example makes use of two RSA permutations with α = 1024 for the modulus size in bits and $v = 2^{16}+1$ (a prime number = 65537 in decimal = '10001' in hexadecimal) as the public exponent. Each public operation is "raising a positive integer, less than n, to the v-th power modulo n".

For entity A, $Id(A)$ = AAAAAAAA

p_1 =
```
D8E1FC6B 5EF57E8A DFDFE1AA 16D166F7 31698158 A0A52504 ACA04E3D 1F6B12DA
387ABA0B ADAD8662 34BC6ED5 04E0611D C54F58EB BC173BCF 55D63165 F597BADF
```

p_2 =
```
DCAA6C35 3285EE3A 1DBCCB2D 3EE4BC7D BA57C624 6426286B 63E07012 A1C13787
9AC8C1FF 627C1C84 CC51A3C5 D83FBDCC FA226AE0 D6C20CDE 6648F8D6 6D03B437
```

$n(A)$ =
```
BAF296AA 4CFC3098 0C37B059 8106F940 81CDEA52 B1665F9F BC44B290 D406AEFE
AA24B26E CCFCAB59 34319D1B E35D55CC 38A58455 7A48BA48 F6CDEF59 0A4EE069
D2C7F3FA 9CE0326D 7F85AD00 43107F12 DA10E0FD 8E202E61 5C2FA31E C3D2FC16
A5797159 420AFC11 7AB315A1 0265383E 38ADB448 9CC9F01A BC1A0758 969AF1E9
```

x =
```
01264ABC E35A4DA6 31B509F5 92F08B09 D58281A5 5E87E5A0 A2D4BD50 5CAD69D8
110DC6B7 1DB5940C 296100B9 522D99C0 76BCB5D2 9CBCE3C0 5D1C2913 A0179A13
```

```
A9D25AEA 03D5EBCD 3C774553 FB19AA24 8F7997D5 0D83231A F8BF2B93 064E261B
998B5165 D8B2AD0C B0D0CDF2 F4B9B70D 275AB200 E142C494 CC022BE7 E77DF3BD
```

For entity *B*, *Id*(*B*) = `BBBBBBBB`

p_1 =
```
AA9F65CF 91FD9997 7D2418A5 AA55F70A 7FFDD510 E8DDE122 B3CB0AA7 8C01F282
0628765A EED2E80D DFF97743 8B545205 6BB2F02A D54EA275 64CA89DE 693C1175
```

p_2 =
```
F1C58E34 3F040078 4B9FBE72 8E2B1A77 39D404D3 837F6BF8 27A52E5E 75E5897D
3DFCDE58 7A181452 69587E45 EC4D5D43 72F9F062 EFAB684E E7D90E83 6ECB4839
```

$n(B)$ =
```
A123BA48 FE04CBBE FA6216FC 2DDF3FAB FBFD5ACE C374C025 C5856D3B 2E214269
F8C2787A B571BDD4 EB3E6CD0 6F37A4DC B67DB9BA 2B41ADC2 97A2C208 F8EE4BFF
B1B074C1 D42D072D 4C73E6DD A279FDBF 13E2B93C 8E3CD8E6 48F85231 FBF9AEDC
FE9FB915 B8926C8C EC3F7040 5D9B27C1 A7484F17 71D01C37 CE52D71F 710FCB0D
```

x =
```
03172D47 ED0218A3 B04E2ECF C5E0DB6A 04DF4A56 EBF60F54 6086FE8A E0614505
C531C7A6 11A583F8 4D00EFF3 6FA76FCE 47AF8B52 51B5F804 EFC57D36 B723ECE3
4B802990 CE9674AE F22E62B9 73386B87 E84DDCD6 BFBCC46A 44DB93BA D0BEAC82
93AAF92C 3554CA3B D4EDEFC8 EFBD211A 21F0D739 25419F9C 2945D5B9 44E4DDE9
```

D.9.2 Mutual authentication exchange

The bit length is ρ = 384 for the fresh strings of random bits. This example makes use of SHA-1, i.e. the third hash-function specified in ISO/IEC 10118-3 [25].

Step 1

Id(*B*) = `BBBBBBBB`

r_B =
```
A7AD8272 F85FD5C1 4CEF982A 64347689 632DFE86 4C15BAAA D5A80CE3 877CF197
41210E0B 00254C8E D091CF32 8E8A247C
```

H_B =
```
9E640DAC 9550C381 9E9BD0CA 22BF28A8 B9BCA67A
```

d_B =
```
0C6687CB C627C016 DCAEE1A9 B0FEA668 C61374AD 4B3940D6 6398FD90 D60AFA93
8F76F1C7 C82C36E3 03AFD19E 9665E6DD 83365FFA B79A02AD 0542B679 DE40495D
0ABFCF0F 38C7F1FF FA7EFD24 FED36FD2 852FE56F F3B7AA9F 344A2EF9 17F3EAF5
3EF4B9F3 FE7B84A9 62C4F848 2FF94565 CC49B4BF 9C554A1F BBF3A2D0 CD0F5304
```

Step 3

$S_A(d_B)$=
```
BBBBBBBB A7AD8272 F85FD5C1 4CEF982A 64347689 632DFE86 4C15BAAA D5A80CE3
877CF197 41210E0B 00254C8E D091CF32 8E8A247C 9E640DAC 9550C381 9E9BD0CA
22BF28A8 B9BCA67A
```

Id(*A*) = `AAAAAAAA`

$r_B{}^*$ =
```
A7AD8272 F85FD5C1 4CEF982A 64347689 632DFE86 4C15BAAA D5A80CE3 877CF197
41210E0B 00254C8E D091CF32 8E8A247C
```

r_A =
```
A8F3D122 572A6C62 FB4E531B 13E00D9A BD5FDE87 4B214BF0 C8357B48 FD26BE12
72B37C25 F461465F 13FEF403 1131639A
```

H_A =
```
4B1F473A 25AF669A 442EA348 A481F897 3A96DBEE
```

d_A =
```
9E8611FF 140D9410 883F38EB E5259D90 EB49002F BA7572D6 1A634795 EB66FCCE
11217F50 07B5A3D2 5E907509 2F768958 71699359 4C7004B7 22E769BC F4BB27A9
9A07419C 905B9AD3 04356698 5C63E8AE 908BA029 B6D27028 2CBA24C2 D270D369
60C284A8 961431AC 955E86E1 D6A32820 441A082B E48BD96B 7C9E59F2 D7758090
```

Step 5

$S_B(d_A)$=
```
AAAAAAAA A7AD8272 F85FD5C1 4CEF982A 64347689 632DFE86 4C15BAAA D5A80CE3
877CF197 41210E0B 00254C8E D091CF32 8E8A247C A8F3D122 572A6C62 FB4E531B
13E00D9A BD5FDE87 4B214BF0 C8357B48 FD26BE12 72B37C25 F461465F 13FEF403
1131639A 4B1F473A 25AF669A 442EA348 A481F897 3A96DBEE
```

Bibliography

[1] M. Bellare and P. Rogaway, *The exact security of digital signatures: How to sign with RSA and Rabin*, in Proc. Eurocrypt '96, U. Maurer, Ed., Lecture Notes in Computer Science, Vol. 1070, Advances in Cryptology, pp 399-416, Berlin, Springer-Verlag, 1996

[2] J. Brandt, I. Damgård, P. Landrock and T. Pedersen, *Zero-knowledge authentication scheme with secret key exchange*, in Proc. Crypto '88, S. Goldwasser, Ed., Lecture Notes in Computer Science, Vol. 403, Advances in Cryptology, pp 583-588, Berlin, Springer-Verlag, 1990

[3] U. Feige, A. Fiat and A. Shamir, *Zero knowledge proofs of identity*, in Journal of Cryptology, Vol. 1, pp 77-94, 1988

[4] A. Fiat and A. Shamir, *How to prove yourself: Practical solutions to identification and signature problems*, in Proc. Crypto '86, A.M. Odlyzko, Ed., Lecture Notes in Computer Science, Vol. 263, Advances in Cryptology, pp 186-194, Berlin, Springer-Verlag, 1987

[5] M. Girault, *Self-Certified Public Keys*, in Proc. Eurocrypt '91, D.W. Davies, Ed., Lecture Notes in Computer Science, Vol. 547, Advances in Cryptology, pp 490-497, Berlin, Springer-Verlag, 1992

[6] M. Girault, L. Juniot, and M.J.B. Robshaw. The feasibility of on-the-tag public key cryptography. In RFIDSEC 2007, 11-13 July 2007

[7] M. Girault and D. Lefranc. Public key authentication with one (online) single addition. In CHES'04, pages 413-427, 2004

[8] M. Girault and J.C. Paillès, *On-line / off-line RSA-like*, Workshop on Cryptography and Coding 2003

[9] M. Girault, G. Poupard, and J. Stern. On the fly authentication and signature schemes based on groups of unknown order. J. Cryptology, 19(4):463-487, 2006

[10] S. Goldwasser, S. Micali and C. Rackoff, *The knowledge complexity of interactive proof systems*, in SIAM Journal on Computing, Vol. 18, pp 186-208, 1989

[11] L.C. Guillou and J.-J. Quisquater, *A practical zero-knowledge protocol fitted to security microprocessor minimizing both transmission and memory*, in Proc. Eurocrypt '88, C.G. Günther, Ed., Lecture Notes in Computer Science, Vol. 330, Advances in Cryptology, pp 123-128, Berlin, Springer-Verlag, 1988

[12] L.C. Guillou, M. Ugon and J.-J. Quisquater, *Cryptographic authentication protocols for smart cards*, in Computer Networks Magazine, Vol. 36, pp 437-451, North Holland Elsevier Publishing, July 2001

[13] D.E. Knuth, *The Art of Computer Programming*, Vol. 2. Addison-Wesley, 3rd edition, 1997

[14] A. K. Lenstra and E. R. Verheul, *Selecting cryptographic key sizes*, in Journal of Cryptology, Vol. 14-4, pp 255-293, 2001

[15] C.H. Lim and P.J. Lee, *A key recovery attack on discrete log based schemes using a prime order subgroup*, in Proc. Crypto '97, B. Kaliski, Ed., Lecture Notes in Computer Science, Vol. 1294, Advances in Cryptology, pp 249-263, Berlin, Springer-Verlag, 1997

[16] A.J. Menezes, P.C. van Oorschot and S.A. Vanstone, *Handbook of Applied Cryptography*, Boca Raton, CRC Press, 1997

[17] C.J. Mitchell and C.Y. Yeun, *Fixing a problem in the Helsinki protocol*, in ACM Operating Systems Review, Vol.. 32-4, pp. 21-24, October 1998

[18] A. M. Odlyzko, *The future of integer factorization*, in Cryptobytes, Vol. 1-2, pp 5-12, Summer 1995

[19] G. Poupard and J. Stern, *Security Analysis of a practical "on the fly" Authentication and Signature Generation*, in Proc. Eurocrypt '98, K. Nyberg, Ed., Lecture Notes in Computer Science, Vol. 1403, Advances in Cryptology, pp 422-436, Berlin, Springer-Verlag, 1998

[20] J.-J., M., M. and M. Quisquater, L.C., M.-A., G., A., G. and S. Guillou, with the help of T. Berson, *How to explain zero-knowledge protocols to your children*, in Proc. Crypto '89, G. Brassard, Ed., Lecture Notes in Computer Science, Vol. 435, Advances in Cryptology, pp 628-631, Berlin, Springer Verlag, 1990

[21] C.P. Schnorr, *Efficient identification and signatures for smart cards*, in Proc. Crypto '89, G. Brassard, Ed., Lecture Notes in Computer Science, Vol. 435, Advances in Cryptology, pp 239-252, Berlin, Springer Verlag, 1990

[22] R. Silverman, *A cost-based security analysis of symmetric and asymmetric key lengths*, RSA Labs Bulletin, Vol. 13, April 2000 (revised November 2001)

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

[24] ISO/IEC 9798-1:1997, *Information technology — Security techniques — Entity authentication — Part 1: General*

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

[26] ISO/IEC 11770-3:1999, *Information technology — Security techniques — Key management — Part 3: Mechanisms using asymmetric techniques*

[27] ISO/IEC 14888-2, *Information technology — Security techniques — Digital signature with appendix — Part 2: Integer factorization based mechanisms*

[28] ISO/IEC 15946-1:2008, *Information technology — Security techniques — Cryptographic techniques based on elliptic curves — Part 1: General*

[29] ISO/IEC 18032, *Information technology — Security techniques — Prime number generation*

[30] ISO/IEC 18033-1:2005, *Information technology — Security techniques — Encryption algorithms — Part 1: General*

[31] ISO/IEC 18033-2, *Information technology — Security techniques — Encryption algorithms — Part 2: Asymmetric ciphers*

ICS 35.040

Price based on 53 pages

This page has been intentionally left blank

BSI Group
Headquarters 389
Chiswick High Road,
London, W4 4AL, UK
Tel +44 (0)20 8996 9001
Fax +44 (0)20 8996 7001
www.bsigroup.com/
standards