INTERNATIONAL STANDARD

ISO/IEC 9798-5

Second edition
2004-12-01

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 9798-5:2004(E)

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

Published in Switzerland

Contents

Page

Foreword

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

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

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

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

This second edition cancels and replaces the first edition (ISO/IEC 9798-5:1999), which has been technically revised.

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 document specifies authentication mechanisms in the form of 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 (see Annex C), the mechanisms can be classified into the following four main groups.

— The first group (Clauses 5 and 6) 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 (Clauses 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 (Clause 9.3) 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 (Clause 9.4) has no possibility of a "coupon" strategy.

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

US 4 748 668 issued 1988-05-31, Inventors: A. Shamir and A. Fiat,

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.

News Digital Systems Ltd. Stoneham Rectory Stoneham Lane Eastleigh, Hampshire SO50 9NW, UK	US 4 748 668
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.

— Clause 5 specifies mechanisms (already present in the first edition, ISO/IEC 9798-4:1999) based on identities and providing unilateral authentication. They have been repaired after the withdrawal of ISO/IEC 9796:1991.

— Clause 6 specifies mechanisms (inserted in this second edition) based on integer factorization and providing unilateral authentication.

— Clauses 7 and 8 specify mechanisms based on discrete logarithms with respect to numbers that are either prime (see Clause 7, mechanisms already present in the first edition) or composite (see Clause 8, mechanisms inserted in the second edition), and providing unilateral authentication.

— Clause 9 specifies mechanisms based on asymmetric encipherment systems and providing either unilateral (see 9.3, mechanisms already present in the first edition), or mutual (see 9.4, mechanisms inserted in the second edition) authentication.

The verifier associates the correct verification key with the claimant by any appropriate procedure, for example, by retrieving it from a certificate. Such procedures are outside the scope of this part of ISO/IEC 9798.

To identify each mechanism, Annex A specifies object identifiers in accordance with ISO/IEC 8825-1.

These mechanisms are constructed using the principles of zero-knowledge techniques, but they will not be zero-knowledge according to the strict definition sketched in Annex B for every choice of parameters.

Annex C compares the mechanisms and provides guidance on parameter choices.

Annex D provides numerical examples.

2 Normative references

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

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

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

3 Terms and definitions

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

3.1
accreditation exponent
secret number related to the verification exponent and used in the production of private numbers

3.2
adaptation parameter
public number specific to the modulus and used in the definition of public numbers in the GQ2 mechanisms

3.3
asymmetric cryptographic technique
cryptographic technique that uses two related operations: a public operation defined by a public data item, key or number, and a private operation defined by a private data item, key or number (the two operations have the property that, given the public operation, it is computationally infeasible to derive the private operation)

3.4
asymmetric encipherment system
system based on asymmetric cryptographic techniques whose public operation is used for encipherment and whose private operation is used for decipherment

3.5
asymmetric pair
two related data items, keys or numbers, where the private data item defines a private operation and the public data item defines a public operation

3.6
challenge
procedure parameter used in conjunction with secret parameters to produce a response

3.7
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

3.8
claimant parameter
public data item, number or bit string, specific to a given claimant within the domain

3.9
decipherment
reversal of a corresponding encipherment
[ISO/IEC 9798-1]

3.10
domain
collection of entities operating under a single security policy, e.g., public key certificates created by a single certification authority, or by a collection of certification authorities using the same security policy

3.11
domain parameter
public number, or function, agreed and used by all entities within the domain

3.12
encipherment
reversible operation by a cryptographic algorithm converting data into ciphertext, so as to hide the information content of the data

3.13
entity authentication
corroboration that an entity is the one claimed
[ISO/IEC 9798-1]

3.14
exchange multiplicity parameter
number of exchanges of information involved in one instance of an authentication mechanism

3.15
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]

3.16
identification data
set of public data items (e.g., an account number, an expiry date and time, a serial number, etc.) assigned to an entity and used to identify it

3.17
mutual authentication
entity authentication that provides both entities with assurance of each other's identity
[ISO/IEC 9798-1]

3.18
number
natural integer, i.e., a non-negative integer

3.19
pair multiplicity parameter
number of asymmetric pairs of numbers involved in one instance of an authentication mechanism

3.20
private key or private number
that data item, key or number, 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

3.21
procedure parameter
public data item involved with a transient value in one instance of an authentication mechanism, e.g., witness, challenge, response

3.22
public key or public number
that data item, key or number, of an asymmetric pair, that can be made public and shall be used by every verifier for establishing the claimant's identity

3.23
random number
time variant parameter whose value is unpredictable
[ISO/IEC 9798-1]

3.24
response
procedure parameter produced by the claimant, and processed by the verifier for checking the identity of the claimant

3.25
secret parameter
number or bit string that does not appear in the public domain, only used by a claimant, e.g., a private number

3.26
token
message consisting of data fields relevant to a particular communication and which contains information that has been produced using a cryptographic technique

3.27
unilateral authentication
entity authentication that provides one entity with assurance of the other's identity but not vice versa
[ISO/IEC 9798-1]

3.28
verification exponent
public number used as exponent by the claimant and the verifier

3.29
verifier
entity including the functions necessary for engaging in authentication exchanges on behalf of an entity requiring an entity authentication

3.30
witness
procedure parameter that provides evidence of the claimant's identity to the verifier

4 Symbols and abbreviated terms

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

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

NOTE 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 [10, 13] 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 By definition, the Legendre symbol of any positive integer a with respect to any odd positive prime integer p is set 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.

$|A|$ 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., $|65\,537 = 2^{16}+1| = 17$), or bit length of the bit string A if A is a bit string

NOTE The binary representation of a number A as a string of $|A|$ bits is straightforward. For representing a number A as a string of α bits with $\alpha > |A|$, $\alpha - |A|$ bits set to 0 are appended on the left of the $|A|$ bits.

$\lfloor A \rfloor$ the greatest integer that is less than or equal to the real number A

$B \,||\, C$ bit string resulting from concatenating the two bit strings B and C in that order

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 number (domain parameter)
$G(A)$, $G_i(A)$	public number (claimant parameter)
h	hash-function
$\|h\|$	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} so that n divides $j - i$
j mod* n	the unique integer i from {0, 1, … (n–1)/2} so 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 number (secret parameter)
r	fresh random number or fresh string of random bits (secret parameter)
v	verification exponent (domain parameter)
W	witness (procedure parameter)
'XY'	notation using the hexadecimal digits '0' to '9' and 'A' to 'F', equal to XY to the base 16
α	modulus size in bits, i.e., $2^{\alpha-1} \leq$ modulus $< 2^{\alpha}$, also denoted $\|$modulus$\|$ (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)
{3, 5, 6}	set of the integers 3, 5 and 6

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 numbers in the first mode (domain parameter)

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

P_A public operation, i.e., encipherment (claimant parameter)

S_A private operation, i.e., decipherment (secret parameter)

x private RSA exponent (secret parameter)

5 Mechanisms based on identities

5.1 Security requirements for the environment

These mechanisms enable a verifier to check that a claimant knows private number(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 [8] 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. 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 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 number(s). During its life, the card uses its private number(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 number(s) as a short-term secret.

   NOTE 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 number(s) in accordance with an identity-based mechanism. The private numbers 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 numbers by some means. In this context, the identification data is a string of bits, nor 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 document. 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.

5.2 Key production

5.2.1 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 $\text{lcm}(p_1 - 1, p_2 - 1)/2$ if $v = 2$, or $\text{lcm}(p_1 - 1, \ldots p_f - 1)$ if v is an odd prime.

5.2.2 Asymmetric pair(s) of numbers

5.2.2.1 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 [21], 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) Named a mask, 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 number 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 numbers denoted Q_1 to Q_m. The private number denoted Q_x is set equal to the u-th modular power of the public number $G_x(A)$.

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

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

— 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.

5.2.2.2 Case where *v* is an odd prime

NOTE The mechanism below derives from the first format mechanism specified in ISO/IEC 14888-2 [21], 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) Named a mask, 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 number, 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 number, denoted Q, set equal to the u-th modular power of the public number $G(A)$.

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

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

— 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 number is the modular inverse of the RSA signature, see ISO/IEC 14888-2 [21]) verifies a relationship governed by the verification key.

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

5.3 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.

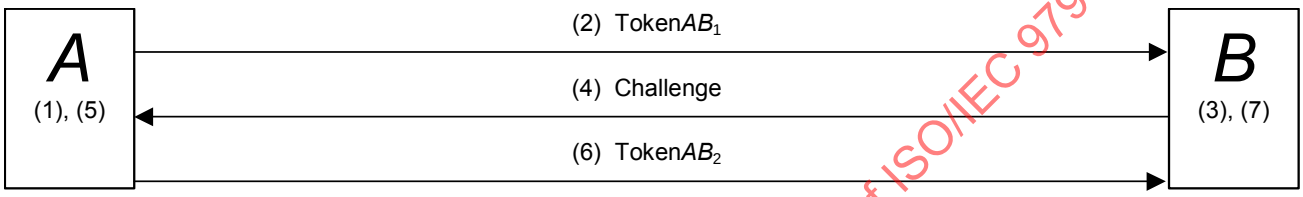

Figure 1 — Identity-based mechanism

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 numbers Q_1 to Q_m if $v = 2$, or
- a single private number 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 = either witness W or a hash-code of W and $Text$, one of four hash variants, to B.

   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 document. Annex A of ISO/IEC 9798-1 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 numbers $Q_1, Q_2, \ldots Q_m$ and the m challenge bits $d_1, d_2, \ldots d_m$ if $v = 2$.

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

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

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

6) *A* sends TokenAB_2 = response *D* to *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 numbers (see 5.2.2.1), denoted $G_1(A), G_2(A), \ldots G_m(A)$, if $v = 2$.
- a single public number (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 $v = 2$: $W^* = D^2 \times \prod_{i=1}^{m} G_i(A)^{d_i} \pmod{*}$ either n or $n(A)$
- Verification formula if v is an odd prime: $W^* = D^{v} \times G(A)^{d} \pmod{}$ 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 number(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 number(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 $r_1, r_2, \ldots r_t$, compute t witnesses $W_1, W_2, \ldots W_t$, 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.

6 Mechanisms based on integer factorization

6.1 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 [9] 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 document. 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.

6.2 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 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 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 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 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 numbers 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 number 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 number 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., $p_1 \times \ldots \times p_f$. It is a claimant parameter denoted $n(A)$.

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

6.3 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.

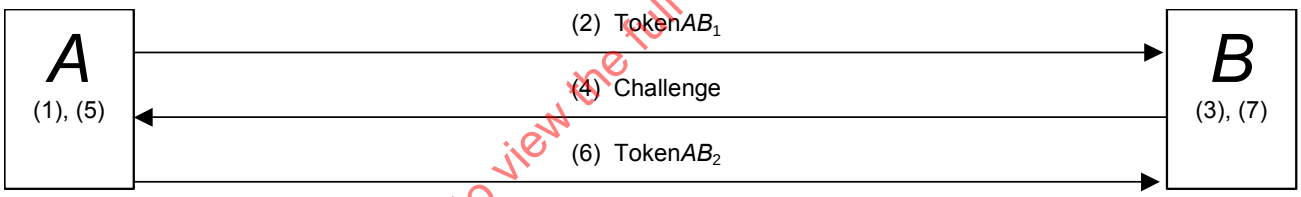

Figure 2 — Mechanism based on the factorization of a modulus

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 numbers Q_1 to Q_f, 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 = either witness *W* or a hash-code of *W* and *Text*, one of four hash variants, to *B*.

   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 document. Annex A of ISO/IEC 9798-1 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.

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

      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 = response *D* to *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.

7 Mechanisms based on discrete logarithms with respect to prime numbers

7.1 Security requirements for the environment

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

NOTE These mechanisms implement schemes due to Schnorr [18] 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 number.

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

   NOTE The exact means by which the verifier obtains a trusted copy of the public number specific to the claimant is beyond the scope of this document. 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.

7.2 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 number may be compromised by an attack of the type described by Lim and Lee [12]. 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 number denoted Q. It is represented by a string of $|q|$ bits.

Denoted $G(A)$, the public number 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.

7.3 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*.

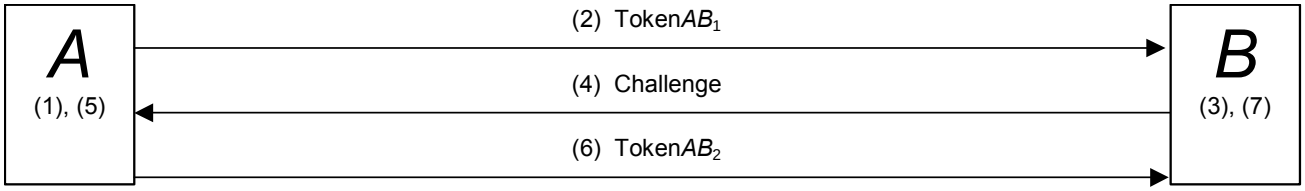

Figure 3 — Mechanism using a discrete logarithm with respect to a prime number

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

In the case of a coupon strategy, the claimant shall store a private number 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 number $G(A)$.

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 = either witness W or a hash-code of W and *Text*, one of four hash variants, to *B*.

   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 document. Annex A of ISO/IEC 9798-1 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 number Q.

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

6) *A* sends TokenAB_2 = response D to *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 number $G(A)$.

      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.

8 Mechanisms based on discrete logarithms with respect to composite numbers

8.1 Security requirements for the environment

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

NOTE These mechanisms implement schemes due to Girault, Poupard and Stern [5, 16] for GPS1, and to Girault and Paillès [6] 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 number.

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

   NOTE The exact means by which the verifier obtains a trusted copy of the public number(s) specific to the claimant is beyond the scope of this document. 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.

8.2 Key production

8.2.1 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.

8.2.2 First mode of use (GPS1)

A number, denoted σ, fixes the number of bits for representing private numbers. 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 number, 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 number 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.

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

8.2.3 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 The verification exponent v and the modulus $n(A)$ together form a public RSA key.

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

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

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

NOTE 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.

8.3 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.

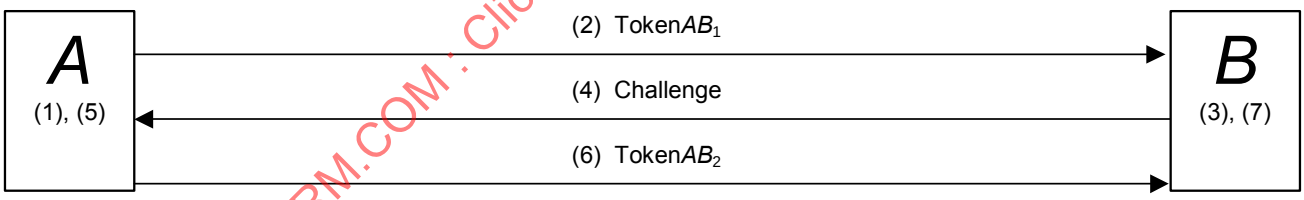

Figure 4 — Mechanism using a discrete logarithm with respect to a composite number

- In the first mode, the claimant shall store a number δ, a base g, a private number 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 number G, a verification exponent v, a private number 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 number 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 number $G(A)$ and a trusted copy of a modulus n or $n(A)$.
- In the second mode, in addition to a number δ, a public number 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 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 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 = either witness *W* or a hash-code of *W* and *Text*, one of four hash variants, to *B*.

   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 document. Annex A of ISO/IEC 9798-1 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 number *Q*.

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

6) *A* sends TokenAB_2 = response *D* to *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 TokenAB_1 can achieve efficiency gains by reducing the number of bits in TokenAB_1.

9 Mechanisms based on asymmetric encipherment systems

9.1 Security requirements for the environment

These mechanisms enable a verifier to check that a claimant knows the decipherment key corresponding to a claimed encipherment key.

NOTE These mechanisms derive from schemes due to Brandt, Damgård, Landrock and Pedersen [2, 13]. The second mechanism also derives from the key transport mechanism 6 from ISO/IEC 11770-3 [20], and Mitchell and Yeun [14].

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, and an asymmetric encipherment system, e.g., one of the systems specified in ISO/IEC 18033-2 [23].

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

9.2 Key production

Unless otherwise specified, each asymmetric key pair is an RSA key pair. Claimant A shall keep secret two or more distinct large prime factors denoted p_1, p_2 … in ascending order (for further details, see C.1.2).

— The public RSA operation denoted P_A is the v-th modular power. Certain values of the public exponent v, such as the prime numbers 3 and $2^{16}+1 = 65\,537$, have some practical advantages. Unless otherwise specified, the public exponent is a domain parameter. For each prime factor p_j, $p_j - 1$ shall be co-prime to v. Set equal to the product of the prime factors, the modulus is a claimant parameter denoted $n(A)$.

— The private RSA operation denoted S_A is the x-th modular power where the private exponent x is the least positive integer so that $x \times v - 1$ is a multiple of $\text{lcm}(p_1-1, \ldots p_f-1)$.

9.3 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.

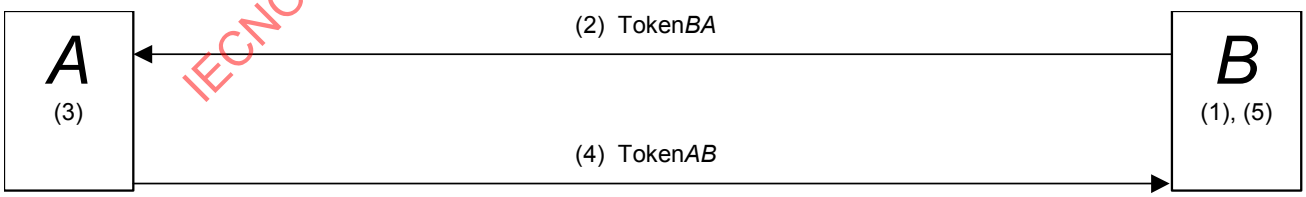

Figure 5 — Mechanism using an asymmetric key pair for encipherment

The claimant shall store a decipherment key, e.g., a private RSA key, fixing a private operation denoted S_A.

The verifier shall be provided with a trusted copy of an encipherment key, e.g., a public RSA key, fixing a public operation denoted P_A.

In the case of a coupon strategy, 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.

For fixing 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) Denoted *H*, a hash-code shall be computed from the fresh string *r*.

      $$H = h(r)$$
   c) Denoted *d*, a number shall be computed using P_A.

      $$d = P_A(r \,||\, H)$$
2) *B* sends Token*BA* = number *d* to *A*.
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* = string *r** to *B*.
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 encipherment system in use provides the property of non-malleability (see ISO/IEC 18033-2 [23]), then as the encipherment system includes a hash-function, the hash-code may be omitted from Token*BA*. In such a case, step 3.b is replaced by a check that the decipherment process completes correctly. However special care should then be taken to prevent the verifier using the claimant as a decrypting oracle.

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

9.4 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.

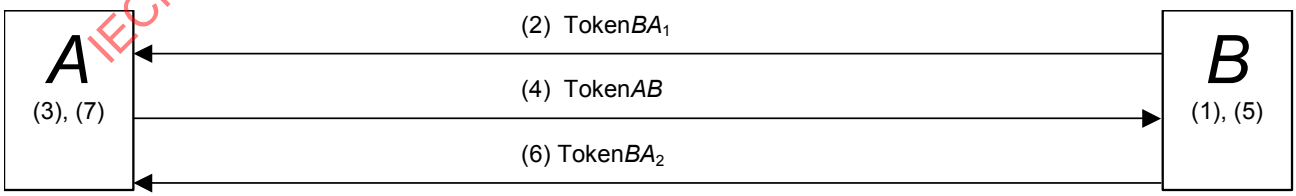

Figure 6— Mechanism using two asymmetric key pairs for encipherment

Each entity shall store a decipherment key, e.g., a private RSA key, fixing a private operation denoted either S_A, or S_B and be provided with a trusted copy of an encipherment key, e.g., a public RSA key, fixing a public operation denoted either P_B, or P_A. It shall also be provided with identification data, 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*).

For fixing 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) Denoted H_B, a hash-code shall be computed from the identification data $Id(B)$ and the fresh string r_B.
   $$H_B = h(Id(B) \,||\, r_B)$$
   c) Denoted d_B, a number shall be computed using P_A.
   $$d_B = P_A(Id(B) \,||\, r_B \,||\, H_B)$$
2) B sends TokenBA_1 = number d_B to A.
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. Denoted H_A, a hash-code shall be computed from the identification data $Id(A)$, the string $r_B{*}$ and the fresh string r_A.
      $$H_A = h(Id(A) \,||\, r_B{*} \,||\, r_A)$$
      iii. Denoted d_A, a number shall be computed using P_B.
      $$d_A = P_B(Id(A) \,||\, r_B{*} \,||\, r_A \,||\, H_A)$$
4) A sends TokenAB = number d_A to B.
5) On receipt of TokenAB, 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.
   $$Id_A{*} \,||\, r_B{**} \,||\, r_A{*} \,||\, H_A{*} = S_B(d_A)$$
   b) If the string $H_A{*}$ and the hash-code $h(Id_A{*} \,||\, r_B{**} \,||\, r_A{*})$ 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 = string $r_A{*}$ to 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 encipherment system in use provides the property of non-malleability (see ISO/IEC 18033-2 [23]), then as the encipherment system includes a hash-function, the hash-codes may be omitted from TokenBA_1 and TokenAB. In such a case, steps 3.b and 5.b are replaced by checks that the decipherment process completes correctly. However special care should then be taken to prevent the verifier using the claimant as a decrypting oracle.

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

Annex A
(normative)

Object identifiers

A.1 Formal definition

```
EntityAuthenticationMechanisms-8 {
    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-encipherment OID ::= { mechanism 7 }
ma-asymmetric-encipherment OID ::= { mechanism 8 }


END -- EntityAuthenticationMechanisms-8 --
```


A.2 Use of subsequent object identifiers

If a mechanism specified in this document 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).

For the last two mechanisms, another object identifier may follow for referring to an encipherment system (e.g., one of the mechanisms specified in ISO/IEC 18033-2 [23]). 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, 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 document.

— 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 encipherment system.

  — '08' identifies the mutual authentication mechanism using an asymmetric encipherment 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 or number. When asymmetric cryptography is being used, the claimant uses the private key or number 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 or number of the claimant that could not have been obtained just from knowledge of the public key or number 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 or number 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 numbers 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 or number 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 or number, which the verifier could not have obtained from the corresponding public key or number.

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 document 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 or number 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 document 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 [17] and a formal basis for a rigorous understanding of zero-knowledge protocols [3, 7].

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 document, 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 [15] 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 [15], 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 [19], and Lenstra and Verheul [11].

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 ISO/IEC 18032 [22] 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 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 numbers, 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 number. With the private number, 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 three 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} and RSA_{MA}, i.e., the mechanisms specified in Clause 9.

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 room for a "coupon" strategy with 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 [13], 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 [10, 13], 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, … x_1-1} and X_2 from {0, 1, … x_2-1}, into the corresponding unique number X from {0, 1, … $x-1$}. It makes use of the two numbers x_1 and x_2 and the CRT coefficient Cr as follows.

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

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

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

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	$W = r^2 \mod * n$	i.e., $X_{\lvert n \rvert}$
Response formula	$D = r \times \prod_{i=1}^{m} Q_i^{di} \mod * n$	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	$W* = D^2 \times \prod_{i=1}^{m} G_i^{di} \mod * n$	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	$W = r^v \mod n$	i.e., $(\lvert v \rvert - 1) \times X_{\lvert n \rvert} + (HW(v) - 1) \times M_{\lvert n \rvert}$
Response formula	$D = r \times Q^d \mod n$	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	$W* = D^v \times G^d \mod n$	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	$W_j = r_j^{2^{k+b}} \mod p_j$	i.e., $2 \times (k + b) \times X_{\lvert p \rvert} + ChC$
Response formula	$D_j = r_j \times \prod_{i=1}^{m} Q_{i,j}^{d_i} \mod p_j$	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	$W* = D^{2^{k+b}} \times \prod_{i=1}^{m} G_i^{d_i} \mod n$	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	$W = g^r \mod p$	i.e., $(\lvert r \rvert - 1) \times X_{\lvert p \rvert} + (HW(r) - 1) \times M_{\lvert p \rvert}$
Response formula	$D = r - d \times Q \mod q$	negligible

As $|r| = |q|$ and $HW(r) = |q|/2$, $CPC\ (M_\alpha) \approx 1{,}25 \times |q|$

Verification formula $W^* = g^D \times G^d \bmod p$ i.e., $(|D|-1)\times X_{|p|}+(HW(D)+ HW(d)-1)\times M_{|p|}$

As $HW(d) = \delta/2$, $|D| = |q|$ and $HW(D) = |q|/2$, $CPV\ (M_\alpha) \approx 1{,}25 \times |q| + 0{,}5 \times \delta$

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

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

C.2.4.5 GPS1

Without CRT, the claimant stores n and Q. CS (bits) $= |n| + |Q| = \alpha + \sigma$

Witness formula $W = g^r \bmod n$ where $g = 2$, i.e., $(|r|-1) \times X_{|n|}$

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

As $|r| = \rho = \sigma + \delta + 80$, $CPC\ (M_\alpha) \approx (3/4) \times (\sigma + \delta) + 60$

With CRT, the claimant stores p_1, p_2, Cr and Q. CS (bits) $= 1{,}5 \times |n| + |Q| = 1{,}5 \times \alpha + \sigma$

Witness formula $W_j = g^r \bmod p_j$ where $g = 2$, i.e., $2 \times (|r|-1) \times X_{|p|} + ChC$

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

As $|r| = \rho$ and $< 0{,}5 \times |n|$ and $ChC \approx 1{,}5 \times M_{|p|}$ $\approx 1{,}5 \times \rho \times M_{|p|}$

As $M_{|p|} \approx M_{|n|}/4$ 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., $(|D|-1) \times X_{|n|} + (HW(d) - 1) \times M_{|n|}$

As $HW(d) = \delta/2$ and $|D| = \rho$, $CPV\ (M_\alpha) \approx 0{,}75 \times \sigma + 1{,}25 \times \delta + 60$

TokenAB_1 = either W as $|n|$ bits or a hash-code as $|h|$ 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) $\approx |h| + \sigma + 2 \times \delta + 80$

C.2.4.6 GPS2

Without CRT, the claimant stores n and Q. CS (bits) $= 2 \times |n| = 2 \times \alpha$

Witness formula $W_j = G^{r \times v} \bmod n$ where $G = 2$, i.e., $(|r| + |v|) \times X_{|n|}$

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

$\approx (|n| + 2 \times \delta + 80) \times 0{,}75 \times M_{|n|}$

$CPC\ (M_\alpha) \approx (3/4) \times (\alpha + 2 \times \delta + 80)$

With CRT, the claimant stores p_1, p_2, Cr and Q. CS (bits) $= 2{,}5 \times |n| = 2{,}5 \times \alpha$

Witness formula $W_j = G^{r \times v \bmod p_j - 1} \bmod p_j$ where $G = 2$, i.e., $2 \times (|p|-1) \times X_{|p|} + ChC$

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

As $2 \times |p| = |n|$ and $ChC \approx 1{,}5 \times M_{|p|}$, $\approx |n| \times 0{,}75 \times M_{|p|}$

As $M_{|p|} \approx M_{|n|}/4$, $CPC\ (M_\alpha) \approx (3/16) \times \alpha$

Verification formula $W^* = G^{d + v \times D} \bmod n$ i.e., $(|D \times v|-1)\times X_{|n|} + (HW(D \times v)-1)\times M_{|n|}$

As $|D \times v| = \rho$, $HW(D \times v) = \rho/2$ and $\rho = \alpha + \delta + 80$, $CPV\ (M_\alpha) \approx 1{,}25 \times (\alpha + \delta) + 100$

TokenAB_1 = either W as $|n|$ bits or a hash-code as $|h|$ 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) $\approx \alpha + |h| + 2 \times \delta + 80$

C.2.4.7 RSA_{UA}

The claimant retains p_1, p_2, s_1, s_2 and Cr. CS (bits) = 2, 5 × $|n|$ = 2, 5 × α

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

TokenBA = d as $|n|$ bits; TokenAB = r^* as $|n|-|h|$ bits. CM (bits) ≈ 2 × $\alpha - |h|$

C.2.4.8 RSA_{MA}

Each entity retains p_1, p_2, s_1, s_2 and Cr. CS (bits) = 2, 5 × $|n|$ = 2, 5 × α

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

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

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 δ = 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 7BEC69EF 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.

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 A3802B4B 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 numbers 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 = 27BA1C93 CA623C79 7CCF0560 184BDBEA 57DC069C 441E0B46 9B647419 87E5AA36
57619FAD B8F176E5 2D6A1D4F 26A0904D FCFF99D4 3453EB0A F3CEEA61 45B7C087
EFC9DC15 4B9933D3 98B0829E 77F8F55C 17F2EC82 0931E239 FB4D246C 84689D7D
A6614867 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 C8427715B 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