

PAKZ Editorial Changes

August 3, 2005

This document describes changes to the P1363.2 draft D21 to clarify and simplify the description, and to resolve ambiguities and errors in conversion functions for signatures and keys. Pending further discussion by the P1363 Working Group, these changes will be incorporated into draft D22 in August, 2005.

What's new

D20 and earlier drafts of PAKZ were intended to accommodate any 1363 signature scheme, but they had errors, especially for use of IFSSA and IFSSR. In D21, use of IF schemes was omitted pending resolution of these issues. However, there were further issues with representations of signatures and public keys. These changes better describe how to use APKAS-PAKZ with any 1363a scheme, by defining specific octet string conversion functions for:

- **Signatures.** Deleted the OS2SIGP-1 and SIG2OSP-1 signature/octet-string conversion functions in favor of re-using the formats and conversion methods described in IEEE Std 1363a-2004 E.3. Using the E.3 descriptions is simpler, and corrects the flaws and omissions in the prior treatment.
- **Private keys.** Introduced section E.4 describing example formats and conversions for representing private keys as octet strings. This was obvious for DL/EC cases, but not so obvious for IF.
- **Public keys.** Specify GE2OSP-X to convert the Server's public key w_S to an octet string. Earlier drafts left this conversion unspecified. GE2OSP-X was a natural choice for this since it is the only primitive that converts group elements to octet strings and it is already used in PAKZ (for converting password-mask group element τ_m).

Functional changes

Some of these changes may either encourage or mandate implementations of PAKZ that are not interoperable with an implementation based on earlier drafts. Potential interoperability changes include:

- The representation of IF private keys as octet strings, since the new method in E.4 may not have occurred to implementors of D20.
- The new GE2OSP-X function that converts public key w_S to an octet string may not be the same as one used in an implementation of D21, such as FE2OSP(GE2SVFEP(w_S))) for the EC setting.

Regarding the improved specification for converting IF signatures and DL/EC private keys, which are single integers, the only obvious working interpretation of earlier drafts would have been to use OS2IP and I2OSP with the appropriate length, which is interoperable with this draft. Regarding {DL,EC}SSR-PV signatures, the most reasonable interpretation for the formerly incorrect description would be to use the method as specified in E.3, now explicitly referenced.

Open questions

Should the conversion functions in E.3 and E.4 should be re-categorized as normative, rather than informative, similar to GE2OSP-X? In this draft, the E.3 and E.4 functions are specified as "may be used", to be consistent with their informative status, whereas use of the normative GE2OSP-X for w_S is mandatory.

Must the Client always validate the private signature key? And why? For the IF case, this is (at least) an editorial problem, as there is no specified method to validate an IF private key. Also, reasons for validation should be noted.

Finally, this draft does not incorporate the proposed additional PAKZ hash operation discussed in the July 20, 2005 teleconference; A separate document describes that proposal.

8. Primitives

8.1 Notation and definitions

8.1.6 Summary of terms

A_S	An octet string representing a hidden and password-masked blinded-private signature key used in <u>APKAS-PAKZ</u> key confirmation
$Os2sigp$	A conversion function that converts an octet string to a signature in signature scheme S_s
$o_{u\pi}$	An octet string <u>representation encoding</u> of a private signature key u masked by a function of π
S_1, S_C	Signature and <u>hidden masked</u> -signature octet strings, respectively, used in <u>APKAS-PAKZ</u> key confirmation
$Sig2osp$	A conversion function that converts a signature produced by $Sign$ to an octet string

8.2 Primitives

8.2.13 PVDGP-PAKZ

{DL,EC}PVDGP-PAKZ is {Discrete Logarithm, Elliptic Curve} Password Verification Data Generation Primitive, version PAKZ. PVDGP-PAKZ is based on the work of [MacK02]. This primitive derives password verification data from a party's password, using {DL,EC} domain parameters. It derives the password verification data used in {DL,EC}APKAS-PAKZ-SERVER.

This primitive is parameterized by the following choices:

- A signature scheme S_s , which may be either {DL,EC}SSA, {DL,EC}SSR, or {DL,EC}SSR-PV (see IEEE Std 1363a-2004), which is the signature scheme option of APKAS-PAKZ that is associated with a private key u , and a corresponding public key v , and an agreed octet string format for representing private keys as strings of where $uLen$ is the length in octets of the I2OSP encoding of private key u as determined by the domain parameters of (u,v) .
- A REDP function $Redp$, which should be ECREDP-1 or ECREDP-2.
- A mask generation function Mgf , which is the mask generation function scheme option of APKAS-PAKZ.

Input:

- The password-based octet string π
- The {DL,EC} domain parameters associated with π

Assumptions: Domain parameters are valid.

Output: The derived password verification data $(\pi_m, v, o_{u\pi})$, consisting of password-based mask group element π_m , public key v for signature scheme S_s , and octet string $o_{u\pi}$ of length $uLen$ containing the password-masked private key associated with v (See Notes 2 and 3)

Operation: The password verification data shall be computed by the following or an equivalent sequence of steps:

1. Compute password-mask group element $\pi_m = Redp(hex(01) \parallel \pi)$ (See Note 1)
2. Compute an octet string $o_1 = hex(02) \parallel \pi$

3. Compute a password-mask octet string o_2 of length $uLen$ as $o_2 = Mgf(o_1, uLen)$
4. Generate a key pair (u, v) for signature scheme S_s
5. Compute a ~~private key~~ octet string o_u of length $uLen$ as the agreed octet string representation of private key using I2OSP(u)
6. Compute password-masked private key octet string $o_{u\pi} = o_u \oplus o_2$
7. Output $(\pi_m, v, o_{u\pi})$ (See Note 2)

Conformance region recommendation: A conformance region should include limitations for any input values as discussed in Annex B.

NOTES

- 1—Steps 2 through 8 of BPKAS-PAK key agreement operation are re-used in APKAS-PAKZ with the value of π_m as computed here. The computation of π_m is different in BPKAS-PAK.
- 2—The output v could be public, but the other outputs must be kept hidden.
- 3—The APKAS-PAKZ server may want to store π_m^{-1} to optimize efficiency.

9. Password-Authenticated Key Agreement Schemes

9.7 APKAS-PAKZ

Editor's Note—The D21 description of APKAS-PAKZ was modified to exclude IF signature schemes due to two inconsistencies in the D20 specification: (1) It refers to domain parameters associated with signature keys (u, v) , which are used in SIG2OSP-1 and OS2SIGP-1, but IF signature schemes have no domain parameters. (2) It uses private key u as an integer, but IF private keys are tuples of two or more integers.

Editor's Note—This D21+ proposal for PAKZ does not yet incorporate the proposed additional hash operation discussed in the July 20 2005 teleconference.

Editor's Note—This D21+ proposal clarifies use of conversion functions, especially for the use of IFSSA, IFSSR, and {DL,EC}SSR-PV signature schemes. Specifically, the OS2SIGP-1 and SIG2OSP-1 functions were deleted in favor of re-using IEEE Std 1363a-2004 E.3. This is both simpler, and it corrects flaws and omissions in the D21 and earlier treatment of various signature schemes. And GE2OSP-X(w_ζ) is now an explicit mandatory step prior to the *Sign* and *Verify* operations.

Editor's Note—The Working Group plans to discuss a problem in the proof of security found by Phil MacKenzie, which may motivate a change in the protocol and/or further clarifying notes.

{DL,EC}APKAS-PAKZ-{CLIENT,SERVER} is {Discrete Logarithm, Elliptic Curve} Augmented Password-Authenticated Key Agreement Scheme, version PAKZ for {Client, Server}. It is based on the work of [MacK02].

9.7.1 Scheme options

Both the Client and Server parties shall establish or otherwise agree upon the following options:

- Primitives for password verification data generation, public key generation and secret value derivation, which shall be PVDGP-PAKZ, PEPKGP-PAK, PKGP-DH, SVDP-PAK1-CLIENT and SVDP-PAK2, and their associated parameters
- A set of valid {DL,EC} domain parameters (including k , q , r , and g), associated with the values π , π_m , u , v , $o_{u\pi}$, and v_π defined below

- A signature scheme S_s associated with a private key u and corresponding public key v , which which should be selected from the schemes specified in IEEE Std 1363a-2004 Clause 10~~may be either {DL,EC}SSA, {DL,EC}SSR, or {DL,EC}SSR-PV (see IEEE Std 1363a-2004), which defines or is otherwise associated with:~~
 - a signature generation operation $Sign$,
 - a signature verification operation $Verify$,
 - a format for representing a signature as an octet string, which may be a format described in IEEE Std 1363a-2004 E.3, where the length of the string is denoted by $sLen$,
 - a format for representing a private key as an octet string, which may be a format described in E.4, where the length of the string is denoted by $uLen$,

Editor's Note—Beginning in this D21+ proposal, octet string representation formats and conversion functions are described in informative sections 1363a E.3 and P1363.2 E.4.

- ~~if S_s is an IF scheme, a representation of IF private keys as an ordered set of integers, which should be one of the representations described in IEEE 1363-2000 8.1.3 (see Note 5), and~~
- ~~if S_s is an IF scheme, and if needed to use the agreed octet string format for representing private keys (as in, for example, the IF formats of E.4), a size for the IF modulus n .~~
- ~~a $Sig2osp$ conversion function that converts a signature to an octet string, which should be SIG2OSP 1, and~~
- ~~an $Os2sigp$ conversion function that converts an octet string to a signature, which should be OS2SIGP 1, where~~
- ~~$uLen$ is the length in octets of the I2OSP encoding of private key u and $sLen$ is the length in octets of the $Sig2osp$ encoding of a signature produced by $Sign$, as determined by the domain parameters of (u,v) .~~
- A mask generation function Mgf , which should be MGF1 (see 14.2.1). The same Mgf parameter shall be used with {DL,EC}PVDGP-PAKZ.
- A key derivation function Kdf , which should be KDF1 or KDF2
- One or more key derivation parameter octet strings $\{P_1, P_2, \dots\}$ to be used to derive agreed keys
- A key confirmation function, which should be KCF1

For CLIENT only:

- A password-based octet string π (See Note 4.)

For SERVER only:

- Password verification data values that were derived from {DL,EC}PVDGP-PAKZ(π), using the Client's values for π and mask generation function Mgf , including:
 - a password-based mask ~~group element~~ octet string π_m ,
 - the public key v associated with the signature scheme S_s , and
 - an octet string $o_{u\pi}$ of length $uLen$ containing the password-masked value of o_u , where o_u is the agreed octet string representation of the private key u that corresponds to v .

9.7.2 Key agreement operation

A sequence of shared secret keys, K_1, K_2, \dots, K_r , shall be generated by each party by performing the following or an equivalent sequence of steps:

1. For *CLIENT* only:
 - 1.1 Compute a mask group element π_m using Step 1 of {DL,EC}PVDGP-PAKZ(π)
2. Perform the *Key agreement operation* Steps 2 through 8 as specified for {DL,EC}BPKAS-PAK-{CLIENT,SERVER} in Subclause 9.2.2 to derive keys K_1, K_2, \dots, K_r

NOTE—The Client shall confirm the Server's knowledge of shared secret Z before any derived keys are used. Key confirmation for APKAS-PAKZ is described in 9.7.3.

3. Output derived keys K_1, K_2, \dots, K_r

9.7.3 Key confirmation operation

It is mandatory in APKAS-PAKZ for the Client to confirm the Server's knowledge of the shared secret Z , before the Client uses Z or any derived shared secrets K_i for other purposes. It is also mandatory for the Server to confirm the Client's knowledge of π , as this provides the augmented advantage of APKAS-PAKZ over BPKAS-PAK.

Key confirmation may be achieved using the following or an equivalent sequence of steps:

9.7.3.1 Key confirmation for Client

(Mandatory) The Client verifies that the Server's knows the verification data corresponding to π as follows:

1. Receive octet string o_S from the Server
2. Compute $o_\pi = \text{GE2OSP-X}(\pi_m)$
3. Compute $o_3 = \text{KCF1}(\text{hex}(03), w_C, w_S, Z, o_\pi)$
4. If $o_3 \neq o_S$, output "invalid" and stop.

Note—Step 4 must be performed before the Client uses Z or any derived shared secrets K_i for other purposes.

(Mandatory) The Client proves knowledge of π to the Server as follows:

5. Receive octet string A_S , a hidden and ~~password-masked~~ blinded-private signature key, from the Server
6. Compute $o_7 = \text{KCF1}(\text{hex}(05), w_C, w_S, Z, o_\pi)$
7. Compute $o_5 = \text{Mgf}(o_7, uLen)$
8. ~~Compute the password-masking value octet string~~ $o_4 = \text{Mgf}(\text{hex}(02) \parallel \pi, uLen)$
9. ~~Combine the password-mask and~~ o_5 values with A_S to produce the octet string o_u , a representation of the signature private key, using $o_u = A_S \oplus o_5 \oplus o_4$ ~~$\text{Mgf}(\text{hex}(02) \parallel \pi, uLen)$~~
10. ~~If the format of~~ o_u is incompatible with the agreed octet string representation of a private key for S_s , output "invalid" and stop.
11. ~~Convert~~ ~~signature private key~~ u from o_u using the agreed octet string representation
12. ~~If~~ u is not a valid private key for the *Sign* operation, output "invalid" and stop. (See Note 6)
13. Compute $o_w = \text{GE2OSP-X}(w_S)$

- ~~1411.~~ Compute $s = \text{Sign}(u, o_w, \pi_s)$
- ~~1412.~~ Convert signature s into an octet string S_1 using the agreed representation of a signature. Compute $S_1 = \text{Sig2osp}(s)$
- ~~1513.~~ Compute $o_8 = \text{KCF1}(\text{hex}(06), w_C, w_S, Z, o_\pi)$
- ~~1614.~~ Compute $o_6 = \text{Mgf}(o_8, sLen)$
- ~~1715.~~ Compute $S_C = S_1 \oplus o_6$
- ~~1816.~~ Send S_C to the server

9.7.3.2 Key confirmation for Server

(Mandatory) The Server proves to the Client knowledge of the verification data corresponding to π as follows:

1. Compute $o_\pi = \text{GE2OSP-X}(\pi_m)$
2. Compute $o_S = \text{KCF1}(\text{hex}(03), w_C, w_S, Z, o_\pi)$
3. Send o_S to the Client

(Mandatory) The Server verifies that the Client's knows π as follows:

4. Compute $o_7 = \text{KCF1}(\text{hex}(05), w_C, w_S, Z, o_\pi)$
5. Compute $o_5 = \text{Mgf}(o_7, uLen)$
6. Compute $A_S = o_{u\pi} \oplus o_5$
7. Send A_S to the Client
8. Receive octet string S_C from the Client
9. Compute $o_8 = \text{KCF1}(\text{hex}(06), w_C, w_S, Z, o_\pi)$
10. Compute $o_6 = \text{Mgf}(o_8, sLen)$
11. Compute octet string $S_1 = S_C \oplus o_6$
12. If the format of S_1 is incompatible with the agreed octet string representation of a signature, output "invalid" and stop.
13. Convert octet string S_1 into a signature s based on the agreed representation. Compute $s = \text{Osp2sig}(S_1)$
14. Compute $o_w = \text{GE2OSP-X}(w_S)$
- ~~1513.~~ If $\text{Verify}(v, s, o_w) = \text{"invalid"}$, output "invalid" and stop.

Conformance region recommendation: A conformance region should include limitations for any input values as discussed in Annex B.

NOTES

1—APKAS-PAKZ is a unilateral commitment scheme, where the Server does not provide a commitment to the password during the key agreement operations. In either the scheme or the invoking application protocol, the Client must verify the Server's proof of knowledge of the password-based value π_m before revealing any information derived from the output derived keys. See D.5.4.9 for discussion of the limitations on the use of unilateral commitment schemes in application protocols.

2—In this scheme, the test for a valid {DL,EC} password-entangled public key is the same as the test for a {DL,EC} public key. That is, a public key w is presumed valid if it specifies an element of order r in the group defined by the {DL,EC} domain parameters.

3—The steps for the Server to verify the Client's key are mandatory because they are needed to achieve the augmented benefit of APKAS-PAKZ.

4—The APKAS-PAKZ server may want to store π_m^{-1} to optimize efficiency.

5—When using an IF signature scheme, the parties may need to agree on the size of the IF modulus n , and on a specific multi-integer representation of private keys, in order to agree on the sizes of the agreed octet string representations of private keys ($uLen$) and signatures ($sLen$). When using a {DL,EC} signature scheme, the APKAS-PAKZ scheme option domain parameter r determines the lengths of the recommended octet string representations.

6—Methods for the Client to validate a {DL,EC} private signature key are discussed in 8.1.3. Validation of IF private keys is discussed in IEEE Std 1363-2000 8.1.3, however, no validation algorithm is specified.

Editor's Note—There is at least an editorial conflict with the mandatory step where the Client validates the private signature key, and the fact that 1363 does not specify how to perform this function for IF schemes. The WG should also consider whether this function is merely to make *Sign* a well-defined operation, or whether it is a larger functional issue.

14. Auxiliary techniques

Editor's Note—In this D21+ proposal, subclause 14.6 “*Converting between signatures and octet strings*” is removed, in favor of the more complete and accurate description in 1363a-2004 E.3. See 9.7 APKAS-PAKZ.

14.6 Converting between signatures and octet strings

14.6.1 SIG2OSP-1

SIG2OSP-1 is signature to octet string conversion primitive version 1. It converts a signature that consists of an ordered tuple of one or more integers into an encoded octet string, by converting each integer into an octet string and concatenating the strings. The number of and order of individual octet strings are determined by the signature scheme specification, and the lengths of the strings are determined by the domain parameters associated with the signature. SIG2OSP-1 is used in {DL,EC}APKAS-PAKZ-CLIENT.

OS2SIGP-1, defined in Subclause 14.6.2, performs the reverse of this operation.

This function is parameterized by the following choices:

- A signature scheme S_s , where S_s may be one of {DL,EC}SSA, {DL,EC}SSR, or {DL,EC}SSR-PV (see IEEE Std 1363a-2004). Each signature scheme S_s defines a representation of a signature as an ordered tuple of j integers (e_1, e_2, \dots, e_j) . (See Notes 1 and 2.)

Input:

- An ordered tuple of j integers (e_1, e_2, \dots, e_j) and associated signature scheme domain parameters that represents a signature generated by S_s . The signature scheme S_s and associated domain parameters define:
 - An ordered tuple of j integers (l_1, l_2, \dots, l_j) , where, for each i in the range $[1, j]$, $l_i = \lceil \log_{256} m_i \rceil$, where m_i is the maximum value of an integer e_i in a signature (e_1, e_2, \dots, e_j) that may be generated by S_s when using the associated signature scheme domain parameters.

Output: An octet string o that is the encoded signature

The octet string o is computed as follows:

1. For each i from 1 to j , convert e_i into an octet string o_i of length l_i using I2OSP
2. Compute $o = o_1 \parallel o_2 \parallel \dots \parallel o_j$
3. Output o

Notes

1—The value j and the ordered tuple of j integers values for each of the specified signature schemes is defined in the following table. The definitions of relevant integer values $\{e, d, C, \text{ and } s\}$ in each signature tuple are provided by the signature scheme.

Signature scheme	j	(e_1, e_2, \dots, e_j)
{DL,EC}SSA	2	(e, d)
{DL,EC}SSR	2	(e, d)
{DL,EC}SSR PV	2	(C, d)

2—Each integer value e_i in a signature tuple (e_1, e_2, \dots, e_j) is presumed to be greater than or equal to zero.

14.6.2 OS2SIGP-1

~~OS2SIGP 1 is octet string to signature conversion primitive version 1. It converts an octet string that represents an encoded signature into a tuple of one or more integers, by splitting the input string into substrings of appropriate lengths, and converting each substring into an integer. The number of and order of the integers are determined by the specification of the input signature scheme, and the sizes of the integers are determined by the signature scheme domain parameters associated with the encoded signature. OS2SIGP 1 is used in {DL,EC}APKAS PAKZ SERVER.~~

~~SIG2OSP 1, defined in Subclause 14.6.1, performs the reverse of this operation.~~

~~This function is parameterized by the following choices:~~

~~—A signature scheme S_s , where S_s may be one of {DL,EC}SSA, {DL,EC}SSR, or {DL,EC}SSR PV (see IEEE Std 1363a 2004). Each signature scheme S_s defines a representation of a signature as an ordered tuple of j integers (e_1, e_2, \dots, e_j) (see Notes 1 and 2 of Subclause 14.6.1 SIG2OSP 1).~~

~~Input:~~

~~—An octet string o and associated signature scheme domain parameters that represents an encoded signature generated by the signature scheme S_s , that has been encoded using SIG2OSP 1. The signature scheme S_s and associated domain parameters define:~~

~~—An ordered tuple of j integers (l_1, l_2, \dots, l_j) , where, for each i in the range $[1, j]$, $l_i = \lceil \log_{256} m_i \rceil$, where m_i is the maximum value of an integer e_i in a signature (e_1, e_2, \dots, e_j) that may be generated by S_s when using the associated signature scheme domain parameters.~~

~~**Assumptions:** The length of octet string o equals $(l_1 + l_2 + \dots + l_j)$.~~

~~**Output:** A tuple of integers (e_1, e_2, \dots, e_j) that is the signature~~

~~The signature is computed as follows:~~

- ~~1. Compute (o_1, o_2, \dots, o_j) from o such that $o = o_1 \parallel o_2 \parallel \dots \parallel o_j$,
—where for each i in the range $[1, j]$, o_i is of length l_i~~
- ~~2. For $i = 1$ to j , let $e_i = \text{OS2IP}(o_i)$.~~
- ~~3. Output the tuple (e_1, e_2, \dots, e_j) .~~

Annex E (Informative) Formats

Editor's Note—Section E.3.1 and E.3.2 are from IEEE Std 1363a-2004, and are copied here for reference only.

E.3 Representing outputs of schemes as octet strings

E.3.1 Output data format for DL/EC signature schemes

For DL/ECSSA and DL/ECSSR, the output of the signature generation function (see 10.2.2 and 10.4.2) is a pair of integers (c, d) . Let r denote the order of the generator (g or G) in the DL or EC domain parameters, and let $l = \lceil \log_{256} r \rceil$ (i.e., l is the length of r in octets). The output (c, d) may be formatted as an octet string as follows: convert the integers c and d to octet strings C and D , respectively, of length l octets each, using the primitive I2OSP, and output the concatenation $C \parallel D$ as the signature. To parse the signature, split the octet string into two components C and D , of length l octets each, and convert them to integers c and d , respectively, using OS2IP. Note that it is essential that both C and D be of length l octets, even if it means that they have leading zero octets.

For DL/ECSSR-PV, the output of the signature generation function (see 10.5.2) is a pair (C, d) where C is an octet string and d is an integer. The output (C, d) may be formatted as an octet string as follows: convert the integer d to an octet string D of length l octets (where l is as defined above) using the primitive I2OSP, and output the concatenation $C \parallel D$. To parse the signature, let C be all but the rightmost l octets of the signature and let D be the rightmost l octets, and convert D to an integer d using OS2IP.

NOTE—The output of DL/ECSSA may also be formatted according to the following method, described in more detail in ANSI X9.57-1997 [B6] and ANSI X9.62-1998 [B11]: combine c and d into an ASN.1 structure (ISO/IEC 8824-1:1998 [B72]) and encode the structure using some encoding rules, such as BER or DER (ISO/IEC 8825-1:1998 [B76]).

E.3.2 Output data format for IF signature schemes

For IFSSA and IFSSR, the output of the signature generation function (see 10.3.2 and 10.6.2) is an integer s . Let $k = \lceil \log_{256} n \rceil$ denote the length in octets of the modulus n in the IF public key. The output s may be formatted as an octet string by simply converting it to an octet string of length k octets using the primitive I2OSP.

E.4 Representing private keys as octet strings

When a private key needs to be represented as an octet string, it may be done as defined in this section.

E.4.1 Data format for DL/EC private keys

A {DL,EC} private key (see 8.1.3 and IEEE 1363-2000 6.1.3 and 7.1.3) is an integer in a range $[1, r-1]$, where r denotes the order of the desired group in the {DL,EC} domain parameters. Let $l = \lceil \log_{256} r \rceil$ denote the length of r in octets. The private key may be formatted as an octet string by converting it to a string of l octets using the primitive I2OSP. The private key may be recovered from this formatted octet string representation by using the primitive OS2IP.

E.4.2 Data format for IF private keys

An IF private key may be represented as an ordered pair, triple, or quintuple of integers, designated by (n, d) , (p, q, d) , or (p, q, d_1, d_2, c) , where each integer is in the range $[1, n-1]$, where n is the modulus in the corresponding IF public key, as described in IEEE 1363-2000 8.1.3. A specific multi-integer representation of an IF private key for a specific size of n may be formatted as an octet string as follows: Let $j_1 = \lceil \log_{256} n \rceil$ denote the length in octets of n . Let $j_2 = \lceil j_1/2 \rceil$. Let $[x]$ denote the octet string component of length j octets corresponding to integer component x of the private key, which is produced by $I2OSP(x, j)$, where $j=j_1$ when x is n or d , and where $j=j_2$ when x is $p, q, d_1, d_2,$ or c . The octet string representation of a private key is formed by concatenating the octet string components, as in $[n] \parallel [d]$ for an integer pair, $[p] \parallel [q] \parallel [d]$ for a triple, and $[p] \parallel [q] \parallel [d_1] \parallel [d_2] \parallel [c]$ for a quintuple. To parse this octet string representation, split the octet string into the specified two, three, or five octet string components, with $[n]$ and $[d]$ of length j_1 , and $[p], [q], [d_1], [d_2],$ and $[c]$ of length j_2 , and convert them to integers using $OS2IP$. Note that it is essential that each of the octet string components be of the specified length, even if it means that they have leading zero octets.

Note—When representing integer triples or quintuples, this format assumes that p and q are each less than $2^{\frac{8 \times \lceil \log_{256} n \rceil}{2}}$. This assumption is true if p and q are of the same size (that is, if $\lfloor \log_2 p \rfloor = \lfloor \log_2 q \rfloor$).

Editor's Note—Consider referring to other common IF private key octet string formats.

Editor's Note—If the noted assumption that p is the same size as q is not desired, consider using j_1 for all component string lengths.

Summary of changes— 8.1.6 *Summary of terms*

Minor notation changes (for example “blinded” to “password-masked”, for consistency).
Removed *Sig2osp/Os2sigp*.

— 8.2.13 *PVDGP-PAKZ*

Parameters: Removed list of recommended *Ss* schemes, already specified in APKAS-PAKZ.

Added reference to agreed octet string format for private keys.

Output: Corrected o_u to o_{int} .

Operation: Replaced incorrect I2OSP with general reference to agreed representation.

— 9.7 *APKAS-PAKZ*

Options: Modified to recommend *any* 1363 signature scheme, including IFSSA and IFSSR.

Changed to refer to 1363a-2004 E.3 instead of the (incorrect and deleted) SIG2OSP-1/OS2SIGP-1.

Added that IF schemes should agree on an integer private key representation and length of modulus n .

Fixed description of group element π_m , and expanded description of o_{int} .

Client key confirmation: Fixed description of A_S .

Expanded description of computing o_u .

Added step to abort if o_u is not in the agreed format, which is not always OS2IP.

Added Note 6 about validation of private keys.

Converts w_S to octet string format, using GE2OSP-X, before *Sign* operation.

Converts s into S_1 using the agreed representation instead of *Sig2osp*.

Add w_S as a parameter to the *Verify* operation.

Server key confirmation: Added step to abort if S_1 is not in the agreed format.

Converts S_1 into s using the agreed representation instead of *Os2sigp*.

Converts w_S to octet string format, using GE2OSP-X, before *Verify* operation.

Adds w_S parameter to *Verify* operation.

Added Notes 5 and 6.

Added *Editor's Notes*

— 14.6 *Converting between signatures and octet strings*

Deleted 14.6.1 *SIG2OSP-1* and 14.6.2 *OS2SIGP-1*, since they didn't work for IFSSA, IFSSR, or {DL,EC}SSR-PV, and correct definitions were already in 1363a-2004 E.3.

— E.4 *Representing private keys as octet strings*: New section.— E.4.1 *Data format for DL/EC private keys*: New section.— E.4.2 *Data format for IF private keys*: New section, with *Editor's Notes*.