

PAKZ Proposal

November 29, 2005

This document describes proposed changes to P1363.2 D22 APKAS-PAKZ draft version D22 that addresses an attack on PAKZ. This proposal corresponds to the [GMR05] proposal discussed in the August 2005 meeting.

The proposed changes include the following:

- 8.2.13 PVDGP-PAKZ: Add $H_u = Hash_{SK}(u)$ as a new output, to be stored by the Server.
- 9.7 APKAS-PAKZ – Key confirmation:
 - Server sends H_u to the Client.
 - Client aborts when H_u doesn't match $Hash_{SK}$ of his retrieved u .
 - The *Sign* and *Verify* operations are performed on $(o_{ID} || w_C || w_S)$, using a new o_{ID} parameter that may include party identifiers, instead of just on (w_S) .
 - The signature sent to the Server is no longer masked.
 - Key confirmation messages use π_m^{-1} is used instead of π_m , to reduce Server computation when storing π_m^{-1} instead of π_m .
- 14.6 *Converting between signatures and their octet string representations*: Deleted this section since these functions no longer need to be normative and they're informatively specified in E.3.
- D.5.5.6.2 *Reduction arguments*: Added summary of security proof in [GMR05].
- Annex G *Bibliography*: Added [GMR05] and [BPR00].

These differences are highlighted in the remainder of this document.

8.1.6 Summary of terms

S_1	<u>An octet string representing a signature s used in APKAS-PAKZ key confirmation</u>
S_C	<u>A masked signature octet string used in APKAS-PAKZ key confirmation</u>
$sLen$	<u>The length in octets of an agreed octet string representation of a signature in signature scheme S</u>
$Hash_{SK}$	<u>A hash function used on signature private keys used in APKAS-PAKZ</u>
H_u	<u>An octet string representing the hash of a signature private key u used in APKAS-PAKZ</u>
M	<u>A message octet string used in APKAS-PAKZ</u>
o_{ID}	<u>An octet string parameter used in APKAS-PAKZ</u>

8.2.5 PEPKGP-PAK

{DL,EC}PEPKGP-PAK is {Discrete Logarithm, Elliptic Curve} Password-Entangled Public Key Generation Primitive, version PAK. It is based on the work of [BMP00] and [MacK02]. This primitive derives a password-entangled public key from the party's private key and password value, using {DL,EC} domain parameters. This primitive is used by the schemes {DL,EC}BPKAS-PAK-CLIENT, {DL,EC}BPKAS-PPK-{CLIENT,SERVER} and {DL,EC}APKAS-PAKZ-CLIENT.

Input:

- The party's own private key s .
- The password-based mask group element π_m
- The {DL,EC} domain parameters (including g and r) associated with key s and group element π_m

Assumptions: Private key s and domain parameters are valid; Group element π_m generates the desired group of order r .

Output: The derived password-entangled public key w

Operation: The password-entangled public key w shall be computed by the following or an equivalent sequence of steps:

1. Compute a group element $w = (g \wedge s) * \pi_m$
2. Output w as the password-entangled public key

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

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] and [GMR05]. 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 Ss , which is the signature scheme option of APKAS-PAKZ that is associated with a private key u , a corresponding public key v , and an agreed format for representing u as a string of $uLen$ octets.
- 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.
- A hash function $Hash_{SK}$, which is the hash 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 $(\pi_m, v, o_{u\pi}, H_u)$, consisting of password-based mask group element π_m , public key v for signature scheme Ss , and octet string $o_{u\pi}$ of length $uLen$ containing the password-masked private key associated with v , and hashed private key octet string H_u (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 $\pi_m = \text{Redp}(\text{hex}(01) \parallel \pi)$ (See Note 1)
2. Compute an octet string $o_1 = \text{hex}(02) \parallel \pi$
3. Compute a password-mask octet string o_2 of length $uLen$ as $o_2 = \text{Mgf}(o_1, uLen)$
4. Generate a key pair (u, v) for signature scheme S_s
5. Compute octet string o_u of length $uLen$ as the agreed octet string representation of private key u
6. Compute password-masked private key octet string $o_{u\pi} = o_u \oplus o_2$
7. Compute a hashed private key octet string $H_u = \text{Hash}_{SK}(o_u)$
8. Output $(\pi_m, v, o_{u\pi}, H_u)$ (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—D22 PAKZ incorporates editorial changes to replace the 14.6 OS2SIGP-1 and SIG2OSP-1 functions with the previously defined 1363A-2004 E.3, which simplifies and corrects flaws and omissions in D21's treatment of various signature schemes. Normative section 14.6 now provides equivalents to E.3, etc., per our August 2005 meeting discussion.

Editor's Note—D22 PAKZ incorporates functional changes to use IFSSA, IFSSR, and {DL,EC}SSR-PV signature schemes, which were intended to be included but were incorrectly specified in earlier drafts.

Editor's Note—D22 PAKZ does not yet include the additional hash function proposed by MacKenzie discussed in the July 20 2005 teleconference.

Editor's Note—This D22+ proposed version is based on the PAK-Z+ protocol discussed in [GMR05] and the August 19, 2005 meeting.

{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] and [GMR05].

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 (see Note 7) and π_m as defined below,
- A signature scheme Ss associated with a private key u and corresponding public key v , which should be selected from the schemes specified in IEEE Std 1363a-2004 Clause 10, which defines or is otherwise associated with:
 - a signature generation operation $Sign(u,M)$ that outputs a signature S_C of a message octet string M ,
 - a signature verification operation $Verify(v,S_C,M)$ that outputs “valid” or “invalid”,
 - a format for representing an Ss signature as fixed length string of $sLen$ octets, which should be a format described in 14.6 that defines $sLen$ in accordance with associated signature scheme options and parameters (see Note 5),
 - a format for representing Ss private key u as a fixed-length string of $uLen$ octets, which should be a format described in 14.6 14.7 that defines $uLen$ in accordance with associated signature scheme options and parameters (see Note 5),
 - if Ss is a DL or EC scheme, Ss is associated with the same domain parameters (including k , q , r , and g).
- 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 hash function $Hash_{SK}$, which should be chosen from the hash functions in 14.1. The same $Hash_{SK}$ parameter shall be used with {DL,EC}PVDGP-PAKZ.
- A message parameter octet string o_{ID} (see Note 7)
- 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 octet string π_m ,
 - the public key v associated with the signature scheme Ss , 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 , and
 - the hashed private key octet string H_u containing $Hash_{SK}(o_u)$.

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_t

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_t

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 , recover private key u , and verify that u is correct, 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 $o_\pi = \text{GE2OSP-X}(\pi_m \wedge (-1))$
3. Compute $o_3 = \text{KCF1}(\text{hex}(03) \parallel o_{ID}, w_C, w_S, Z, o_\pi)$
4. If $o_3 \neq o_S$, output “invalid” and stop.

Editor’s Note—For this D22+ proposal, the Working Group considered whether the Client’s key confirmation steps 1 through 4 could be omitted, as being effectively replaced by the check for the correct value of H_u in step 9. These steps were retained to align with the security analysis in [GMR05].

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

(Mandatory) The Client recovers private signature key u and verifies that it is correct as follows:

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

5. Receive octet string A_S , a hidden and password-masked private signature key, and hashed private key octet string H_u from the Server
6. Compute $o_7 = \text{KCF1}(\text{hex}(05) \parallel o_{ID}, w_C, w_S, Z, o_\pi)$
7. Compute $o_5 = \text{Mgf}(o_7, uLen)$
8. Compute an octet string representation of the signature private key $o_u = A_S \oplus o_5 \oplus \text{Mgf}(\text{hex}(02) \parallel \pi, uLen)$
9. If $\text{Hash}_{SK}(o_u) \neq H_u$, output “invalid” and stop.
10. Compute signature private key u from the agreed octet string representation o_u
11. Optional If u is not a valid private key for the *Sign* operation, output “invalid” and stop. (See Note 6)

Note—Step 9 must be performed before the Client reveals information derived from u .

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

12. Compute $M = o_{ID} \parallel \text{GE2OSP-X}(w_C) \parallel \text{GE2OSP-X}(w_S)$
13. Compute signature $S_C s = \text{Sign}(u, M-w_S)$
12. Convert signature s into an octet string S_1 using the agreed representation of a signature

- ~~13.~~ Compute $o_8 = \text{KCF1}(\text{hex}(06), w_C, w_S, Z, o_\pi)$
- ~~14.~~ Compute $o_6 = \text{Mgf}(o_8, sLen)$
- ~~15.~~ Compute $S_C = S_1 \oplus o_6$
- ~~1416.~~ Send S_C to the server (see Note 8)

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

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

4. Compute $o_7 = \text{KCF1}(\text{hex}(05) \parallel o_{ID}, w_C, w_S, Z, o_\pi)$
5. Compute $o_5 = \text{Mgf}(o_7, uLen)$
6. Compute $A_S = o_{u\pi} \oplus o_5$
7. Send A_S and H_u to the Client
8. Receive signature octet string S_C from the Client (see Note 8)
- ~~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 an octet string representation of a signature $S_1 = S_C \oplus o_6$
- ~~12.~~ Convert octet string S_1 into a signature s based on the agreed representation of a signature
- ~~9.~~ Compute $M = o_{ID} \parallel \text{GE2OSP-X}(w_C) \parallel \text{GE2OSP-X}(w_S)$
- ~~1013.~~ If $\text{Verify}(v, \underline{S_C}, M) = \text{"invalid"}$, 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 desired 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 $\pi_m \wedge (-1)$ to optimize efficiency.

5—The parties must agree on a fixed size $sLen$ for the length of the agreed octet string format of signatures. When using a {DL,EC} signature scheme, $sLen$ depends on the scheme option domain parameter r for the recommended signature formats (see 14.6.1, 14.6.2). When using {DL,EC}SSR-PV, $sLen$ further depends on a fixed length for $cLen$, the length of message representative octet string C (see 14.6.2), which depends on the selected method for message encoding (see {DL,EC}SSR-PV Signature generation operation in IEEE 1363a-2004 10.5.2). When using an IF

~~signature scheme with the recommended signature format (see 14.6.3), the parties may need to agree on the size of the IF modulus n in order to determine $sLen$.~~

~~56—The parties must agree on a fixed size $uLen$ for the length of the agreed octet string format of signature private keys. When using a {DL,EC} signature scheme, $uLen$ depends on the scheme option domain parameter r for the recommended private key format (see 14.6.14.7.4). When using an IF signature scheme with the recommended private key format (see 14.6.24.7.2), the parties need to agree on the size of the IF modulus n associated with (u,v) , and agree on a specific representation of private keys as an ordered set of integers (see *Keys in the IF family* IEEE 1363-2000 8.1.3), in order to determine $uLen$.~~

~~6—The check for u being a valid private key in Client key confirmation Step 11 is not necessary for the security of the scheme, but it may be needed to ensure that $Sign(u, M)$ is a well-defined operation. Note, however, that IEEE Std 1363-2000 defines IF signature scheme private key operations as assuming that the private key is valid, but it does not define a validation method.~~

~~7—To be aligned with the PAK-Z+ protocol described in [GMR05], message parameter oID should include identifiers for the client and server, and password value π should include oID .~~

~~8—Formats for representing signatures as octet strings are given in IEEE Std 1363A-2004 E.3.~~

8.2.13 PVDGP-PAKZ [Drawings]

```

{ $\pi_m, v, o_u, H_u$ } = PVDGP-PAKZ( $\pi$ )
{
   $\pi_m = Redp(hex(01) \parallel \pi)$ 
  ( $u, v$ ) = a generated signature key pair, where  $uLen$  is the private key length in octets
   $o_u = u$ , in an octet string of length  $uLen$ 
   $o_{u\pi} = Mgf(hex(02) \parallel \pi) \oplus o_u$ 
   $H_u = Hash_{SK}(o_u)$ 
}
    
```

9.7 APKAS-PAKZ [Drawings]

Editor's Note—The D22 PAKZ summary has not yet been updated with the July 2005 proposal.

	<i>Client</i>		<i>Server</i>
Enrollment:			
PVDGP-PAKZ	$\pi_m = Redp(hex(01) \parallel \pi)$ $o_2 = Mgf(hex(02) \parallel \pi, uLen)$ Generate (u, v) sig keys $H_u = Hash_{SK}(o_u)$ $o_{ID} = ClientID \parallel ServerID$ $o_{u\pi} = o_2 \oplus u$	$\pi_m, v, o_{u\pi}, H_u, o_{ID}$ \rightarrow	$\pi_m, v, o_{u\pi}, H_u, o_{ID}$
Key Agreement:			
PVDGP-PAKZ	$\pi_m = Redp(hex(01) \parallel \pi)$		$s \in_R [1, r-1]$
PEPKGP-PAK	$w_C = (g \wedge s) * \pi_m$	$w_C \rightarrow$	Abort if $w_C \notin$ parent group
PKGP-DH	Abort if $w_S \notin$ parent group	$\leftarrow w_S$	$w_S = (g \wedge s)$
SVDP-PAK1-CLIENT	$z = w_S \wedge s$		
SVDP-PAK2			$z = (w_C * \pi_m^{-1}) \wedge s$

NOTE—The key agreement operation of APKAS-PAKZ has unilateral commitment from Client. Client must first verify key confirmation from Server, before using z.

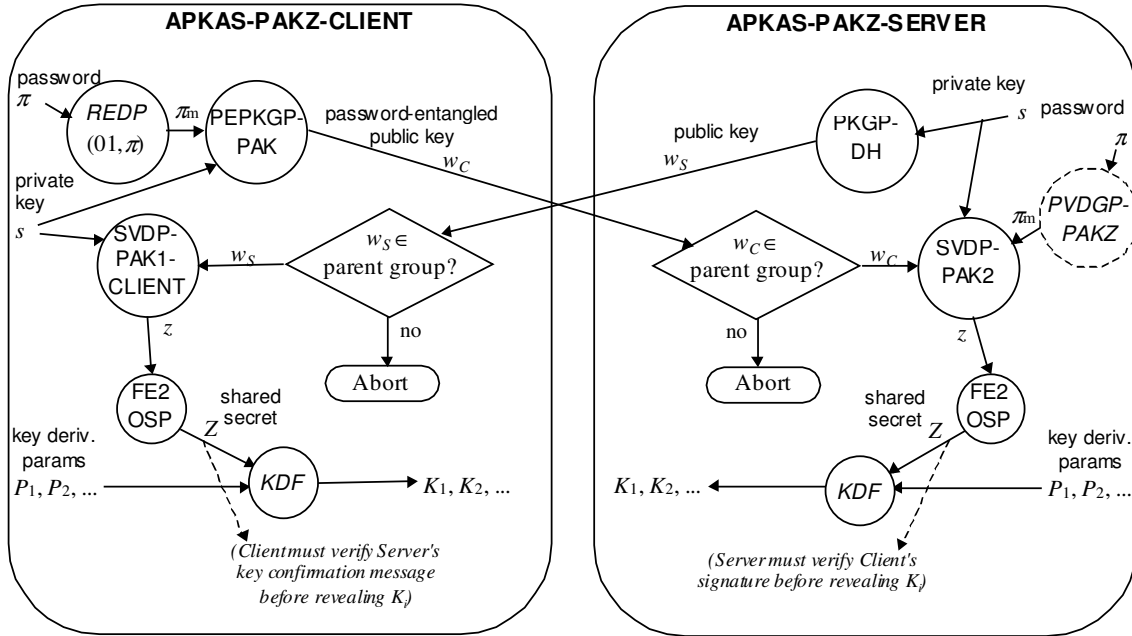


Figure 9.7.2—APKAS-PAKZ key agreement operation

(PAKZ continues with Key Confirmation on the following page.)

9.7 APKAS-PAKZ (continued)

Key confirmation of Server:	$o_3 = \text{KCF1}(03, w_C, w_S, z, \pi_n^{(-1)})$ $o_7 = \text{KCF1}(05, w_C, w_S, z, \pi_n^{(-1)})$ $o_8 = \text{KCF1}(06, w_C, w_S, z, \pi_m)$ $o_5 = \text{Mgf}(o_7, uLen)$ $o_6 = \text{Mgf}(o_8, sLen)$ Abort if $o_3 \neq o_5$	$o_S = \text{KCF1}(03, w_C, w_S, z, \pi_n^{(-1)})$ $o_7 = \text{KCF1}(05, w_C, w_S, z, \pi_n^{(-1)})$ $o_8 = \text{KCF1}(06, w_C, w_S, z, \pi_m)$ $o_5 = \text{Mgf}(o_7, uLen)$ $o_6 = \text{Mgf}(o_8, sLen)$		
Key confirmation of Client:	$o_u = A_S \oplus o_5 \oplus \text{Mgf}(\text{hex}(02) \parallel \pi, uLen)$ Abort if $\text{Hash}_{SK}(o_u) \neq H_u$ Convert octet string o_u into u (opt) Abort if u is invalid $M = o_{ID} \parallel w_C \parallel w_S$ $S_C s = \text{Sign}(u, M \parallel s)$ Convert s into octet string S_1 $S_C = S_1 \oplus o_6$	$\leftarrow o_S$ $\leftarrow A_S H_u$ $A_S = o_{u\pi} \oplus o_5$ $S_C \rightarrow$	$S_1 = S_C \oplus o_6$ Convert octet string S_1 into s $M = o_{ID} \parallel w_C \parallel w_S$ Abort if $\text{Verify}(v, S_C s, M)$ fails	

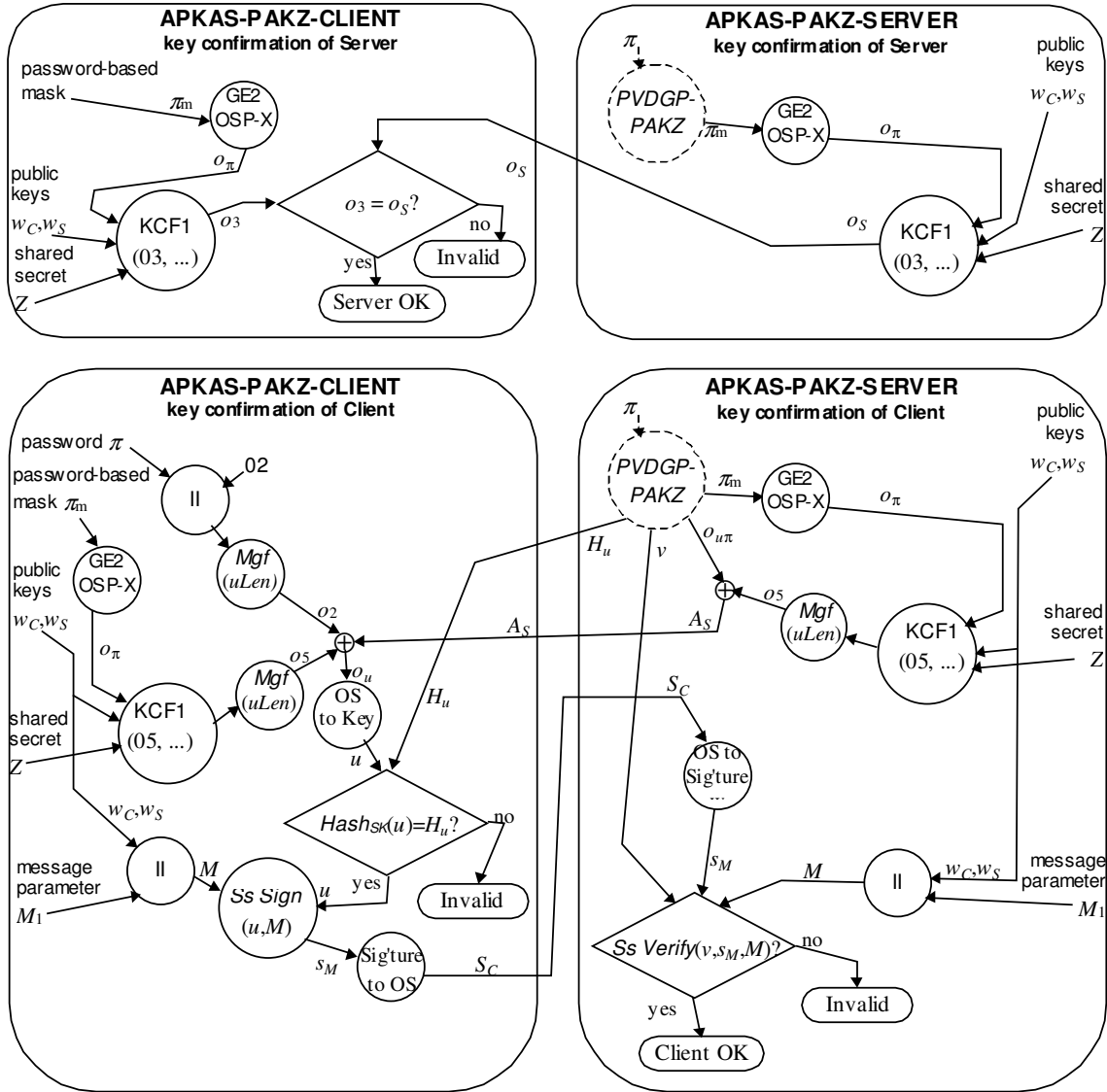


Figure 9.7.3—APKAS-PAKZ key confirmation operation

14.6 Converting between signatures and their octet string representations

14.6 14.7 Converting between private keys and their octet string representations

14.7 14.8 Multiplier value creation functions for APKAS-SRP6

Editor's Note—This D22+ proposal deletes Subclause 14.6 *Converting between signatures and their octet string representations*, as these normative specifications are no longer required for APKAS-PAKZ. Subclauses 14.7 and 14.8 should be renumbered appropriately.

Annex B (Normative) Conformance

B.3 Examples

B.3.6 ECAPKAS-PAKZ-SERVER

A module claiming conformance to the ECAPKAS-PAKZ-SERVER key agreement and key confirmation operations, under one set of reasonable conditions, might document its conformance as follows:

“Conforms with P1363.2 ECAPKAS-PAKZ-SERVER / ECPVDGP-PAKZ / ECREDP-2(MGF1(SHA-1, 160), any valid g_a and g_b) / ECSSA(ECSP-DSA(...), EMSA1) / MGF1(SHA-1, 160) / SHA-1 / KDF1(SHA-1) / KCF1(SHA-1) key agreement and key confirmation operations over the region where ...”

An indented form that highlights the structure of this statement is as follows:

“Conforms with P1363.2 ECAPKAS-PAKZ-SERVER /
 ECPVDGP-PAKZ /
 ECREDP-2(
 MGF1(SHA-1, 160) /
 any valid g_a and g_b
) /
 ECSSA(
 ECSP-DSA(...),
 EMSA1
) /
 MGF1(SHA-1, 160) /
 SHA-1 /
 KDF1(SHA-1) /
 KCF1(SHA-1)
 key agreement and key confirmation operations over the region where ...”

Annex C (Informative) Rationale

C.3 Schemes and Primitives

C.3.14 Why have multiple Augmented PKA schemes?

C.3.14.3 Why have PAKZ?

PAKZ is associated with a security reduction argument to the CDH assumption in the random oracle model.

Annex D (Informative) Security Considerations

D.5 Scheme-specific considerations

D.5.5 Considerations for specific password schemes (PKAS and PKRS)

D.5.5.6 Considerations for PAK schemes

D.5.5.6.2 Reduction arguments

Editor's Note—Add discussion and references for security reduction analysis of PAK₊ and PPK₊ and PAKZ.

A PAK-Z+ protocol and an argument for its security is presented in [GMR05]. The argument is based on the random oracle model of [BPR00], the Computational Diffie Hellman assumption, and the assumption that signature scheme S_s is existentially unforgeable against adaptive chosen message attacks. APKAS-PAKZ is a reasonably close instantiation of PAK-Z+ when the shared password-based value π and the message parameter o_{ID} incorporate identifiers for the Client and Server.

Annex G (Informative) Bibliography

[BMP00] V. Boyko, P. MacKenzie & S. Patel, “Provably Secure Password Authenticated Key Exchange Using Diffie-Hellman”, Advances in Cryptology - EUROCRYPT 2000, Preneel, B., (Ed.), May 14-18, 2000.

[GMR05] C. Gentry, P. MacKenzie and Z. Ramzan, “PAK-Z+”, Contribution to the IEEE P1363 Working Group, August 15, 2005. Available at <http://grouper.ieee.org/groups/1363/passwdPK/contributions.html#GMR05b>.

[MacK01a] P. MacKenzie, “More Efficient Password-Authenticated Key Exchange”, LNCS 2020: Topics in Cryptology -- CT-RSA 2001, April 8-12, 2001 Proceedings, pp. 361-377, 2001, Springer-Verlag.

[MacK02] P. MacKenzie, “The PAK suite: Protocols for Password-Authenticated Key Exchange”, a P1363.2 submission to the IEEE P1363 Working Group, April 24, 2002.

[MacK02b] P. MacKenzie, “Submission Update to PAK Schemes”, contribution to IEEE P1363 Working Group, received September 9, 2002. Available at <http://grouper.ieee.org/groups/1363/passwdPK/contributions.html#Mac02b>.

[BPR00] M. Bellare, D. Pointcheval and P. Rogaway, “Authenticated key exchange secure against dictionary attacks”, EUROCRYPT 2000 (LNCS 1807), pp. 139-155, 2000.