

PAKZ Proposal

August 3, 2005

This is a summary of the changes to P1363.2 D21 APKAS-PAKZ that were discussed in the July 2005 teleconference to address the recent attack on PAKZ described by MacKenzie. Some open questions regarding further clarification or simplification of the method are presented in *Editor's Notes*.

This document does not include the changes for (re)defining the octet string conversion functions to accomodate various signature schemes. A separate document describes those and further editorial changes.

Summary of changes between D21 and this proposal

- PVDGP-PAKZ: Added new H_u output, computed using new $Hash_{SK}(u)$ operation.
- APKAS-PAKZ: Server sends H_u to the Client, who aborts if it doesn't match $Hash_{SK}(u)$.

Potential open issues

The Working Group should consider:

- whether to delete or amend step 11 (check for valid u) to be optional, since for the IF case, no validity test is defined in 1363-2000;
- whether this step 11 is really needed, now that the Client checks H_u . If it works as intended, only a party that knows π can make the Client use an invalid u that passes the H_u test. Yet, this step may be desired merely to ensure that $Sign(u, \dots)$ is well-defined;
- whether the check for H_u should be considered part of verification of the Server, and whether this should state that it must occur before the Client uses Z for other purposes, as noted for Step 4; and
- whether steps 1 through 4 may be omitted, being effectively replaced by the check for H_u .

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 Ss , 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 , 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
- 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 S_s , 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 $\pi_m = Redp(hex(01) \parallel \pi)$ (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 $uLen$ using I2OSP(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 $H_u = Hash_{SK}(o_u)$
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.

9.7 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± proposed version is based on the proposal by MacKenzie discussed in the July 20 2005 teleconference.

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 Ss associated with a private key u and corresponding public key v , which may be either {DL,EC}SSA, {DL,EC}SSR, or {DL,EC}SSR-PV (see IEEE Std 1363a-2004), which defines:
 - a signature generation operation $Sign$,
 - a signature verification operation $Verify$,
 - 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
- 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.

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 the private key u that corresponds to v .
 - 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_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, and hashed private key octet string H_u 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 $o_u = A_S \oplus o_5 \oplus \text{Mgf}(\text{hex}(02) \parallel \pi, uLen)$
9. If $\text{Hash}_{SK}(o_u) \neq H_u$, output "invalid" and stop.
- ~~10~~9. Compute $u = \text{OS2IP}(o_u)$
- ~~11~~10. If u is not a valid private key for the *Sign* operation, output "invalid" and stop.

Editor's Note—Consider deleting or amending step 11 (check for valid u) to be optional, since for the IF case, no validity test is defined in 1363-2000. Also consider whether this step is needed, now that the Client checks H_u . If it works as intended, only a party that knows π can make the Client use an invalid u that passes the H_u test. Yet, this step may be desired merely to ensure that $\text{Sign}(u, \dots)$ is well-defined.

Editor's Note—Consider whether the check for H_u should be considered part of verification of the Server, and whether this should state that it must occur before the Client uses Z for other purposes, as noted for Step 4.

Editor's Note—Consider whether steps 1 through 4 may be omitted, being effectively replaced by the check for H_u .

- ~~12~~¹⁴. Compute $s = \text{Sign}(u, w_S)$
- ~~13~~¹⁵. Compute $S_1 = \text{Sig2osp}(s)$
- ~~14~~¹⁶. Compute $o_8 = \text{KCF1}(\text{hex}(06), w_C, w_S, Z, o_\pi)$
- ~~15~~¹⁷. Compute $o_6 = \text{Mgf}(o_8, sLen)$
- ~~16~~¹⁸. Compute $S_C = S_1 \oplus o_6$
- ~~17~~¹⁹. 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 and H_u 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 $S_1 = S_C \oplus o_6$
12. Compute $s = \text{Os2sigp}(S_1)$
13. If $\text{Verify}(v, s) = \text{"invalid"}$, 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.