

## AMP Proposal

December 8, 2005

The APKAS-AMP scheme in the current P1363.2 draft (D22, Oct. 22, 2005) is vulnerable to the attack described in [Kw05] in which an enemy can masquerade as the Client using  $v_\pi$ . This document describes the latest proposal to repair this flaw discussed in the December 7 2005 teleconference. Among several of the versions of AMP considered by the Working Group, listed in Appendix below, this proposal is closest to the AMP2 protocol of [Kw03a]. The remainder of this document highlights the differences between this proposal and D22 APKAS-AMP.

## Proposed amendment of APKAS-AMP D22

### 8.1.6 Summary of terms

$Hash_{wl}$       A hash function used in APKAS-AMP  
 $o_{ID}$             An octet string parameter used in APKAS-AMP

### 8.2.4 PEPKGP-AMP-SERVER

{DL,EC}PEPKGP-AMP-SERVER is {Discrete Logarithm, Elliptic Curve} Password-Entangled Public Key Generation Primitive, version AMP for Server. PEPKGP-AMP-SERVER is based on the work of [Kw03a]~~[Kw00]~~ and [Kw02]. This primitive derives a password-entangled public key from password verification data, the Server's private key, and a Client's public key, using {DL,EC} domain parameters.

This primitive is parameterized by the following choices:

- A hash function  $Hash_{wl}$  (see Note 1), which should be one of the hash functions in 14.1.
- An octet string  $o_{ID}$  that may provide additional input to  $Hash_{wl}$ .

The Client and Server shall use the same  $Hash_{wl}$  and  $o_{ID}$  parameters with PEPKGP-AMP-SERVER and SVDP-AMP-CLIENT.

#### Input:

- The Server's private key  $s$
- The password verification data  $v_\pi$ , a password-limited public key
- The Client's public key value  $w_C$
- The {DL,EC} domain parameters (including  $r$ ) associated with the keys  $s$ ,  $v_\pi$ , and  $w_C$

**Assumptions:** Private key  $s$ , password-limited public key  $v_\pi$ , and associated domain parameters are valid;  $w_C$  is an element of the parent group.

**Output:** The derived password-entangled public key value  $w_S$ , which is an element of the parent group

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

1. Compute  $o_C = \text{GE2OSP-X}(w_C)$
2. Compute  $o_1 = \text{Hash}_{w_1}(o_C \parallel o_{1D})$
3. Compute  $i_1 = \text{OS2IP}(o_1)$
4. Compute  $w_S = ((w_C \wedge i_1) * v_\pi) \wedge s$
5. Output  $w_S$

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

#### NOTES

1—See D.5.5.1.9 *Criteria for selecting Hash<sub>w1</sub> in AMP*.

### 8.2.11 PVDGP-AMP

{DL,EC}PVDGP-AMP is {Discrete Logarithm, Elliptic Curve} Password Verification Data Generation Primitive, version AMP. PVDGP-AMP is based on the work of [Kw03a][Kw00]. This primitive derives password verification data and associated values from a party's password value, using {DL,EC} domain parameters. This primitive derives the password-limited private key used in {DL,EC}APKAS-AMP-CLIENT and derives the password verification data used in {DL,EC}APKAS-AMP-SERVER.

This primitive is parameterized by the following choices:

- A hash function  $\text{Hash}_{PVD}$  (see Note 1), which should be one of the hash functions in 14.1 or MGF1 (see 14.2.1)

#### Input:

- The password-based octet string  $\pi$
- The {DL,EC} domain parameters (including  $g$  and  $r$ ) associated with  $\pi$

**Assumptions:** Domain parameters are valid.

**Output:** The password-limited private key  $u_\pi$  and associated password verification data consisting of password-limited public key  $v_\pi$

**Operation:** The password-limited private key  $u_\pi$  and verification data  $v_\pi$  shall be computed by the following or an equivalent sequence of steps:

1. Compute  $o_\pi = \text{Hash}_{PVD}(\pi)$
2. Compute  $u_\pi = \text{OS2IP}(o_\pi) \bmod r$
3. Compute  $v_\pi = g \wedge u_\pi$
4. Output  $u_\pi$  and verification data  $v_\pi$

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

#### NOTES

1—See D.5.4.18 *Range of password-limited private keys* for discussion of appropriate choices for  $\text{Hash}_{PVD}$ .

### 8.2.19 SVDP-AMP-CLIENT

{DL,EC}SVDP-AMP-CLIENT is {Discrete Logarithm, Elliptic Curve} Secret Value Derivation Primitive, version AMP for Client. It is based on the work of [Kw03a][Kw00] and [Kw02]. It may be used with the scheme {DL,EC}APKAS-AMP-CLIENT. This primitive derives a shared secret value from the Client's private key, the Client's password-limited private key  $u_\pi$ , and a Server's password-entangled public key. This primitive is used by the scheme {DL,EC}APKAS-AMP-CLIENT.

This primitive is parameterized by the following choices:

- A hash function  $Hash_{w,l}$  (see Note), which should be one of the hash functions in 14.1.
- An octet string  $o_{ID}$  that may provide additional input to  $Hash_{w,l}$ .

The Client and Server shall use the same  $Hash_{w,l}$  functions and  $o_{ID}$  parameters with PEPKGP-AMP-SERVER and SVDP-AMP-CLIENT.

#### Input:

- The Client's private key  $s$
- The Client's password-limited private key  $u_\pi$
- The Client's public key  $w_C$
- The Server's password-entangled public key  $w_S$
- The {DL,EC} domain parameters (including  $q$ ) associated with the values  $s$ ,  $u_\pi$ ,  $w_C$  and  $w_S$

**Assumptions:** Private keys  $s$  and  $u_\pi$  and associated domain parameters are valid.  $w_C$  and  $w_S$  are in the parent group.

**Output:** The derived shared secret value  $z$ , which is a field element of  $GF(q)$

**Operation:** The shared secret value  $z$  shall be computed by the following or an equivalent sequence of steps:

1. Compute  $o_C = \text{GE2OSP-X}(w_C)$
2. Compute  $o_1 = \text{Hash}_{w,l}(o_C \parallel o_{ID})$
3. Compute  $i_1 = \text{OS2IP}(o_1)$
4. Compute  $i_2 = ((s + 1) / ((s \times i_1) + u_\pi)) \bmod r$
5. Compute  $z_g = w_S^{i_2}$
6. Compute  $z = \text{GE2SVFEP}(z_g)$
7. Output  $z$

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

NOTE—See D.5.5.1.9 Criteria for selecting  $Hash_{w,l}$  in AMP.

### 8.2.20 SVDP-AMP-SERVER

{DL,EC}SVDP-AMP-SERVER is {Discrete Logarithm, Elliptic Curve} Secret Value Derivation Primitive, version AMP for Server. This primitive derives a shared secret value from a Client's public key, the Server's private key, and the domain parameter  $g$ . This primitive is used by the scheme {DL,EC}APKAS-AMP-SERVER. APKAS-AMP is based on the work of [Kw03a][Kw00] and [Kw02].

**Input:**

- The Server's own private key  $s$
- The Client's public key  $w_C$
- The {DL,EC} domain parameters (including  $q$  and  $g$ ) associated with the keys  $s$  and  $w_C$

**Assumptions:** Private key  $s$  and associated DL domain parameters are valid.  $w_C$  is in the parent group.

**Output:** The derived shared secret value  $z$ , which is a field element of  $GF(q)$ , or "invalid"

**Operation:** The shared secret value  $z$  shall be computed by the following or an equivalent sequence of steps:

1. Compute  $z_g = (w_C * g) ^ s$
2. If the order of  $z_g$  is unacceptably small, output "invalid" and stop. (See Note)
3. Compute  $z = \text{GE2SVFEP}(z_g)$
4. Output  $z$

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

NOTE—See D.5.5.1.4 *Server validation of shared secret key* for how and why one should validate that  $z_g$  is not a small order element and the meaning of "unacceptably small".

## 9. Password-Authenticated Key Agreement Schemes

### 9.5 APKAS-AMP

*Editor's Note*—An attack on this D20/D21 version of APKAS-AMP was described in T. Kwon's June 8 2005 contribution. An adversary who obtains  $v_\pi$  (but doesn't know  $\pi$ ) can negotiate the shared key  $z$ , and thereby masquerade as the Client, using the following steps:

1. Obtain a private key  $s$
2. Compute  $w_C = v_\pi ^{(s-1)} * g^{(-s)}$
3. Send  $w_C$  to the server
4. Receive  $w_S$  from the server
5. Compute  $z = \text{GE2SVFEP}(w_S ^{\Delta((s-1)/s)})$

The Working Group will discuss proposals for resolving this issue in the July 2005 teleconference.

{DL,EC}APKAS-AMP-{CLIENT,SERVER} is {Discrete Logarithm, Elliptic Curve} Augmented Password-Authenticated Key Agreement Scheme, version AMP for {Client, Server}. It is based on the work of [Kw03a][Kw00] and [Kw02]. The Server uses a password-limited public key  $v_\pi$  as password verification data that was derived using PVDGP-AMP with the input value  $\pi$ , which is a password-based octet string used by the Client.

#### 9.5.1 Scheme options

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

*For CLIENT only:*

- A password-based octet string  $\pi$

For SERVER only:

- A password-limited public key  $v_\pi$  that was generated using {DL,EC}PVDGP-AMP with the same parameter and input values as used in the Client's key agreement operation.

For both CLIENT and SERVER:

- Primitives for password verification data generation, public key generation and secret value derivation, which shall be PVDGP-AMP, PKGP-DH, PEPKGP-AMP-SERVER, SVDP-AMP-CLIENT, and SVDP-AMP-SERVER, and their associated parameters. The AMP Client and Server shall use the same  $Hash_{w_j}$  and  $o_{jD}$  parameters with PEPKGP-AMP-SERVER and SVDP-AMP-CLIENT.
- A set of valid {DL,EC} domain parameters (including  $q$  and  $r$ ) associated with  $\pi$  and  $v_\pi$ . (see Note 3)
- 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

### 9.5.2 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:

#### 9.5.2.1 Key agreement for Client

1. Obtain private key  $s$ , a random integer in the range  $[1, r-1]$  (See D.5.4.17)
2. Compute public key  $w_C = \{\text{DL,EC}\}\text{PKGP-DH}(s)$
3. Send  $w_C$  to the Server

NOTE—Step 3 shall occur before Step 4, since the Server uses  $w_C$  to compute  $w_S$ .

4. Receive password-entangled public key value  $w_S$  from the Server
  - 4.1 If  $w_S$  is not in the parent group, output “invalid” and stop. (See Note 2)
  - 4.2 If the order of  $w_S$  is unacceptably small, output “invalid” and stop. (See Note 2)
  - 4.3 (Optional) If  $w_S$  is not a valid public key, output “invalid” and stop. (See Note 5)
5. Compute password-limited private key  $u_\pi$  using the steps described in {DL,EC}PVDGP-AMP( $\pi$ )
6. Compute field element  $z$  using {DL,EC}SVDP-AMP-CLIENT( $s, u_\pi, w_C, w_S$ )
7. Compute octet string  $Z = \text{FE2OSP}(z)$
8. For each key derivation parameter  $P_i$ , derive a shared secret key  $K_i$  from the shared secret octet string  $Z$  and  $P_i$  using  $K_i = \text{Kdf}(Z, P_i)$ .
9. Output derived keys  $K_1, K_2, \dots, K_r$

#### 9.5.2.2 Key agreement for Server

1. Obtain private key  $s$ , a random integer in the range  $[1, r-1]$  (See D.5.4.17)
2. Receive public key value  $w_C$  from the Client
  - 2.1 If  $w_C$  is not an element of the parent group, output “invalid” and stop. (See Note 2)
  - 2.2 (Optional) If  $w_C$  is not a valid public key, output “invalid” and stop. (See Note 4.3)

3. Generate password-entangled public key value  $w_S$  using  $\{DL,EC\}PEPKGP-AMP-SERVER(s, v_p, w_C)$ 
  - ~~3.1 If the order of  $w_S$  is unacceptably small, let  $w_S$  = a random valid public key. (See Note 2)~~
4. Send  $w_S$  to the Client
5. Compute field element  $z$  using  $\{DL,EC\}SVDP-AMP-SERVER(s, w_C)$ 
  - 5.1 If the SVDP function in Step 5 outputs “invalid”, output “invalid” and stop. (See Note 2)
6. Compute octet string  $Z = FE2OSP(z)$
7. For each key derivation parameter  $P_i$ , derive a shared secret key  $K_i$  from the shared secret octet string  $Z$  and  $P_i$  using  $K_i = Kdf(Z, P_i)$ .

NOTE—The Server shall confirm the Client’s knowledge of shared secret  $Z$  before any derived keys are used. Key confirmation is described in 9.5.3.

8. Output derived keys  $K_1, K_2, \dots, K_i$

### 9.5.3 Key confirmation operation

It is mandatory in BPKAS-AMP for the Server to confirm the Client’s knowledge of the shared secret  $Z$ , before the Server uses  $Z$  or any derived shared secrets  $K_i$  for other purposes. Explicit confirmation of the Server’s knowledge of  $Z$  to the Client is optional.

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

#### 9.5.3.1 Key confirmation for Server

*Mandatory:*

- 1.1 Receive octet string  $o_C$  from the Client
- 1.2 Compute  $o_4 = KCF1(hex(04), w_C, w_S, Z, "")$
- 1.3 If  $o_4 \neq o_C$ , output “invalid” and stop.

NOTE—Step 1.3 shall occur before Step 2.1, and before any other use of shared keys  $K_i$  that are derived from the Server’s key agreement operation.

*Optional:*

- 2.1 Compute  $o_S = KCF1(hex(03), w_C, w_S, Z, "")$
- 2.2 Send  $o_S$  to the Client

#### 9.5.3.2 Key confirmation for Client

*Mandatory:*

- 1.1 Compute  $o_C = KCF1(hex(04), w_C, w_S, Z, "")$
- 1.2 Send  $o_C$  to the Server

*Optional:*

- 2.1 Receive octet string  $o_S$  from the Server
- 2.2 Compute  $o_3 = KCF1(hex(03), w_C, w_S, Z, "")$
- 2.3 If  $o_3 \neq o_S$ , 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-AMP is a unilateral commitment scheme, where the Client does not provide a commitment to the password during the key agreement operations. In either the scheme or the invoking application protocol, the Server must verify the Client's proof of knowledge of the agreed key before revealing any information derived from the agreed key. See D.5.4.9 for discussion of the limitations on the use of unilateral commitment schemes in application protocols.

2—~~The required public key validation (Client Steps 4.1 and 4.2, and Server Step 2.1) may be implemented in a variety of ways. For example, in the DL settings recommended in Note 3, it is sufficient for the Client to ensure that  $w_C$  is an integer in the range  $[2, q-2]$  and for the Server to ensure that  $w_C$  is an integer in the range  $[1, q-1]$ . See D.5.5.1 for discussion of the security considerations for AMP, including the reasons for these steps of checking for acceptable values and replacing an unacceptably small order  $w_C$  with a random valid public key, the meaning of "unacceptably small" order, and a potential timing attack related to Step 3.1 of the Key agreement for Server operation.~~

3—~~For DL settings, it is recommended that  $q-1$  have no factors in the range  $[3, r-1]$ . For EC settings, it is recommended that  $k$  be either 1, 2, or 4. See D.5.5.1.1 for how domain parameter choice is related to public key validation, performance, security, and alignment with other standards.~~

4<sup>3</sup>—~~When using domain parameters as recommended in Note 3, the Server's does not need to ensure that validate  $w_C$  is a valid public key in Step 2.2, during key agreement depends on the domain parameters. It is optional for the Server may perform this step to abort when the received value  $w_C$  is determined to be an invalid public key without adversely affecting the method, although this. This step is is both unnecessary and may require significant added computation, such as when using certain recommended settings, such as a DL  $GF(p)$  setting with a "safe prime"  $p$ . When using domain parameters other than as recommended in Note 3, However, this step 2.2 or some other alternative further validation may be necessary to prevent Pohlig-Hellman decomposition attack in other settings. See D.5.5.1.1 and D.5.5.1.3 for discussion of validating acceptable values for  $w_C$  and the related issues of recommended domain parameter selection, performance, and security. See D.5.5.1 for discussion of how public key validation is related to domain parameter choice, performance, and security in APKAS-AMP.~~

5—~~Other than ensuring that the order of  $w_C$  is acceptably large in Step 2.1, the Client does not need to ensure that  $w_C$  is a valid public key. However, the Client may perform the stricter validity check in optional Step 2.1 without adversely affecting the method.~~

## D.5 Scheme-specific considerations

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

#### D.5.5.1 Considerations for APKAS-AMP

APKAS-AMP and its related primitives perform validity checks on several received, transmitted and computed values to prevent a variety of potential attacks. Some of these potential attacks are described here. Others may be found in [Kw00], [Kw02], and [WW04].

*Editor's Note*—Add reference for general discussion of unilateral commitment.

##### D.5.5.1.1 Selection of domain parameters for AMP

For DL settings, it is strongly recommended that  $q-1$  has no factors in the range  $[3, r-1]$ , and for EC settings, it is recommended that  $k$  be either 1, 2, or 4~~of factor  $k$  not have any factors (other than a single factor of 2) that are smaller than  $r$ , in order to prevent the Pohlig-Hellman decomposition attack discussed in D.5.5.1.3~~D.5.4.6. Such settings may also simplify the required checks for unacceptably small order elements as discussed in D.5.5.1.2 and D.5.5.1.4. These recommendations are in alignment with those for a similar scheme in a draft of ISO/IEC 11770-4 [ISO05].

*Editor's Note*—Update the reference to [ISO05] as 11770-4 progresses.

##### D.5.5.1.2 Client validation of Server's public key

The Client tests the Server's public key  $w_s$  and aborts if it is not in the parent group or is an element of unacceptably small order. This prevents the Client from computing a small order  $z$ , which could allow the Server to guess  $z$  without knowing the password verification data value  $v_\pi$ . See D.5.4.6.2–D.5.4.5 for the meaning of “unacceptably small” and further discussion of small subgroup confinement. To avoid computation in validating that  $w_s$  is a large order element of the parent group, it is recommended that the domain parameters be selected as described in D.5.5.1.1 *Selection of domain parameters for AMP*.

### D.5.5.1.3 Server validation of Client's public key

The Server tests the Client's public key and aborts if it is not in the parent group. This prevents unexpected values from being passed to implementations of PEPKGP-AMP-SERVER or SVDP-AMP-SERVER, the latter of which might compute  $z = 0$  in a DL setting. PEPKGP-AMP-SERVER that could result in the Server computing  $z = 0$ .

Furthermore, the order of  $((w_C \wedge i_1) * v_\pi) (w_C * v_\pi)$  must not be a composite with multiple factors significantly smaller than  $r$ , to avoid the Pohlig-Hellman decomposition attack discussed in D.5.4.6. To preclude such attack, the Server may check  $w_C$  and abort when it is not a valid public key. Alternately, to ~~avoid~~ avoid computational expense of validating in checking  $w_C$ , it is recommended that the domain parameters be selected to preclude such attack, as described in D.5.5.1.1 *Selection of domain parameters for AMP*.

*Editor's Note*—In this D22+ version, sections D.5.5.1.4 through D.5.5.1.7 were removed in accordance with the [Kw05] proposal and subsequent Working Group discussion.

### D.5.5.1.4 Server validation of shared secret key

SVDP-AMP-SERVER checks the order of its computed shared secret group element  $z_g$  and returns “invalid” if the order is unacceptably small. Without this step, an enemy posing as a client could choose a small order element  $e$ , send  $w_C = e/g$  to the APKAS-AMP Server, and confine  $z_g$  to a small group, and thus determine the APKAS-AMP Server's value for  $z$  without knowledge of  $\pi$ . See D.5.4.6.2 for the meaning of “unacceptably small” and further discussion of small subgroup confinement.

### D.5.5.1.5 Special handling of Server's invalid public key

~~When PKAS AMP SERVER detects that PEPKGP AMP SERVER has computed small order value for  $w_s$ , it sets  $w_s$  to a random substitute valid public key, before sending  $w_s$  to the client. Without these steps, an enemy posing as a client could verify two guesses for the password  $(\pi, \pi_2)$  in a single run. He could compute  $w_C = g^{(-Hash(\pi_2))}$ , test  $w_s = 1$  to detect whether the password is  $\pi_2$ , and if not, test whether the server agrees on the value for  $z$  to detect whether the password is  $\pi$ . Or, in a similar attack, an enemy could choose a small order element  $e$  and compute  $w_C = e * g^{(-Hash(\pi_2))}$ .~~

### D.5.5.1.6 Potential timing attack for special handling of Server's invalid public key

~~When PKAS AMP SERVER detects an unacceptably small order  $w_s$ , and decides to set  $w_s$  to a random substitute value, it must do so in a way that does not reveal to the client that such a decision has been made. If the time it takes to return a substitute random  $w_s$  is different than the time it takes to return the genuine computed  $w_s$ , a malicious client might be able detect the difference in the time to make the extra guess for the password. See IEEE Std 1363-2000 D.7 *Implementation Considerations* for a broader discussion of error analysis and other threats to containment of sensitive information.~~

### D.5.5.1.7 Need for independence of Server's substitute public key

~~The random substitute value for the Server's public key  $w_s$  that is assigned in Step 5 of the Server's key agreement operation when must not be equal to the Server's agreed key value  $z := GE2SVFEP((w_C * g)^\wedge s)$ ,~~

and it must be computationally independent from  $z$ , so as to ensure that an attacking Client who has stolen  $v_\pi$  cannot derive  $z$  from  $w_S$ .

**D.5.5.1.5 Need for  $Hash_{w_I}$  in AMP**

The  $Hash_{w_I}$  function in APKAS-AMP (which was used in AMP<sup>+</sup> in [Kw00], and in AMP2 in [Kw03a]), prevents a vulnerability noted in [Kw05] that applied to an earlier draft version of APKAS-AMP that was based on AMP of [Kw02]. The earlier version permitted an enemy who obtained the password verification data to masquerade as the Client without performing a dictionary attack, thus removing the augmented benefit.

**D.5.5.1.6 Criteria for selecting  $Hash_{w_I}$  in AMP**

The  $Hash_{w_I}$  function should be selected such that it outputs at least  $\lceil \log_{256}(r) \rceil$  octets.

It is recommended in [Kw01] and [Kw03b] that Client and Server identifiers be included in the  $o_{ID}$  input parameter for  $Hash_{w_I}$ .

*Editor's Note*—Kwon may propose some additional material for D.5.5.1.10.

*Editor's Note*—Consider checking the rationale on performance, and the suggestion that SVDP-AMP-SERVER could set  $Hash_{w_2}$  as 1 for better performance when using domain parameters as stated in D.5.5.1.1.

**9.5 APKAS-AMP**

*Editor's Note*—The D22 APKAS-AMP summary has not yet been updated with the June 2005 proposal.

	<i>Client</i>		<i>Server</i>
<b>Enrollment:</b> PVDGP-AMP	$\pi = \text{salt} \parallel \text{pwd} \parallel \text{IDs} \dots$ $u_\pi = Hash_{PVD}(\pi)$ $v_\pi = g \wedge u_\pi$	$v_\pi \rightarrow$	$v_\pi$
<b>Key Agreement:</b> PKGP-DH PEPKGP-AMP-SERVER	$s \in_R [1, r-1]$ $w_C = g \wedge s$	$w_C \rightarrow$	$s \in_R [1, r-1]$ Abort if $w_C \notin$ parent group (opt) Abort if $w_C$ invalid $i_1 = Hash_{w_I}(w_C \parallel o_{ID})$ $w_S = ((w_C \wedge i_1) * v_\pi)^s$ If $o(w_S)$ too small, — $w_S =$ random public key
PVDGP-AMP SVDP-AMP-CLIENT	Abort if $w_S \notin$ parent group Abort if $o(w_S)$ too small (opt) Abort if $w_S$ invalid $u_\pi = Hash_{PVD}(\pi)$ $i_1 = Hash_{w_I}(w_C \parallel o_{ID})$ $z = w_S \wedge ((s+1) / (s \times i_1 + u_\pi))$	$\leftarrow w_S$	
SVDP-AMP-SERVER			$z = (w_C * g) \wedge s$ If $o(z)$ too small, Abort

NOTE—APKAS-AMP has only unilateral commitment from Server. Server must first verify key confirmation from Client before using  $z$ .

*Editor's Note*—See related Editor's notes in the main document regarding validity checks.

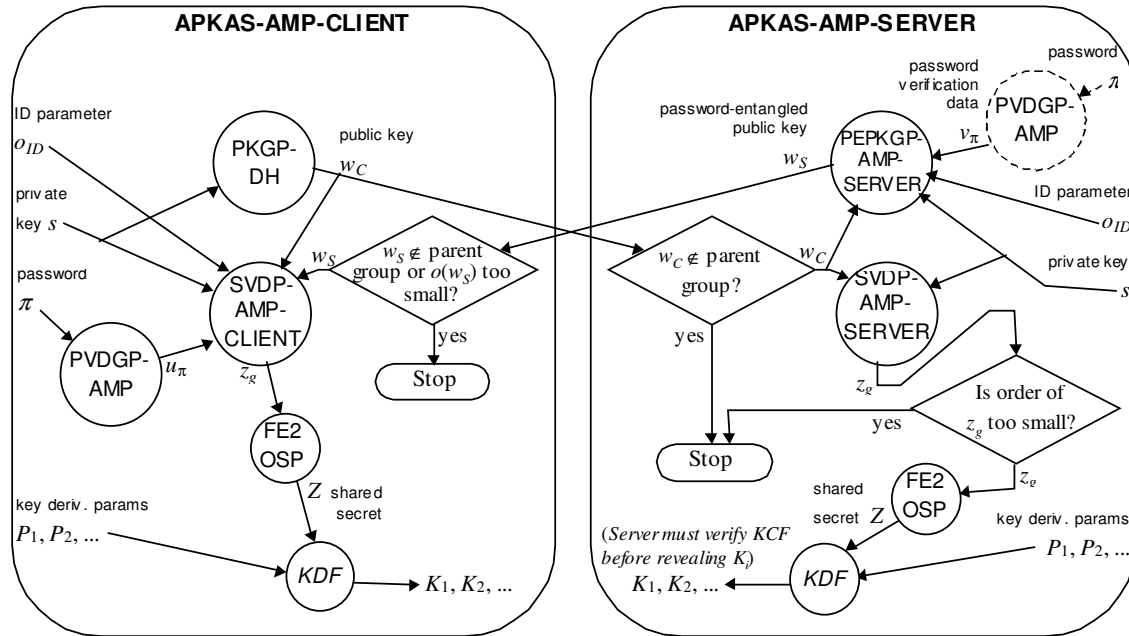


Figure 9.5.2—APKAS-AMP key agreement operation

## Annex G (Informative) Bibliography

[ISO05] ISO/IEC FDIS 11770-4, "Information technology -- Security techniques -- Key management -- Part 4: Mechanisms based on weak secrets", a standard draft under development, October 31, 2005. Status information available at

<http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=39723>

[Kw00] T. Kwon, "Ultimate Solution to Authentication via Memorable Password", May 30, 2000.

Submission to the IEEE P1363 Working Group, received June 22, 2000. Available at <http://grouper.ieee.org/groups/1363>

[Kw01] T. Kwon, "Authentication and Key Agreement via Memorable Password", NDSS 2001 Symposium Conference Proceedings, February 7-9, 2001.

[Kw02] T. Kwon, "Authentication via Memorable Passwords — Revised Submission to IEEE P1363.2", Submission to the IEEE P1363 Working Group, received October 31, 2002. Available at <http://grouper.ieee.org/groups/1363/passwordPK/contributions>

[Kw03a] T. Kwon, "Summary of AMP (Authentication and key agreement via Memorable Passwords)", Contribution to the IEEE P1363 Working Group, received August 22, 2003. Available at <http://grouper.ieee.org/groups/1363>

[Kw03b] T. Kwon, "Addendum to Summary of AMP", Submission to the IEEE P1363 Working Group, received November 20, 2003. Available at <http://grouper.ieee.org/groups/1363/passwordPK/contributions>.

[Kw05] T. Kwon, "Revision of AMP in IEEE P1363.2 and ISO/IEC 11770-4", Contribution to IEEE P1363 Working Group, received June 8, 2005. Available at <http://grouper.ieee.org/groups/1363>

[WW04] Z. Wan & S. Wang, "Cryptanalysis of Two Password-Authenticated Key Exchange Protocols", H. Wang et al. (Eds.): ACISP 2004, LNCS 3108, pp. 164-175, Springer-Verlag, Berlin, Heidelberg, 2004.

## Appendix

This table shows how the Client's secret group element  $z_g$  was computed in various AMP proposals.  $C$  and  $S$  represent identifiers for the Client and Server, and  $o_{ID} = C \parallel S$ .

Date	Document	Client's $z_g = w_S^{(x/y)}$	
		$x$	$y$
2005-12-07	This proposal	$s + 1$	$(s \times \text{Hash}_1(w_C \parallel o_{ID})) + \text{Hash}_2(\pi)$
2005-06-08	[Kw05] AMP2	$s + 1$	$(s \times \text{Hash}_1(C, S, w_C)) + \text{Hash}_2(C, \pi)$
2005-06-08	[Kw05] AMP+	$s + \text{Hash}(w_C \parallel w_S \parallel o_{ID})$	$(s \times \text{Hash}_{w_1}(o_{ID} \parallel w_C)) + \text{Hash}(C \parallel \pi)$
2005-03-08	P1363.2 D20	$s + 1$	$s + \text{Hash}(\pi)$
2003-08-22	[Kw03a] AMP2	$s + 1$	$(s \times \text{Hash}_1(o_{ID} \parallel w_C)) + \text{Hash}_2(o_{ID} \parallel \pi)$
2003-08-12	P1363.2 D11	$s + 1$	$s + \text{Hash}(\pi)$
2003-06-14	[Kw03] L-AMP	$s + 1$	$s + \text{Redp}(\pi)$
2002-12-20	P1363.2 D7	$s + 1$	$s + \text{Hash}(\pi)$
2002-10-31	[Kw02] AMP	$s + 1$	$s + \text{Redp}(\pi)$
2002-08-10	P1363.2 draft	$s + \text{Hash}(w_C \parallel w_S \parallel o_{ID})$	$s + \text{Redp}(\pi)$
2002-03-07	P1363.2 draft	$s + \text{Hash}(w_C \parallel w_S \parallel o_{ID})$	$s + \text{Redp}(\pi)$
2001-05-14	P1363.2 draft	<i>No AMP specified</i>	
2001-02-07	[Kw01] AMP	$s + \text{Hash}(w_C \parallel w_S \parallel o_{ID})$	$s + \text{Hash}(\pi)$
2001-02-07	[Kw01] AMP <sup>n</sup>	$s$	$s + \text{Hash}(\pi)$
2001-02-07	[Kw01] AMP <sup>+</sup>	$s + \text{Hash}(w_C \parallel w_S \parallel o_{ID})$	$(s \times \text{Hash}_{w_1}(o_{ID} \parallel w_C)) + \text{Hash}(\pi)$
2000-06-22	[Kw00] AMP	$s + \text{Hash}(w_C \parallel w_S \parallel o_{ID})$	$s + \text{Hash}(\pi)$
2000-06-22	[Kw00] AMP <sup>n</sup>	$s$	$s + \text{Hash}(\pi)$
2000-06-22	[Kw00] AMP <sup>+</sup>	$s + \text{Hash}(w_C \parallel w_S \parallel o_{ID})$	$(s \times \text{Hash}_{w_1}(o_{ID} \parallel w_C)) + \text{Hash}(\pi)$