APKAS-AMP-Kwon-2005-06-18

This is a version of APKAS-AMP, proposed by T. Kwon to align APKAS-AMP more closely with AMP+ [Kw01]. The purpose of this change is to prevent the attack on D20 APKAS-AMP, described in [Kw05], in which an enemy can masquerade as the Client using v_π,. This document highlights the difference between this proposal and P1363.2 draft D20. The P1363 WG plans to discuss this and any other proposals for resolving this attack in the July 20, 2005 teleconference.

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 [Kw01~~00~~] and [Kw03b~~02~~]. 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 $Hash_{w1}$ (see Note 1), which should be one of the hash functions in 14.1 or MGF1 (see 14.2.1).

— An octet string o_{ID} that may provide additional input to $Hash_{w1}$.

The Client and Server shall use the same $Hash_{w1}$ and o_{ID} parameters with PEPKGP-AMP-SERVER and SVDP-AMP-CLIENT.

Input:

— The password verification data v_π, a password-limited public key

— The Server's private key s

— The Client's public key w_C

— The {DL,EC} domain parameters (including r) associated with the keys v_π, s, and w_C

Assumptions: Private key s, password-limited public key v_π, 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 a group element of order r

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 = GE2OSP-X(w_C)
2. Compute $o_1 = Hash_{w1}(o_C \,||\, o_{ID})$
3. Compute i_1 = OS2IP(o_1)

4~~1~~. Compute $w_S = ((w_C \;\hat{}\; i_1) * v_\pi) \;\hat{}\; s$

5~~2~~. 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 $Hash_{w1}$ in AMP.*

8.2.9 PKGP-DH

{DL,EC}PKGP-DH is {Discrete Logarithm, Elliptic Curve} Public Key Generation Primitive, Diffie-Hellman version. PKGP-DH is based on the work of [DH76] and IEEE Std 1363-2000. This primitive derives a {DL,EC} public key from the party's {DL,EC} private key using {DL,EC} domain parameters. This primitive is used by the schemes {DL,EC}BPKAS-PAK-SERVER, {DL,EC}APKAS-AMP-CLIENT and ECAPKAS-SRP5-CLIENT.

Input:

— The party's own private key integer s

— The {DL,EC} domain parameters (including g and r) associated with the key s

Assumptions: Private key s and associated domain parameters are valid.

Output: The computed public key value w, which is a group element of order r

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

1. Compute public key group element $w = g$^s
2. Output w as the public key value

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

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 [Kw01~~00~~] and [Kw03b]. 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 $Hash_{PVD}$ (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 π

— The {DL,EC} domain parameters (including g and r) associated with π

Assumptions: Domain parameters are valid.

Output: The password-limited private key u_π and associated password verification data consisting of password-limited public key v_π

Operation: The password-limited private key u_π and verification data v_π shall be computed by the following or an equivalent sequence of steps:

1. Compute $o_\pi = Hash_{PVD}(\pi)$
2. Compute $u_\pi = \mathrm{OS2IP}(o_\pi) \bmod r$
3. Compute $v_\pi = g \wedge u_\pi$
4. Output u_π and verification data v_π

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~~nn~~ *Range of password-limited private keys* for discussion of appropriate choices for $Hash_{PVD}$.

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 [Kw01~~00~~] and [Kw03b~~02~~]. 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_π, 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 $Hash_{w1}$ (see Note 1), which should be one of the hash functions in 14.1 or MGF1 (see 14.2.1).

— A hash function $Hash_{w2}$ (see Note 2), which should be one of the hash functions in 14.1 or MGF1 (see 14.2.1).

— An octet string o_{ID} that may provide additional input to $Hash_{w1}$ and $Hash_{w2}$.

The Client and Server shall use the same $Hash_{w1}$, $Hash_{w2}$, and o_{ID} parameters with PEPKGP-AMP-SERVER, SVDP-AMP-CLIENT, and SVDP-AMP-SERVER.

Input:

— The Client's private key s

— The Client's password-limited private key u_π

— 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, w_C, w_S and u_π

Assumptions: Private keys s and u_π and associated domain parameters are valid. w_C and w_S are ~~is~~ 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_S = GE2OSP-X(w_S)
3. Compute $o_1 = Hash_{w1}(o_C \| o_{ID})$
4. Compute $o_2 = Hash_{w2}(o_C \| o_S \| o_{ID})$
5. Compute i_1 = OS2IP(o_1)
6. Compute i_2 = OS2IP(o_2)
7~~1~~. Compute $i_{3\cancel{2}} = ((s + i_2\cancel{1}) / ((s \times i_1) + u_\pi)) \bmod r$
8~~2~~. Compute $z_g = w_S$ ^ $i_{3\cancel{2}}$
9~~3~~. Compute z = GE2SVFEP(z_g)
10~~4~~. Output z

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* $Hash_{w1}$ *in AMP*.

2—See D.5.5.1.10 *Criteria for selecting* $Hash_{w2}$ *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 [Kw01~~00~~] and [Kw03b~~02~~].

This primitive is parameterized by the following choices:

— A hash function $Hash_{w2}$ (see Note 1), which should be one of the hash functions in 14.1 or MGF1 (see 14.2.1).

— An octet string o_{ID} that may provide additional input to $Hash_{w2}$.

The Client and Server shall use the same $Hash_{w2}$ and o_{ID} parameters with SVDP-AMP-CLIENT and SVDP-AMP-SERVER.

Input:

— The Server's own private key s

— The Client's public key w_C

— The Server's password-entangled public key w_S

— The {DL,EC} domain parameters (including q and g) associated with the keys s, w_C and $w_{S\cancel{C}}$

Assumptions: Private key s and associated DL domain parameters are valid. w_C and w_S are ~~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 o_C = GE2OSP-X(w_C)
2. Compute o_S = GE2OSP-X(w_S)
3. Compute $o_2 = Hash_{w2}(o_C \| o_S \| o_{ID})$

4. Compute i_2 = OS2IP(o_2)
5~~1~~. Compute $z_g = (w_C * (g\hat{}i_2)) \hat{}\ s$
~~2. If the order of z_g is unacceptably small, output “invalid” and stop. (See Note)~~
6~~3~~. Compute z = GE2SVFEP(z_g)
7~~4~~. Output z

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 $Hash_{w2}$ in AMP*.

~~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”.~~

9.5 APKAS-AMP

{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 [Kw01~~00~~] and [Kw03b~~02~~]. The Server uses a password-limited public key v_π as password verification data that was derived using PVDGP-AMP with the input value π, 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 π

For SERVER only:

— A password-limited public key v_π 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 *$Hash_{w1}$*, *$Hash_{w2}$* and o_{ID} parameters with PEPKGP-AMP-SERVER, SVDP-AMP-CLIENT, and SVDP-AMP-SERVER.

— A set of valid {DL,EC} domain parameters (including q and r) associated with π and v_π.~~ (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, ...} 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, ... K_t, 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.nn *Private keys*)
2. Compute public key w_C = {DL,EC}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 w_S from the Server
   4.1 If w_S is not in the parent group, output “invalid” and stop.
   4.2 If the order of w_S is unacceptably small, output “invalid” and stop. (See Note 2)
5. Compute password-limited private key u_π using the steps described in {DL,EC}PVDGP-AMP(π)
6. Compute field element z using {DL,EC}SVDP-AMP-CLIENT(s, u_π, w_C, w_S)
7. Compute octet string Z = 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 $K_i = Kdf(Z, P_i)$.
9. Output derived keys K_1, K_2, ... K_t

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.nn *Private keys*)

2. Receive public key 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 3)~~
3. Generate password-entangled public key w_S using {DL,EC}PEPKGP-AMP-SERVER(v_π, s, 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, w_S)
   ~~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, ... K_t

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~~xxx ===~~ for discussion of the limitations on the use of unilateral commitment schemes in application protocols.

~~2—See D.5.5.1 for discussion of the security considerations for AMP, including the reasons for these steps of checking for acceptable values, the meaning of “unacceptably small” order, and a potential timing attack related to Step 3.1 of the Key agreement for Server operation.~~

~~3—The Server’s need to validate w_C during key agreement depends on the domain parameters. It is optional for the Server to abort when the received w_C is determined to be an invalid public key. This step is is both unnecessary and may require significant added computation when using certain recommended settings, such as DL $GF(p)$ with a “safe prime” p. However, this step or some 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.~~

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 the referenced [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 recommended that cofactor 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.5.4.6.

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_{π}. See D.5.4.6.2 for the meaning of "unacceptably small" and further discussion of small subgroup confinement.

Editor's Note—Sections D.5.5.1.3 through D.5.5.1.7 were removed in accordance with a proposal by Kwon to align with AMP+ [Kw01]. The purpose of the proposal is to prevent the attack on D20 APKAS-AMP that is described in Kwon's June 8 2005 contribution, in which an enemy can masquerade as the Client using v_{π},.

~~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 PEPKGP-AMP-SERVER that could result in the Server computing $z = 0$.~~

~~Furthermore, the order of ($w_C * v_{\pi}$) must not be a composite with multiple factors significantly smaller than r, to avoid problem discussed in D.5.4.6. To avoid computational expense 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*.~~

~~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 π. 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 subtitute 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 (π, π_2) in a single run. He could compute $w_C = g$^(– *Hash*(π_2)), test $w_S = 1$ to detect whether the password is π_2, and if not, test whether the server agrees on the value for z to detect whether the password is π. Or, in a similar attack, an enemy could choose a small order element e and compute $w_C = e * g$^(– *Hash*(π_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)^s), and it must be computationally independent from z, so as to ensure that an attacking Client who has stolen v_π cannot derive z from w_S.~~

D.5.5.1.8 Need for $Hash_{w1}$ and $Hash_{w2}$ in AMP

Editor's Note—Summarize the problem and solution described in [Kw05].

The use of the $Hash_{w1}$ and $Hash_{w2}$ functions was introduced in the AMP+ protocol of [Kw01], and distinguishes it from the AMP protocol of [Kw00]. [Kw05] describes how an earlier draft version of APKAS-AMP, that was based in part on the AMP protocol of [Kw00], was subject to an attack by one who obtains the password verification data and can subsequently masquerade as the Client without performing a dictionary attack, thus removing the augmented benefit.

Editor's Note—Discuss whether use of $Hash_{w1}$ and $Hash_{w2}$ prevents any other potential attacks, or eliminates the need for other constraintsm, or amend other D.5.5.1 subsections as appropriate.

D.5.5.1.9 Criteria for selecting $Hash_{w1}$ in AMP

Editor's Note—Add this new section. Discuss size of output. Discuss incorporation of Client and Server identifiers in the o_{ID} input parameter for $Hash_{w1}$ (see [Kw01] and [Kw03b]).

Editor's Note—Consider also allowing $Hash_{w1}$ = MGF1, for more variety in size of output.

Editor's Note—Consider whether $Hash_{w1}$ should really be a KDF instead of a hash, since a KDF already has the extra input parameter.

D.5.5.1.9 Criteria for selecting $Hash_{w2}$ in AMP

Editor's Note—Add this new section. Discuss size of output. Discuss incorporation of Client and Server identifiers in the o_{ID} input parameter for $Hash_{w2}$ (see [Kw01] and [Kw03b]).

Editor's Note—Consider also allowing $Hash_{w2}$ = MGF1, for more variety in size of output.

Editor's Note—Consider whether $Hash_{w2}$ should really be a KDF instead of a hash, since a KDF already has the extra input parameter.

9.5 APKAS-AMP

	Client		***Server***
Enrollment:	π = salt $\Vert$ pwd $\Vert$ IDs ...		
PVDGP-AMP	$u_\pi = Hash_{PVD}(\pi)$ $v_\pi = g$ ^ u_π	$v_\pi \rightarrow$	v_π
Key Agreement:	$s \in_R [1, r-1]$		$s \in_R [1, r-1]$
PKGP-DH	$w_C = g$ ^ s	$w_C \rightarrow$	~~Abort if $w_C \notin$ parent group~~ ~~(opt) Abort if w_C invalid~~
PEPKGP-AMP-SERVER	Abort if $w_S \notin$ parent group Abort if $o(w_S)$ too small	$\leftarrow w_S$	$i_1 = Hash_{w1}(w_C \Vert o_{ID})$ $w_S = ((w_C$^$i_1)*v_\pi)$^s ~~If $o(w_S)$ too small,~~ ~~w_S = random public key~~

PVDGP-AMP	$u_\pi = Hash_{PVD}(\pi)$	
SVDP-AMP-CLIENT	$i_1 = Hash_{w1}(w_C \Vert o_{ID})$ $i_2 = Hash_{w2}(w_C \Vert w_S \Vert o_{ID})$ $z = w_S\,\hat{}\,((s+i_2\cancel{1})/(s \times i_1+u_\pi))$	
SVDP-AMP-SERVER		$i_2 = Hash_{w2\cancel{3}}(w_C \Vert w_S \Vert o_{ID})$ $z = (w_C * g\,\hat{}\,i_2)\,\hat{}\,s$ ~~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.

Annex G (Informative) Bibliography

[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 Password — 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/passwdPK/contributions.

[Kw03b] T. Kwon, "Addendum to Summary of AMP~~ — Revised Submission to IEEE P1363.2~~", Submission to the IEEE P1363 Working Group, received November 20, 2003. Available at http://grouper.ieee.org/groups/1363/passwdPK/contributions.

[Kw05] T. Kwon, contribution to P1363 Working Group, received June 8, 2005.