Internet Engineering Task Force (IETF)
Request for Comments: 6628
Category: Experimental
ISSN: 2070-1721

S. Shin
K. Kobara
AIST

June 2012

Efficient Augmented Password-Only Authentication and Key Exchange for IKEv2

Abstract

This document describes an efficient augmented password-only authentication and key exchange (AugPAKE) protocol where a user remembers a low-entropy password and its verifier is registered in the intended server. In general, the user password is chosen from a small set of dictionary words that allows an attacker to perform exhaustive searches (i.e., off-line dictionary attacks). The AugPAKE protocol described here is secure against passive attacks, active attacks, and off-line dictionary attacks (on the obtained messages with passive/active attacks), and also provides resistance to server compromise (in the context of augmented PAKE security). In addition, this document describes how the AugPAKE protocol is integrated into the Internet Key Exchange Protocol version 2 (IKEv2).

Status of This Memo

This document is not an Internet Standards Track specification; it is published for examination, experimental implementation, and evaluation.

This document defines an Experimental Protocol for the Internet community. This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Not all documents approved by the IESG are a candidate for any level of Internet Standard; see Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc6628.

Table of Contents

1. Introduction

In the real world, many applications, such as Web mail and Internet banking/shopping/trading, require secure channels between participating parties. Such secure channels can be established by using an authentication and key exchange (AKE) protocol, which allows the involved parties to authenticate each other and to generate a temporary session key. The temporary session key is used to protect the subsequent communications between the parties.

Until now, password-only AKE (called PAKE) protocols have attracted much attention because password-only authentication is very convenient to the users. However, it is not trivial to design a secure PAKE protocol due to the existence of off-line dictionary attacks on passwords. These attacks are possible since passwords are chosen from a relatively-small dictionary that allows for an attacker to perform the exhaustive searches. This problem was brought forth by Bellovin and Merritt [BM92], and many subsequent works have been conducted in the literature (see some examples in [IEEEP1363.2]). A PAKE protocol is said to be secure if the best attack an active attacker can take is restricted to the on-line dictionary attacks, which allows a guessed password to be checked only by interacting with the honest party.

An augmented PAKE protocol (e.g., [BM93], [RFC2945], [ISO]) provides extra protection for server compromise in the sense that an attacker, who obtains a password verifier from a server, cannot impersonate the corresponding user without performing off-line dictionary attacks on the password verifier. This additional security is known as "resistance to server compromise". The AugPAKE protocol described in this document is an augmented PAKE, which also achieves measurable efficiency over some previous works (i.e., SRP [RFC2945] and AMP [ISO]). We believe the following (see [SKI10] for the formal security proof): 1) The AugPAKE protocol is secure against passive attacks, active attacks, and off-line dictionary attacks (on the obtained messages with passive/active attacks), and 2) It provides resistance to server compromise. At the same time, the AugPAKE protocol has similar computational efficiency to the plain Diffie-Hellman key exchange [DH76] that does not provide authentication by itself. Specifically, the user and the server need to compute 2 and 2.17 modular exponentiations, respectively, in the AugPAKE protocol. After excluding pre-computable costs, the user and the server are required to compute only 1 and 1.17 modular exponentiations, respectively. Compared with SRP [RFC2945] and AMP [ISO], the AugPAKE protocol is more efficient 1) than SRP in terms of the user's computational costs and 2) than AMP in terms of the server's computational costs.

This document also describes how the AugPAKE protocol is integrated into IKEv2 [RFC5996].

1.1. Keywords

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2. AugPAKE Specification

2.1. Underlying Group

The AugPAKE protocol can be implemented over the following group.

- Let p and q be sufficiently large primes such that q is a divisor of ((p - 1) / 2), and every factor of ((p - 1) / 2) are also primes comparable to q in size. This p is called a "secure" prime. By G, we denote a multiplicative subgroup of prime order q over the field GF(p), the integers modulo p. Let g be a generator for the subgroup G so that all the subgroup elements are generated by g. The group operation is denoted multiplicatively (in modulo p).

By using a secure prime p, the AugPAKE protocol has computational efficiency gains. Specifically, it does not require the order check of elements received from the counterpart party. Note that the groups defined in Discrete Logarithm Cryptography [SP800-56A] and RFC 5114 [RFC5114] are not necessarily the above secure prime groups.

Alternatively, one can implement the AugPAKE protocol over the following groups.

- Let p and q be sufficiently large primes such that p = (2 * q) + 1. This p is called a "safe" prime. By G, we denote a multiplicative subgroup of prime order q over the field GF(p), the integers modulo p. Let g be any element of G other than 1. For example, g = h^2 mod p where h is a primitive element. The group operation is denoted multiplicatively (in modulo p).

- Let p and q be sufficiently large primes such that q is a divisor of ((p - 1) / 2). By G, we denote a multiplicative subgroup of prime order q over the field GF(p), the integers modulo p. Let g be a generator for the subgroup G so that all the subgroup elements are generated by g. The group operation is denoted multiplicatively (in modulo p). If p is not a "secure" prime, the AugPAKE protocol MUST perform the order check of received elements.

2.2. Notation

The AugPAKE protocol is a two-party protocol where a user and a server authenticate each other and generate a session key. The following notation is used in this document:

U
: The user's identity (e.g., as defined in [RFC4282]). It is a string in {0,1}^* where {0,1}^* indicates a set of finite binary strings.

S
: The server's identity (e.g., as defined in [RFC4282]). It is a string in {0,1}^*.

b = H(a)
: A binary string a is given as input to a secure one-way hash function H (e.g., SHA-2 family [FIPS180-3]), which produces a fixed-length output b. The hash function H maps {0,1}^* to {0,1}^k, where {0,1}^k indicates a set of binary strings of length k and k is a security parameter.

b = H'(a)
: A binary string a is given as input to a secure one-way hash function H', which maps the input a in {0,1}^* to the output b in Z_q^*, where Z_q^* is a set of positive integers modulo prime q.

a | b
: It denotes a concatenation of binary strings a and b in {0,1}^*.

0x
: A hexadecimal value is shown preceded by "0x".

X * Y mod p
: It indicates a multiplication of X and Y modulo prime p.

X = g^x mod p
: The g^x indicates a multiplication computation of g by x times. The resultant value modulo prime p is assigned to X. The discrete logarithm problem says that it is computationally hard to compute the discrete logarithm x from X, g, and p.

w
: The password remembered by the user. This password may be used as an effective password (instead of itself) in the form of H'(0x00 | U | S | w).

W
: The password verifier registered in the server. This password verifier is computed as follows: W = g^w mod p where the user's password w is used itself, or W = g^w' mod p where the effective password w' = H'(0x00 | U | S | w) is used.

bn2bin(X)
: It indicates a conversion of a multiple precision integer X to the corresponding binary string. If X is an element over GF(p), its binary representation MUST have the same bit length as the binary representation of prime p.

U -> S: msg
: It indicates a message transmission that the user U sends a message msg to the server S.

U:
: It indicates a local computation of user U (without any outgoing messages).

2.2.1. Password Processing

The input password MUST be processed according to the rules of the [RFC4013] profile of [RFC3454]. The password SHALL be considered a "stored string" per [RFC3454], and unassigned code points are therefore prohibited. The output SHALL be the binary representation of the processed UTF-8 character string. Prohibited output and unassigned code points encountered in SASLprep pre-processing SHALL cause a failure of pre-processing, and the output SHALL NOT be used with the AugPAKE protocol.

The following table shows examples of how various character data is transformed by the rules of the [RFC4013] profile.

#	Input	Output	Comments
1	I<U+00AD>X	IX	SOFT HYPHEN mapped to nothing
2	user	user	no transformation
3	USER	USER	case preserved, will not match #2
4	<U+00AA>	a	output is NFKC, input in ISO 8859-1
5	<U+2168>	IX	output is NFKC, will match #1
6	<U+0007>		Error - prohibited character
7	<U+0627><U+0031>		Error - bidirectional check

2.3. Protocol

The AugPAKE protocol consists of two phases: initialization and actual protocol execution. The initialization phase SHOULD be finished in a secure manner between the user and the server, and it is performed all at once. Whenever the user and the server need to establish a secure channel, they can run the actual protocol execution through an open network (i.e., the Internet) in which an active attacker exists.

2.3.1. Initialization

U -> S: (U, W)
The user U computes W = g^w' mod p, where w' is the effective password, and transmits W to the server S. The W is registered in the server as the password verifier of user U. Of course, user U just remembers password w only.

If resistance to server compromise is not necessary and a node needs to act as both initiator and responder, e.g., as a gateway, then the node can store w' instead of W even when it acts as server S. In either case, server S SHOULD NOT store any plaintext passwords.

As noted above, this phase SHOULD be performed securely and all at once.

2.3.2. Actual Protocol Execution

The actual protocol execution of the AugPAKE protocol allows the user and the server to share an authenticated session key through an open network (see Figure 1).

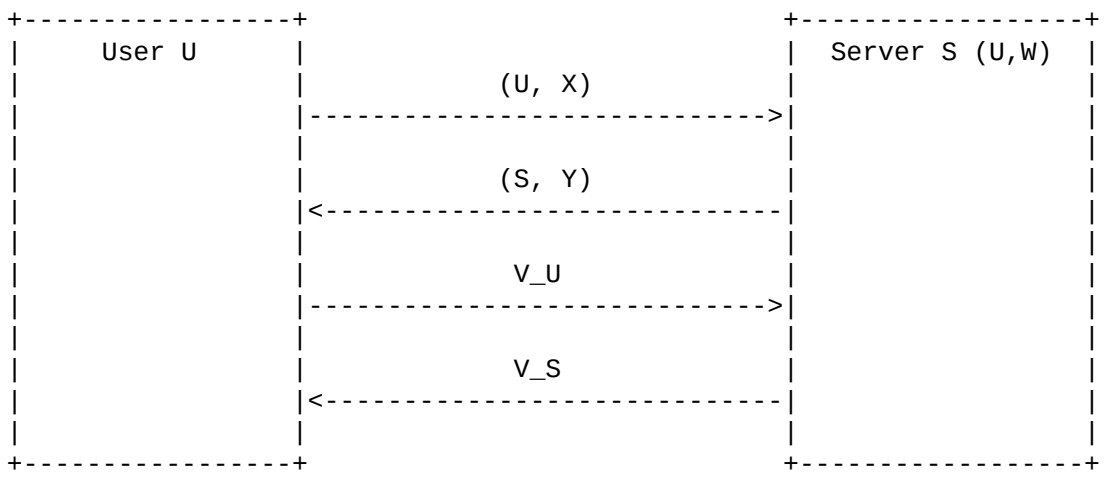


Figure 1: Actual Protocol Execution

U -> S: (U, X)
The user U chooses a random element x from Z_q^* and computes its Diffie-Hellman public value X = g^x mod p. The user sends the first message (U, X) to the server S.

S -> U: (S, Y)
If the received X from user U is 0, 1, or -1 (mod p), server S MUST terminate the protocol execution. Otherwise, the server chooses a random element y from Z_q^* and computes Y = (X * (W^r))^y mod p where r = H'(0x01 | U | S | bn2bin(X)). Note that X^y * g^(w * r * y) mod p can be computed from y and (w * r * y) efficiently using Shamir's trick [MOV97]. Then, server S sends the second message (S, Y) to the user U.

U -> S: V_U
If the received Y from server S is 0, 1, or -1 (mod p), user U MUST terminate the protocol execution. Otherwise, the user computes K = Y^z mod p where z = 1 / (x + (w * r)) mod q and r = H'(0x01 | U | S | bn2bin(X)). Also, user U generates an authenticator V_U = H(0x02 | U | S | bn2bin(X) | bn2bin(Y) | bn2bin(K)). Then, the user sends the third message V_U to the server S.

S -> U: V_S
If the received V_U from user U is not equal to H(0x02 | U | S | bn2bin(X) | bn2bin(Y) | bn2bin(K)) where K = g^y mod p, server S MUST terminate the protocol execution. Otherwise, the server generates an authenticator V_S = H(0x03 | U | S | bn2bin(X) | bn2bin(Y) | bn2bin(K)) and a session key SK = H(0x04 | U | S | bn2bin(X) | bn2bin(Y) | bn2bin(K)). Then, server S sends the fourth message V_S to the user U.

U:
If the received V_S from server S is not equal to H(0x03 | U | S | bn2bin(X) | bn2bin(Y) | bn2bin(K)), user U MUST terminate the protocol execution. Otherwise, the user generates a session key SK = H(0x04 | U | S | bn2bin(X) | bn2bin(Y) | bn2bin(K)).

In the actual protocol execution, the sequential order of message exchanges is very important to avoid any possible attacks. For example, if the server S sends the second message (S, Y) and the fourth message V_S together, any attacker can easily derive the correct password w with off-line dictionary attacks.

The session key SK, shared only if the user and the server authenticate each other successfully, MAY be generated by using a key derivation function (KDF) [SP800-108]. After generating SK, the user and the server MUST delete all the internal states (e.g., Diffie-Hellman exponents x and y) from memory.

For the formal proof [SKI10] of the AugPAKE protocol, we need to slightly change the computation of Y (in the above S -> U: (S, Y)) and K (in the above S -> U: V_S) as follows: Y = (X * (W^r))^y' and K = g^y' where y' = H'(0x05 | bn2bin(y)).

3. Security Considerations

This section shows why the AugPAKE protocol (i.e., the actual protocol execution) is secure against passive attacks, active attacks, and off-line dictionary attacks, and also provides resistance to server compromise.

3.1. General Assumptions

- An attacker is computationally bounded.

- Any hash functions used in the AugPAKE protocol are secure in terms of pre-image resistance (one-wayness), second pre-image resistance, and collision resistance.

3.2. Security against Passive Attacks

An augmented PAKE protocol is said to be secure against passive attacks in the sense that an attacker, who eavesdrops the exchanged messages, cannot compute an authenticated session key (shared between the honest parties in the protocol).

In the AugPAKE protocol, an attacker can get the messages (U, X), (S,Y), V_U, V_S by eavesdropping, and then wants to compute the session key SK. That is, the attacker's goal is to derive the correct K from the obtained messages X and Y, because the hash functions are secure and the only secret in the computation of SK is K = g^y mod p. Note that

X = g^x mod p and

Y = (X * (W^r))^y = X^y * W^(r * y) = X^y * (g^y)^t = X^y * K^t

hold where t = w' * r mod q. Though t is determined from possible password candidates and X, the only way for the attacker to extract K from X and Y is to compute X^y. However, the probability for the attacker to compute X^y is negligible in the security parameter for the underlying groups since both x and y are random elements chosen from Z_q^*. Therefore, the AugPAKE protocol is secure against passive attacks.

3.3. Security against Active Attacks

An augmented PAKE protocol is said to be secure against active attacks in the sense that an attacker, who completely controls the exchanged messages, cannot compute an authenticated session key (shared with the honest party in the protocol) with the probability better than that of on-line dictionary attacks. In other words, the probability for an active attacker to compute the session key is restricted by the on-line dictionary attacks where it grows linearly to the number of interactions with the honest party.

In the AugPAKE protocol, the user (respectively, the server) computes the session key SK only if the received authenticator V_S (respectively, V_U) is valid. There are three cases to be considered in the active attacks.

3.3.1. Impersonation Attacks on User U

When an attacker impersonates the user U, the attacker can compute the same SK (to be shared with the server S) only if the authenticator V_U is valid. For a valid authenticator V_U, the attacker has to compute the correct K from X and Y because the hash

functions are secure. In this impersonation attack, the attacker of course knows the discrete logarithm x of X and guesses a password w'' from the password dictionary. So, the probability for the attacker to compute the correct K is bounded by the probability of w = w''. That is, this impersonation attack is restricted by the on-line dictionary attacks where the attacker can try a guessed password communicating with the honest server S. Therefore, the AugPAKE protocol is secure against impersonation attacks on user U.

3.3.2. Impersonation Attacks on Server S

When an attacker impersonates the server S, the attacker can compute the same SK (to be shared with the user U) only if the authenticator V_S is valid. For a valid authenticator V_S, the attacker has to compute the correct K from X and Y because the hash functions are secure. In this impersonation attack, the attacker chooses a random element y and guesses a password w'' from the password dictionary so that

Y = (X * (W'^r))^y = X^y * W'^(r * y) = X^y * (g^y)^t'

where t' = w'' * r mod q. The probability for the attacker to compute the correct K is bounded by the probability of w = w''. Also, the attacker knows whether the guessed password is equal to w or not by seeing the received authenticator V_U. However, when w is not equal to w'', the probability for the attacker to compute the correct K is negligible in the security parameter for the underlying groups since the attacker has to guess the discrete logarithm x (chosen by user U) as well. That is, this impersonation attack is restricted by the on-line dictionary attacks where the attacker can try a guessed password communicating with the honest user U. Therefore, the AugPAKE protocol is secure against impersonation attacks on server S.

3.3.3. Man-in-the-Middle Attacks

When an attacker performs the man-in-the-middle attack, the attacker can compute the same SK (to be shared with the user U or the server S) only if one of the authenticators V_U, V_S is valid. Note that if the attacker relays the exchanged messages honestly, it corresponds to the passive attacks. In order to generate a valid authenticator V_U or V_S, the attacker has to compute the correct K from X and Y because the hash functions are secure. So, the attacker is in the same situation as discussed above. Though the attacker can test two passwords (one with user U and the other with server S), it does not change the fact that this attack is restricted by the on-line dictionary attacks where the attacker can try a guessed password

communicating with the honest party. Therefore, the AugPAKE protocol is also secure against man-in-the-middle attacks.

3.4. Security against Off-line Dictionary Attacks

An augmented PAKE protocol is said to be secure against off-line dictionary attacks in the sense that an attacker, who completely controls the exchanged messages, cannot reduce the possible password candidates better than on-line dictionary attacks. Note that in the on-line dictionary attacks, an attacker can test one guessed password by running the protocol execution (i.e., communicating with the honest party).

As discussed in Section 3.2, an attacker in the passive attacks does not compute X^y (and the correct K = g^y mod p) from the obtained messages X, Y. This security analysis also indicates that, even if the attacker can guess a password, the K is derived independently from the guessed password. Next, we consider an active attacker whose main goal is to perform the off-line dictionary attacks in the AugPAKE protocol. As in Section 3.3, the attacker can 1) test one guessed password by impersonating the user U or the server S, or 2) test two guessed passwords by impersonating the server S (to the honest user U) and impersonating the user U (to the honest server S) in the man-in-the-middle attacks. Whenever the honest party receives an invalid authenticator, the party terminates the actual protocol execution without sending any message. In fact, this is important to prevent an attacker from testing more than one password in the active attacks. Since passive attacks and active attacks cannot remove the possible password candidates more efficiently than on-line dictionary attacks, the AugPAKE protocol is secure against off-line dictionary attacks.

3.5. Resistance to Server Compromise

We consider an attacker who has obtained a (user's) password verifier from a server. In the (augmented) PAKE protocols, there are two limitations [BJKMRSW00]: 1) the attacker can find out the correct password from the password verifier with the off-line dictionary attacks because the verifier has the same entropy as the password; and 2) if the attacker impersonates the server with the password verifier, this attack is always possible because the attacker has enough information to simulate the server. An augmented PAKE protocol is said to provide resistance to server compromise in the sense that the attacker cannot impersonate the user without performing off-line dictionary attacks on the password verifier.

In order to show resistance to server compromise in the AugPAKE protocol, we consider an attacker who has obtained the password

verifier W and then tries to impersonate the user U without off-line dictionary attacks on W. As a general attack, the attacker chooses two random elements c and d from Z_q^*, and computes

X = (g^c) * (W^d) mod p

and sends the first message (U, X) to the server S. In order to impersonate user U successfully, the attacker has to compute the correct K = g^y mod p where y is randomly chosen by server S. After receiving Y from the server, the attacker's goal is to find out a value e satisfying Y^e = K mod p. That is,

log_g (Y^e) = log_g K mod q

(c + (w' * d) + (w' * r)) * y * e = y mod q

(c + w' * (d + r)) * e = 1 mod q

where log_g K indicates the logarithm of K to the base g. Since there is no off-line dictionary attacks on W, the above solution is that e = 1 / c mod q and d = -r mod q. However, the latter is not possible since r is determined by X (i.e., r = H'(0x01 | U | S | bn2bin(X))) and H' is a secure hash function. Therefore, the AugPAKE protocol provides resistance to server compromise.

4. Implementation Consideration

As discussed in Section 3, the AugPAKE protocol is secure against passive attacks, active attacks, and off-line dictionary attacks, and provides resistance to server compromise. However, an attacker in the on-line dictionary attacks can check whether one password (guessed from the password dictionary) is correct or not by interacting with the honest party. Let N be the number of possible passwords within a dictionary. Certainly, the attacker's success probability grows with the probability of (I / N) where I is the number of interactions with the honest party. In order to provide a reasonable security margin, implementation SHOULD take a countermeasure to the on-line dictionary attacks. For example, it would take about 90 years to test 2^(25.5) passwords with a one minute lock-out for 3 failed password guesses (see Appendix A in [SP800-63]).

5. AugPAKE for IKEv2

5.1. Integration into IKEv2

IKE is a primary component of IPsec in order to provide mutual authentication and establish security associations between two peers.

The AugPAKE protocol, described in Section 2, can be easily integrated into IKEv2 [RFC5996] as a "weak" pre-shared key authentication method (see Figure 2). This integrated protocol preserves the IKEv2 structure and security guarantees (e.g., identity protection). Note that the AugPAKE protocol can be used in three scenarios for IKEv2: "Security Gateway to Security Gateway Tunnel", "Endpoint-to-Endpoint Transport", and "Endpoint to Security Gateway Tunnel".

```
 Initiator                                Responder
-----------                              -----------

IKE_SA_INIT:

 HDR, SAi1, KEi, Ni,
 N(SECURE_PASSWORD_METHODS)      -->
                                 <--  HDR, SAr1, KEr, Nr,
                                      N(SECURE_PASSWORD_METHODS)

IKE_AUTH:

 HDR, SK {IDi, GSPM(PVi), [IDr,]
          SAi2, TSi, TSr}        -->
                                 <--  HDR, SK {IDr, GSPM(PVr)}
 HDR, SK {AUTHi}                 -->
                                 <--  HDR, SK {AUTHr, SAr2, TSi, TSr}
```

Figure 2: AugPAKE into IKEv2

The changes from IKEv2 are summarized as follows:

- In addition to IKEv2, one round trip is added.
- The initiator (respectively, the responder) sends an N(SECURE_PASSWORD_METHODS) notification to indicate its willingness to use AugPAKE in the IKE_SA_INIT exchange.
- The added values GSPM(PVi) and GSPM(PVr) in the IKE_AUTH exchange correspond to X and Y of the AugPAKE protocol in Section 2, respectively.
- From K (represented as an octet string) derived in Section 2, the AUTH values in the IKE_AUTH exchange are computed as

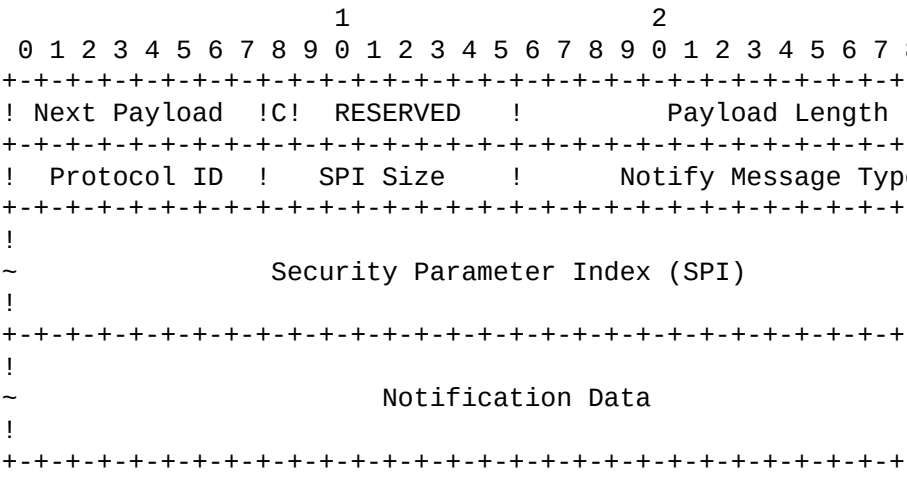
  AUTHi = prf(prf(K, "AugPAKE for IKEv2"),
  <InitiatorSignedOctets> | GSPM(PVi) | GSPM(PVr) | IDi | IDr)

```
AUTHr = prf( prf(K, "AugPAKE for IKEv2"),
<ResponderSignedOctets> | GSPM(PVr) | GSPM(PVi) | IDr | IDi)
```

5.2. Payload Formats

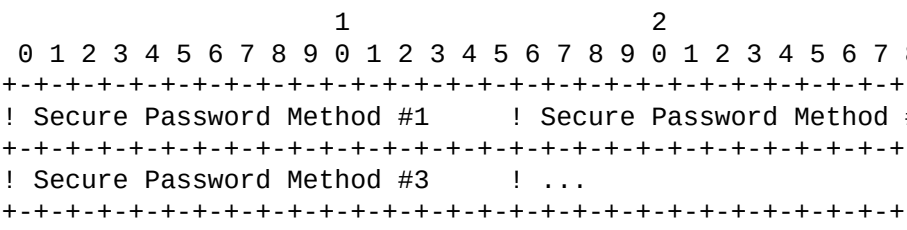
5.2.1. Notify Payload

The Notify Payload N(SECURE_PASSWORD_METHODS) [RFC6467], indicating a willingness to use AugPAKE in the IKE_SA_INIT exchange, is defined as follows:



As in [RFC5996], the Protocol ID and SPI Size SHALL be set to zero and, therefore, the SPI field SHALL be empty. The Notify Message Type will be 16424 [RFC6467].

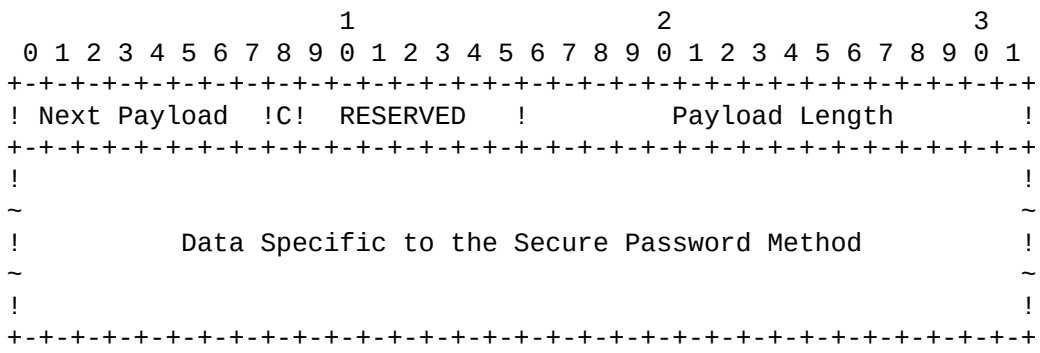
The Notification Data contains the list of the 16-bit secure password method numbers:



The response Notify Payload contains exactly one 16-bit secure password method number (i.e., for AugPAKE here) inside the Notification Data field.

5.2.2. Generic Secure Password Method Payload

The Generic Secure Password Method (GSPM) Payload, denoted GSPM(PV) in Section 5.1, is defined as follows:



The GSPM Payload Type will be 49 [RFC6467].

Since the GSPM(PV) value is a group element, the encoded octet string is actually used in the "Data Specific to the Secure Password Method" field.

6. IANA Considerations

IANA has assigned value 2 to the method name "AugPAKE" from the "IKEv2 Secure Password Methods" registry in [IKEV2-IANA].

7. References

7.1. Normative References

[FIPS180-3] Information Technology Laboratory, "Secure Hash Standard (SHS)", NIST FIPS Publication 180-3, October 2008, <http://csrc.nist.gov/publications/fips/fips180-3/fips180-3_final.pdf>.

[IKEV2-IANA] IANA, "Internet Key Exchange Version 2 (IKEv2) Parameters", <http://www.iana.org/assignments/ikev2-parameters>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3454] Hoffman, P. and M. Blanchet, "Preparation of Internationalized Strings ("stringprep")", RFC 3454, December 2002.

[RFC4013] Zeilenga, K., "SASLprep: Stringprep Profile for User Names and Passwords", RFC 4013, February 2005.

[RFC4282] Aboba, B., Beadles, M., Arkko, J., and P. Eronen, "The Network Access Identifier", RFC 4282, December 2005.

[RFC5996] Kaufman, C., Hoffman, P., Nir, Y., and P. Eronen, "Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 5996, September 2010.

[SP800-108] Chen, L., "Recommendation for Key Derivation Using Pseudorandom Functions (Revised)", NIST Special Publication 800-108, October 2009, <http://csrc.nist.gov/publications/nistpubs/800-108/sp800-108.pdf>.

7.2. Informative References

[BJKMRSW00] Bellare, M., Jablon, D., Krawczyk, H., MacKenzie, P., Rogaway, P., Swaminathan, R., and T. Wu, "Proposal for P1363 Study Group on Password-Based Authenticated-Key-Exchange Methods", IEEE P1363.2: Password-Based Public-Key Cryptography, Submissions to IEEE P1363.2 , February 2000, <http://grouper.ieee.org/groups/1363/passwdPK/contributions/p1363-pw.pdf>.

[BM92] Bellovin, S. and M. Merritt, "Encrypted Key Exchange: Password-based Protocols Secure against Dictionary Attacks", Proceedings of the IEEE Symposium on Security and Privacy, IEEE Computer Society, 1992.

[BM93] Bellovin, S. and M. Merritt, "Augmented Encrypted Key Exchange: A Password-based Protocol Secure against Dictionary Attacks and Password File Compromise", Proceedings of the 1st ACM Conference on Computer and Communication Security, ACM Press, 1993.

[DH76] Diffie, W. and M. Hellman, "New Directions in Cryptography", IEEE Transactions on Information Theory Volume IT-22, Number 6, 1976.

[H10] Harkins, D., "Password-Based Authentication in IKEv2: Selection Criteria and Considerations", Work in Progress, October 2010.

[IEEEP1363.2] IEEE P1363.2, "Password-Based Public-Key Cryptography", Submissions to IEEE P1363.2 , <http://grouper.ieee.org/groups/1363/passwdPK/submissions.html>.

[ISO] ISO/IEC JTC 1/SC 27 11770-4, "Information technology -- Security techniques -- Key management -- Part 4: Mechanisms based on weak secrets", April 2006, <http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=39723>.

[MOV97] Menezes, A., Oorschot, P., and S. Vanstone, "Simultaneous Multiple Exponentiation", in Handbook of Applied Cryptography, CRC Press, 1997.

[RFC2945] Wu, T., "The SRP Authentication and Key Exchange System", RFC 2945, September 2000.

[RFC5114] Lepinski, M. and S. Kent, "Additional Diffie-Hellman Groups for Use with IETF Standards", RFC 5114, January 2008.

[RFC6467] Kivinen, T., "Secure Password Framework for Internet Key Exchange Version 2 (IKEv2)", RFC 6467, December 2011.

[SKI10] Shin, S., Kobara, K., and H. Imai, "Security Proof of AugPAKE", Cryptology ePrint Archive: Report 2010/334, June 2010, <http://eprint.iacr.org/2010/334>.

[SP800-56A] Barker, E., Johnson, D., and M. Smid, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography (Revised)", NIST Special Publication 800-56A, March 2007, <http://csrc.nist.gov/publications/nistpubs/800-56A/SP800-56A_Revision1_Mar08-2007.pdf>.

[SP800-63] Burr, W., Dodson, D., and W. Polk, "Electronic Authentication Guideline", NIST Special Publication 800-63 Version 1.0.2, April 2006, <http://csrc.nist.gov/publications/nistpubs/800-63/SP800-63V1_0_2.pdf>.

Appendix A. Evaluation by PAKE Selection Criteria

Below is a self-evaluation of the AugPAKE protocol following PAKE selection criteria [H10].

SEC1: AugPAKE is zero knowledge (password) proof. It is secure against passive/active/off-line dictionary attacks. It is also resistant to server-compromise impersonation attacks.

SEC2: AugPAKE provides Perfect Forward Secrecy (PFS) and is secure against Denning-Sacco attack.

SEC3: IKEv2 identity protection is preserved.

SEC4: Any cryptographically secure Diffie-Hellman groups can be used.

SEC5: The formal security proof of AugPAKE can be found at [SKI10].

SEC6: AugPAKE can be easily used with strong credentials.

SEC7: In the case of server compromise, an attacker has to perform off-line dictionary attacks while computing modular exponentiation with a password candidate.

SEC8: AugPAKE is secure regardless of the transform negotiated by IKEv2.

IPR1: AugPAKE was publicly disclosed on Oct. 2008.

IPR2: AIST applied for a patent in Japan on July 10, 2008. AIST would provide royal-free license of AugPAKE.

IPR3: IPR disclosure (see https://datatracker.ietf.org/ipr/1284/)

MISC1: AugPAKE adds one round trip to IKEv2.

MISC2: The initiator needs to compute only 2 modular exponentiation computations while the responder needs to compute 2.17 modular exponentiation computations. AugPAKE needs to exchange 2 group elements and 2 hash values. This is almost the same computation/communication costs as the plain Diffie-Hellman (DH) key exchange. If we use a large (e.g., 2048/3072-bits) parent group, the hash size would be relatively small.

MISC3: AugPAKE has the same performance for any type of secret.

MISC4: Internationalization of character-based passwords can be supported.

MISC5: AugPAKE can be implemented over any ECP (Elliptic Curve Group over GF[P]), EC2N (Elliptic Curve Group over GF[2^N]), and MODP (Modular Exponentiation Group) groups.

MISC6: AugPAKE has request/response nature of IKEv2.

MISC7: No additional negotiation is needed.

MISC8: No Trusted Third Party (TTP) and clock synchronization

MISC9: No additional primitive (e.g., Full Domain Hashing (FDH) and/or ideal cipher) is needed.

MISC10: As above, AugPAKE can be implemented over any ECP/EC2N groups.

MISC11: Easy implementation. We already implemented AugPAKE and have been testing in AIST.

Authors' Addresses

SeongHan Shin
AIST
Central 2, 1-1-1, Umezono
Tsukuba, Ibaraki 305-8568
JP

Phone: +81 29-861-2670
EMail: seonghan.shin@aist.go.jp

Kazukuni Kobara
AIST

EMail: kobara_conf@m.aist.go.jp