Password-Authenticated Key (PAK) Diffie-Hellman Exchange

Abstract

   This document proposes to add mutual authentication, based on a
   human-memorizable password, to the basic, unauthenticated Diffie-
   Hellman key exchange. The proposed algorithm is called the Password-
   Authenticated Key (PAK) exchange. PAK allows two parties to
   authenticate themselves while performing the Diffie-Hellman exchange.

   The protocol is secure against all passive and active attacks. In
   particular, it does not allow either type of attacker to obtain any
   information that would enable an offline dictionary attack on the
   password. PAK provides Forward Secrecy.

Status of This Memo

   This document is not an Internet Standards Track specification; it is
   published for informational purposes.

   This is a contribution to the RFC Series, independently of any other
   RFC stream. The RFC Editor has chosen to publish this document at
   its discretion and makes no statement about its value for
   implementation or deployment. Documents approved for publication by
   the RFC Editor are not 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/rfc5683.

Copyright Notice

Table of Contents

1. Introduction

    PAK has the following advantages:

    - It provides a secure, authenticated key-exchange protocol.
    - It is secure against offline dictionary attacks when passwords are
       used.
    - It ensures Forward Secrecy.
    - It has been proven to be as secure as the Diffie-Hellman solution.

    The PAK protocol ([BMP00], [MP05], [X.1035]) has been proven to be as
    secure as the Diffie-Hellman ([RFC2631], [DH76]) in the random oracle
    model [BR93]. That is, PAK retains its security when used with low-
    entropy passwords. Therefore, it can be seamlessly integrated into
    existing applications, requiring secure authentication based on such
    low-entropy shared secrets.

2. Conventions

    - A is an identity of Alice.

    - B is an identity of Bob.

    - Ra is a secret random exponent selected by A.

    - Rb is a secret random exponent selected by B.

    - Xab denotes a value (X presumably computed by A) as derived by B.

    - Yba denotes a value (Y presumably computed by B) as derived by A.

    - A mod b denotes the least non-negative remainder when a is divided
       by b.

    - Hi(u) denotes an agreed-on function (e.g., based on SHA-1,
       SHA-256, etc.) computed over a string u; the various H() act as
       independent random functions. H1(u) and H2(u) are the key
       derivation functions. H3(u), H4(u), and H5(u) are the hash
       functions.

    - s|t denotes concatenation of the strings s and t.

    - ^ denotes exponentiation.

    - Multiplication, division, and exponentiation are performed over
       (Zp)*; in other words:

1) a*b always means a*b (mod p).

2) a/b always means a * x (mod p), where x is the multiplicative
   inverse of b modulo p.

3) a^b means a^b (mod p).

3. Password-Authenticated Key Exchange

Diffie-Hellman key agreement requires that both the sender and
recipient of a message create their own secret, random numbers and
exchange the exponentiation of their respective numbers.

PAK has two parties, Alice (A) and Bob (B), sharing a secret password
PW that satisfies the following conditions:

    H1(A|B|PW) != 0
    H2(A|B|PW) != 0

The global Diffie-Hellman publicly known constants, a prime p and a
generator g, are carefully selected so that:

1. A safe prime p is large enough to make the computation of
    discrete logarithms infeasible, and

2. Powers of g modulo p cover the entire range of p-1 integers from
    1 to p-1. (References demonstrate working examples of
    selections).

Initially, Alice (A) selects a secret, random exponent Ra and
computes g^Ra; Bob (B) selects a secret, random exponent Rb and
computes g^Rb. For efficiency purposes, short exponents could be
used for Ra and Rb, provided they have a certain minimum size. Then:

A --> B: {A, X = H1(A|B|PW)*(g^Ra)}
        (The above precondition on PW ensures that X != 0)

    Bob
      receives Q (presumably Q = X), verifies that Q != 0
       (if Q = 0, Bob aborts the procedure);
      divides Q by H1(A|B|PW) to get Xab, the recovered value of g^Ra

```
  B --> A: {Y = H2(A|B|PW)*(g^Rb), S1 = H3(A|B|PW|Xab|g^Rb|(Xab)^Rb)}
           (The above precondition on PW ensures that Y != 0)

     Alice
       verifies that Y != 0;
       divides Y by H2(A|B|PW) to get Yba, the recovered value of g^Rb,
       and computes S1' = H3(A|B|PW|g^Ra|Yba|(Yba)^Ra);
       authenticates Bob by checking whether S1' = S1;
       if authenticated, then sets key K = H5(A|B|PW|g^Ra|Yba|(Yba)^Ra)


  A --> B:  S2 = H4(A|B|PW|g^Ra|Yba|(Yba)^Ra)

     Bob
       Computes S2' = H4(A|B|PW|Xab|g^Rb|(Xab)^Rb) and
       authenticates Alice by checking whether S2' = S2;
       if authenticated, then sets K = H5(A|B|PW|Xab|g^Rb|(Xab)^Rb)
```

   If any of the above verifications fails, the protocol halts;
   otherwise, both parties have authenticated each other and established
   the key.

4. Selection of Parameters

   This section provides guidance on selection of the PAK parameters.
   First, it addresses general considerations, then it reports on
   specific implementations.

4.1. General Considerations

   In general implementations, the parameters must be selected to meet
   algorithm requirements of [BMP00].

4.2. Over-the-Air Service Provisioning (OTASP) and Wireless Local Area
      Network (WLAN) Diffie-Hellman Parameters and Key Expansion
      Functions

   [OTASP], [TIA683], and [WLAN] pre-set public parameters p and g to
   their "published" values. This is necessary to protect against an
   attacker sending bogus p and g values, tricking the legitimate user
   to engage in improper Diffie-Hellman exponentiation and leaking some
   information about the password.

   According to [OTASP], [TIA683], and [WLAN], g shall be set to
   00001101, and p to the following 1024-bit prime number (most
   significant bit first):

```
0xFFFFFFFF  0xFFFFFFFF  0xC90FDAA2  0x2168C234  0xC4C6628B
0x80DC1CD1  0x29024E08  0x8A67CC74  0x020BBEA6  0x3B139B22
0x514A0879  0x8E3404DD  0xEF9519B3  0xCD3A431B  0x302B0A6D
0xF25F1437  0x4FE1356D  0x6D51C245  0xE485B576  0x625E7EC6
0xF44C42E9  0xA637ED6B  0x0BFF5CB6  0xF406B7ED  0xEE386BFB
0x5A899FA5  0xAE9F2411  0x7C4B1FE6  0x49286651  0xECE65381
0xFFFFFFFF  0xFFFFFFFF
```

In addition, if short exponents [MP05] are used for Diffie-Hellman parameters Ra and Rb, then they should have a minimum size of 384 bits. The independent, random functions H1 and H2 should each output 1152 bits, assuming prime p is 1024 bits long and session keys K are 128 bits long. H3, H4, and H5 each output 128 bits. More information on instantiating random functions using hash functions can be found in [BR93]. We use the FIPS 180 SHA-1 hashing function [FIPS180] below to instantiate the random function as done in [WLAN]; however, SHA-256 can also be used:

H1(z):
SHA-1(1|1|z) mod 2^128 | SHA-1(1|2|z) mod 2^128 |...|
| SHA-1(1|9|z) mod 2^128

H2(z):
SHA-1(2|1|z) mod 2^128 | SHA-1(2|2|z) mod 2^128 |...|
| SHA-1(2|9|z) mod 2^128

H3(z): SHA-1(3|len(z)|z|z) mod 2^128
H4(z): SHA-1(4|len(z)|z|z) mod 2^128
H5(z): SHA-1(5|len(z)|z|z) mod 2^128

In order to create 1152 output bits for H1 and H2, nine calls to SHA-1 are made and the 128 least significant bits of each output are used. The input payload of each call to SHA-1 consists of:

a) 32 bits of function type, which for H1 is set to 1 and for H2 is set to 2;
b) a 32-bit counter value, which is incremented from 1 to 9 for each call to SHA-1;
c) the argument z [for (A|B|PW)].

The functions H3, H4, and H5 require only one call to the SHA-1 hashing function and their respective payloads consist of:

a) 32 bits of function type (e.g., 3 for H3);
b) a 32-bit value for the bit length of the argument z;
c) the actual argument repeated twice.

Finally, the 128 least significant bits of the output are used.

5. Security Considerations

    Security considerations are as follows:

    - Identifiers

       Any protocol that uses PAK must specify a method for producing a
       single representation of identity strings.

    - Shared secret

       PAK involves the use of a shared secret. Protection of the shared
       values and managing (limiting) their exposure over time is
       essential and can be achieved using well-known security policies
       and measures. If a single secret is shared among more than two
       entities (e.g., Alice, Bob, and Mallory), then Mallory can
       represent himself as Alice to Bob without Bob being any the wiser.

    - Selection of Diffie-Hellman parameters

       The parameters p and g must be carefully selected in order not to
       compromise the shared secret. Only previously agreed-upon values
       for parameters p and g should be used in the PAK protocol. This
       is necessary to protect against an attacker sending bogus p and g
       values and thus tricking the other communicating party in an
       improper Diffie-Hellman exponentiation. Both parties also need to
       randomly select a new exponent each time the key-agreement
       protocol is executed. If both parties re-use the same values,
       then Forward Secrecy property is lost.

       In addition, if short exponents Ra and Rb are used, then they
       should have a minimum size of 384 bits (assuming that 128-bit
       session keys are used). Historically, the developers, who strived
       for 128-bit security (and thus selected 256-bit exponents), added
       128 bits to the exponents to ensure the security reduction proofs.
       This should explain how an "odd" length of 384 has been arrived
       at.

    - Protection against attacks

       a) There is a potential attack, the so-called discrete logarithm
          attack on the multiplicative group of congruencies modulo p, in
          which an adversary can construct a table of discrete logarithms
          to be used as a "dictionary". A sufficiently large prime, p,
          must be selected to protect against such an attack. A proper
          1024-bit value for p and an appropriate value for g are
          published in [WLAN] and [TIA683]. For the moment, this is what
          has been implemented; however, a larger prime (i.e., one that

is 2048 bits long, or even larger) will definitely provide
better protection. It is important to note that once this is
done, the generator must be changed too, so this task must be
approached with extreme care.

    b) An online password attack can be launched by an attacker by
repeatedly guessing the password and attempting to
authenticate. The implementers of PAK should consider
employing mechanisms (such as lockouts) for preventing such
attacks.

- Recommendations on H() functions

The independent, random functions H1 and H2 should output 1152
bits each, assuming prime p is 1024 bits long and session keys K
are 128 bits long. The random functions H3, H4, and H5 should
output 128 bits.

An example of secure implementation of PAK is provided in [Plan9].

6. Acknowledgments

The authors are grateful for the thoughtful comments received from
Shehryar Qutub, Ray Perlner, and Yaron Sheffer. Special thanks go to
Alfred Hoenes, Tim Polk, and Jim Schaad for their careful reviews and
invaluable help in preparing the final version of this document.

7. References

7.1. Normative References

[X.1035]   ITU-T, "Password-authenticated key exchange (PAK)
              protocol", ITU-T Recommendation X.1035, 2007.

[TIA683]   TIA, "Over-the-Air Service Provisioning of Mobile
              Stations in Spread Spectrum Systems", TIA-683-D, May
              2006.

7.2. Informative References

[Plan9]   Alcatel-Lucent, "Plan 9 from Bell Labs",
              http://netlib.bell-labs.com/plan9/.

[BMP00]   Boyko, V., MacKenzie, P., and S. Patel, "Provably secure
              password authentication and key exchange using Diffie-
              Hellman", Proceedings of Eurocrypt 2000.

   [BR93]   Bellare, M. and P. Rogaway, "Random Oracles are
                Practical: A Paradigm for Designing Efficient Protocols",
                Proceedings of the 5th Annual ACM Conference on Computer
                and Communications Security, 1998.

   [DH76]   Diffie, W. and M.E. Hellman, "New directions in
                cryptography", IEEE Transactions on Information Theory 22
                (1976), 644-654.

   [FIPS180]   NIST Federal Information Processing Standards,
                Publication FIPS 180-3, "Secure Hash Standard", 2008.

   [MP05]   MacKenzie, P. and S. Patel, "Hard Bits of the Discrete
                Log with Applications to Password Authentication", CT-RSA
                2005.

   [OTASP]   3GPP2, "Over-the-Air Service Provisioning of Mobile
                Stations in Spread Spectrum Systems", 3GPP2 C.S0016-C v.
                1.0 5, October 2004.

   [RFC2631]   Rescorla, E., "Diffie-Hellman Key Agreement Method", RFC
                2631, June 1999.

   [WLAN]   3GPP2, "Wireless Local Area Network (WLAN) Interworking",
                3GPP2 X.S0028-0, v.1.0, April 2005.

Authors' Addresses

    Alec Brusilovsky
    Alcatel-Lucent
    Room 9B-226, 1960 Lucent Lane
    Naperville, IL 60566-7217 USA
    Tel: +1 630 979 5490
    EMail: Alec.Brusilovsky@alcatel-lucent.com

    Igor Faynberg
    Alcatel-Lucent
    Room 2D-144, 600 Mountain Avenue
    Murray Hill, NJ 07974 USA
    Tel: +1 908 582 2626
    EMail: igor.faynberg@alcatel-lucent.com

    Sarvar Patel
    Google, Inc.
    76 Ninth Avenue
    New York, NY 10011 USA
    Tel: +1 212 565 5907
    EMail: sarvar@google.com

    Zachary Zeltsan
    Alcatel-Lucent
    Room 2D-150, 600 Mountain Avenue
    Murray Hill, NJ 07974 USA
    Tel: +1 908 582 2359
    EMail: zeltsan@alcatel-lucent.com