Network Working Group
Internet-Draft
Intended status: Informational
Expires: November 22, 2013

F. Hao, Ed.
Newcastle University (UK)

May 21, 2013

J-PAKE: Password Authenticated Key Exchange by Juggling
draft-hao-jpake-00

Abstract

This document specifies a Password Authenticated Key Exchange by Juggling (J-PAKE) protocol. This protocol allows the establishment of a secure end-to-end communication channel between two remote parties over an insecure network solely based on a shared password, without requiring a Public Key Infrastructure (PKI) or any trusted third party.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Password-Authenticated Key Exchange (PAKE) is a technique that aims to establish secure communication between two remote parties solely based on their shared password, without relying on a Public Key Infrastructure or any trusted third party [BM92]. The first PAKE protocol, called EKE, was proposed by Steven Bellovin and Michael Merrit in 1992 [BM92]. Other well-known PAKE protocols include SPEKE (by David Jablon in 1996) [Jab96] and SRP (by Tom Wu in 1998) [Wu98]. SRP has been revised several times in response to reported attacks. The last version of SRP is 6, hence it is also known as SRP-6 [RFC5054].

This document specifies a PAKE protocol called Password Authenticated Key Exchange by Juggling (J-PAKE), which was designed by Feng Hao and Peter Ryan in 2008 [HR08]. There are a few factors that may be considered in favor of J-PAKE. First, J-PAKE has security proofs, while equivalent proofs are lacking in EKE, SPEKE and SRP-6. Second, J-PAKE is not patented. It follows a completely different design

approach from all other PAKE protocols, and is the first PAKE scheme that is built upon a well-established Zero Knowledge Proof (ZKP) primitive: Schnorr signature. Third, J-PAKE is efficient. It adopts novel engineering techniques to optimize the use of ZKP so that overall the protocol is sufficiently efficient for practical use. Fourth, J-PAKE is designed to work generically in both the finite field and elliptic curve setting (i.e., DSA and ECDSA-like groups). Unlike SPEKE, it does not require any extra primitive of hashing passwords onto a designated elliptic curve. Finally, J-PAKE has already been used in real-world applications at a relatively large scale. Since 2008, it has been included into widely distributed open source libraries such as OpenSSL, OpenSSH, Network Security Services (NSS) and the Bouncy Castle. In 2010, it was adopted by Mozilla and built into the Firefox browser (version 4 and onwards) to implement the secure sync service.

1.1. Requirements language

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].

1.2. Notations

The following notations are used in this document:

o Alice: the assumed identity of the first party in the protocol

o Bob: the assumed identify of the second party in the protocol

o s: a low-entropy secret shared between Alice and Bob

o p: a large prime

o q: a large prime divisor of p-1

o Zp*: a multiplicative group of integers modulo p

o Gq: a subgroup of Zp* with primer order q

o g: a generator of Gq

o g^x: g raised to the power of x

o a mod b: a modulo b

o a * b: a multiplied by b

- a || b: concatenation of a and b

- H: a secure one-way hash function

- KDF(a): Key Derivation Function with input a

- HMAC(MacKey, MacData): HMAC function with MacKey as the key and MacData as the input data

2. J-PAKE Protocol

2.1. Protocol setup

The J-PAKE protocol uses exactly the same group setting as DSA (or ECDSA). For simplicity, this document will only describe the J-PAKE protocol in the DSA-like group setting. The protocol works basically the same in the ECDSA-like group setting, except that the underlying multiplicative group over a finite field is replaced by an additive group over an elliptic curve.

Let Gq denote a subgroup of Zp* with prime order q, in which the Decisional Diffie-Hellman problem (DDH) is intractable. The p and q are large primes and q divides p-1. Let g be a generator in Gq. Any non-identity element in Gq can be a generator. The two communicating parties, Alice and Bob, both agree on (p, q, g). Values of (p, q, g), as defined by NIST, can be found in the appendix of [I-D-Schnorr]. [[: Q1:The reference is an accompanying internet draft submission to IETF and it needs to be updated once it is accepted by IETF.]]

Let s be the shared secret between Alice and Bob. The secret may be a password, a hash of the password or any other derivative from a password. This does not make any difference to the protocol. The only assumptions are that s has low-entropy and that the value of s falls within [1, q-1]. (Note that s must not be 0 for any non-empty secret.)

2.2. Two-round key exchange

Round 1: Alice selects at random x1 from [0, q-1] and x2 from [1, q-1]. Similarly, Bob selects at random x3 from [0, q-1] and x4 from [1, q-1].

- Alice -> Bob: g^{x1} mod p, g^{x2} mod p and knowledge proofs for x1 and x2

- Bob -> Alice: g^{x3} mod p, g^{x4} mod p and knowledge proofs for x3 and x4

In this round, the sender must demonstrate the knowledge of the ephemeral private keys. A suitable technique is to use the Schnorr signature [I-D-Schnorr]. As an example, suppose one wishes to prove the knowledge of the exponent for X = g^x mod p. The generated Schnorr signature will contain: {SignerID, V = g^v mod p, r = v - x * h mod q} where SignerID is the unique identifier for the signer, v is a random element chosen from [0, q-1] and h = H(g || V || X || SignerID). The "uniqueness" of SignerID is defined from the user's perspective -- for example, if Alice communicates with several parties, she shall associate a unique identity with each party to avoid confusion. During the key exchange process using J-PAKE, each party shall ensure the two identities used by both parties are different (to prevent replaying one's signature to herself) and that the other party has been consistently using the same identity throughout the protocol execution. Details about the Schnorr signature, including the generation and the verification procedures, can be found in [I-D-Schnorr].

When this round finishes, Alice verifies the received knowledge proofs as specified in [I-D-Schnorr] and also checks that g^{x4} != 1 mod p. Similarly, Bob verifies the received knowledge proofs and also checks that g^{x2} != 1 mod p.

Round 2:

o Alice -> Bob: A=g^{(x1+x3+x4)*x2*s} mod p and a knowledge proof for (x2*s)

o Bob -> Alice: B=g^{(x1+x2+x3)*x4*s} mod p and a knowledge proof for (x4*s)

In this round, the Schnorr signature is computed in the same way as in the previous round except that the generator is different. For Alice, the generator used is g^(x1+x3+x4) instead of g; for Bob, the generator is g^(x1+x2+x3) instead of g. Since any non-identity element in Gq can be used as a generator, Alice and Bob just need to ensure g^(x1+x3+x4) != 1 mod p and g^(x1+x2+x3) != 1 mod p. With overwhelming probability, these inequalities are statistically guaranteed even when the user is communicating with an adversary (i.e., in an active attack). Nonetheless, for absolute guarantee, the receiving party may wish to explicitly check if these inequalities hold, and the cost of doing that is negligible.

When the second round finishes, Alice and Bob verify the received knowledge proofs and then compute key material K as follows:

o Alice computes K = (B/g^{x2*x4*s})^{x2} mod p = g^{(x1+x3)*x2*x4*s} mod p

o Bob computes K = (A/g^{x2*x4*s})^{x4} mod p = g^{(x1+x3)*x2*x4*s} mod p

With the same keying material K, both parties can derive a common session key k using a Key Derivation Function (KDF). If the subsequent secure communication uses a symmetric cipher in an authenticated mode (say AES-GCM), then one key is sufficient, i.e., k = KDF(K). Otherwise, the session key should comprise an encryption key (for confidentiality) and a MAC key (for integrity), i.e., k = k_enc || k_mac, where k_enc = KDF(K || "JPAKE_ENC") and k_mac = KDF(K || "JPAKE_MAC"). The exact choice of the KDF is left to specific applications to define. (In many cases, the KDF can simply be a cryptographic hash function, e.g., SHA-256.)

2.3. Three-pass variant

The two-round J-PAKE protocol is completely symmetric, which significantly simplifies the security analysis. In practice, one party normally initiates the communication and the other party responds. In that case, the protocol will be completed in three passes instead of two rounds. The two-round J-PAKE protocol can be trivially changed to three passes without losing security. Assume Alice initiates the key exchange. The three-pass variant works as follows:

1. Alice -> Bob: g^{x1} mod p, g^{x2} mod p, knowledge proofs for x1 and x2.

2. Bob -> Alice: g^{x3} mod p, g^{x4} mod p, B=g^{(x1+x2+x3)*x4*s mod p, knowledge proofs for x3, x4, and x4*s.

3. Alice -> Bob: A=g^{(x1+x3+x4)*x2*s} mod p and a knowledge proof for x2*s.

Both parties compute the session keys in exactly the same way as before.

2.4. Key confirmation

The two-round J-PAKE protocol (or three-pass variant) provides cryptographic guarantee that only the authenticated party who used the same password at the other end is able to compute the same session key. So far the authentication is only implicit.

For achieving explicit authentication, an additional key confirmation procedure should be performed. This is to ensure that both parties have actually obtained the same session key.

There are several explicit key confirmation methods available. They are generically applicable to all key exchange protocols, not just J-PAKE. In general, it is recommended to use a different key from the session key for key confirmation, say using k' = KDF(K || "JPAKE_KC"). The advantage of using a different key for key confirmation is that the session key remains indistinguishable from random after the key confirmation process (although this perceived advantage is actually subtle and only theoretical). Two key confirmation methods are presented here.

The first method is based on the one used in the SPEKE protocol [Jab96].

1. Alice -> Bob: H(H(k'))

2. Bob -> Alice: H(k')

The second method is based on the unilateral key confirmation scheme specified in NIST SP 800-56A Revision 1 [BJS07].

o Alice -> Bob: MacTagAlice = HMAC(k', "KC_1_U || Alice || Bob || g^x1 || g^x2 || g^x3 || g^x4")

o Bob -> Alice: MacTagBob = HMAC(k', "KC_1_U || Bob || Alice || g^x3 || g^x4 || g^x1 || g^x2")

The second method assumes an additional secure MAC function (HMAC) and is slightly more complex than the first method; however, it can be completed within one round and it preserves the overall symmetry of the protocol implementation. This may prove desirable in some applications.

2.5. Computational cost

In J-PAKE, the modular exponentiations are the predominant factors in the computation. Hence, the computational cost is estimated based on counting the number of such modular exponentiations. Note that it takes one exponentiation to generate a Schnorr signature and two to verify it [I-D-Schnorr]. For Alice, she has to perform 8 exponentiations in the first round, 4 in the second round, and 2 in the final computation of the session key. Hence, that is 14 modular exponentiations in total. Based on symmetry, the computational cost for Bob is exactly the same.

3. Security Considerations

A PAKE protocol is designed to provide two functions in one protocol execution. The first one is to provide zero-knowledge authentication

of a password. It is called "zero knowledge" because at the end of the protocol, the two communicating parties will learn nothing more than one bit information: whether the passwords supplied at two ends are equal. Therefore, a PAKE protocol is naturally resistant against phishing attacks. The second function is to provide session key establishment if the two passwords are equal. The session key will be used to protect the confidentiality and integrity of the subsequent communication.

More concretely, a secure PAKE protocol shall satisfy the following security requirements [HR10].

1. Off-line dictionary attack resistance: It does not leak any information that allows a passive/active attacker to perform off-line exhaustive search of the password.

2. Forward secrecy: It produces session keys that remain secure even when the password is later disclosed.

3. Known-key security: It prevents a disclosed session key from affecting the security of other sessions.

4. On-line dictionary attack resistance: It limits an active attacker to test only one password per protocol execution.

First, a PAKE protocol must resist off-line dictionary attacks. A password is inherently weak. Typically, it has only about 20-30 bits entropy. This level of security is subject to exhaustive search. Therefore, in the PAKE protocol, the communication must not reveal any data that allows an attacker to learn the password through off-line exhaustive search.

Second, a PAKE protocol must provide forward secrecy. The key exchange is authenticated based on a shared password. However, there is no guarantee on the long-term secrecy of the password. A secure PAKE scheme shall protect past session keys even when the password is later disclosed. This property also implies that if an attacker knows the password but only passively observes the key exchange, he cannot learn the session key.

Third, a PAKE protocol must provide known key security. A session key lasts throughout the session. An exposed session key must not cause any global impact on the system, affecting the security of other sessions.

Finally, a PAKE protocol must resist on-line dictionary attacks. If the attacker is directly engaging in the key exchange, there is no way to prevent such an attacker trying a random guess of the

password. However, a secure PAKE scheme should mitigate the effect of the on-line attack to the minimum. In the best case, the attacker can only guess exactly one password per impersonation attempt. Consecutively failed attempts can be easily detected and the subsequent attempts can be thwarted accordingly.

It has been proven in [HR10] that J-PAKE satisfies all of the four requirements based on the assumptions that there exists a secure cryptographic hash function and that the Decisional Diffie-Hellman problem is intractable. By comparison, it has been known that EKE has the problem of leaking partial information about the password to a passive attacker, hence not satisfying the first requirement [Jas96]. For SPEKE and SRP-6, an attacker may be able to test more than one password in one on-line dictionary attack (see [Zha04] and [Hao10]), hence they do not satisfy the fourth requirement in the strict theoretical sense.

4. IANA Considerations

This document has no actions for IANA.

5. Acknowledgements

The editor of this document would like to thank Dr Dylan Clarke for useful comments. This work was supported by the EPSRC First Grant EP /J011541/1.

6. References

6.1. Normative References

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

[RFC5054] Taylor, D., Wu, T., Mavrogiannopoulos, N., and T. Perrin, "Using the Secure Remote Password (SRP) Protocol for TLS Authentication", RFC 5054, November 2007.

[BM92] Bellovin, S. and M. Merrit, "Encrypted Key Exchange: Password-based Protocols Secure against Dictionary Attacks", IEEE Symposium on Security and Privacy, May 1992.

[HR08] Hao, F. and P. Ryan, "Password Authenticated Key Exchange by Juggling", 16th Workshop on Security Protocols (SPW'08), May 2008.

[HR10] Hao, F. and P. Ryan, "J-PAKE: Authenticated Key Exchange Without PKI", Springer Transactions on Computational Science XI, 2010.

[Jab96] Jablon, D., "Strong Password-Only Authenticated Key Exchange", ACM Computer Communications Review, October 1996.

[Wu98] Wu, T., "The Secure Remote Password protocol", Symposimum on Network and Distributed System Security, March 1998.

[I-D-Schnorr] Hao, F., "Schnorr Signature: Non-interactive Zero Knowledge Proof for Discrete Logarithm", Internet Draft submitted to IETF, 2013.

6.2. Informative References

[BJS07] 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>.

[Jas96] Jaspan, B., "Dual-Workfactor Encrypted Key Exchange: Efficiently Preventing Password Chaining and Dictionary Attacks", USENIX Symphosium on Security, July 1996.

[Zha04] Zhang, M., "Analysis of The SPEKE Password-Authenticated Key Exchange Protocol", IEEE Communications Letters, January 2004.

[Hao10] Hao, F., "On Small Subgroup Non-Confinement Attacks", IEEE conference on Computer and Information Technology, 2010.

Author's Address

Feng Hao (editor)
Newcastle University (UK)
Claremont Tower, School of Computing Science, Newcastle University
Newcastle Upon Tyne
United Kingdom

Phone: +44 (0)192-222-6384
EMail: feng.hao@ncl.ac.uk