Internet Research Task Force
Internet-Draft
Intended status: Informational
Expires: May 4, 2017

Harkins
HP Enterprise
October 31, 2016

PKEX
draft-harkins-pkex-01

Abstract

This memo describes a password-authenticated protocol to allow two devices to exchange "raw" (uncertified) public keys and establish trust that the keys belong to their respective identities.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Many authenticated key exchange protocols allow for authentication using uncertified, or "raw", public keys. Usually these specifications-- e.g. [RFC7250] for TLS and [RFC7670] for IKEv2-- assume keys are exchanged in some out-of-band mechanism.

[RFC7250] further states that "the main security challenge [to using 'raw' public keys] is how to associate the public key with a specific entity. Without a secure binding between identifier and key, the protocol will be vulnerable to man-in-the- middle attacks."

The Public Key Exchange (PKEX) is designed to fill that gap: it establishes a secure binding between exchanged public keys and identifiers, it provides proof-of-possession of the exchanged public keys to each peer, and it enables the establishment of trust in public keys that can subsequently be used to faccilitate authentication in other authentication and key exchange protocols.

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. Notation

This memo describes a cryptographic exchange using sets of elements called groups. Groups can be either traditional finite field or can be based on elliptic curves. The public keys exchanged by PKEX are elements in a group. Elements in groups are denoted in upper-case and scalar values are denoted with lower-case. The generator of the group is G.

When both the initator and responder use a similar, but unique, datum it is denoted by appending an "i" for initiator or "r" for responder, e.g. if each side needs an element C then the initiator's is Ci and the responder's is Cr.

During the exchange, one side will generate data and the other side will attempt to reconstruct it. The reconstructed data is "primed". That is, if the initiator generates C then when responder tries to reconstruct it, the responder will refer to it as C'. Data that is directly sent and received is not primed.

The following notation is used in this memo:

C = A + B
: The "group operation" on two elements, A and B, that produces a third element, C. For finite field cryptography this is the modular multiplication, for elliptic curve cryptography this is point addition.

C = a * B
: This denotes repeated application of the group operation to B-- i.e. B + B-- (a - 1) times.

a = H(b)
: A cryptographic hash function that takes data b of indeterminate length and returns a fixed sized digest a.

a = F(B)
: A mapping function that takes an element and returns a scalar. For elliptic curve cryptography, F() returns the x-coordinate of the point B. For finite field cryptography, F() is the identity function.

a = KDF-b(c, d)
: A key derivation function that derives an output key a of length b from an input key c and context d.

c = a | b
: Concatentation of data a with data b producing c.

{a}b
Authenticated-encryption of data a with key b.

2. Properties

Subversion of PKEX involves an adversary being able to insert its own public key into the exchange without the exchange failing, resulting in one of the parties to the exchange believing the adversary's public key actually belongs to the protocol peer.

PKEX has the following properties:

- An adversary is unable to subvert the exchange without knowing the password.

- An adversary is unable to discover the password through passive attack.

- The only information exposed by an active attack is whether a single guess of the password is correct or not.

- Proof-of-possession of the private key is provided.

- At the end of the protocol, either trust is established in the peer's public key and the public key is bound to the peer's identity, or the exchange fails.

3. Assumptions

Due to the nature of the exchange, only DSA ([DSS]) and ECDSA ([X9.62]) keys can be exchanged with PKEX.

PKEX requires fixed elements that are unique to the particular role in the protocol, an initiator-specific element and a responder-specific element. They need not be secret. It is assumed that both parties know the role-specific elements for the particular group in which their key pairs were derived. This memo does not proscribe any way to generate these role-specific elements but the "Hunting and Pecking" technique of [RFC7664] could be used with a slight variation. Instead of inputting a password and generating a secret element, a common string such as "PKEX Initiator Element" can be used to generate a public element. For elliptic curve cryptography, the technique of "hashing into an elliptic curve" from [hash2ec] could be used, again with a common string, to produce role-specific elements.

The authenticated-encryption algorithm provides deterministic "key wrapping". To achieve this the AE scheme used in PKEX is [RFC5297].

The KDF provides for the generation of a cryptographically strong secret key from an "imperfect" source of randomness. To achieve this the KDF used in PKEX is the unsalted version of [RFC5869].

The following assumptions are made on PKEX:

- Only the peers involved in the exchange know the password.
- The peers' public keys are from the same group.
- The discrete logarithms of the public role-specific elements are unknown, and determining them is computationally infeasible.

4. Cryptographic Primitives

HKDF requires an underlying hash function and AES-SIV requires a key length. To provide for consistent security the hash algorithm and key length depend on the group chosen to use with PKEX.

For ECC, the hash algorithm and key length depends on the size of the prime defining the curve, p:

- SHA-256 and 256 bits: when len(p) <= 256
- SHA-384 and 384 bits: when 256 < len(p) <= 384
- SHA-512 and 512 bits: when 384 < len(p)

For FFC, the hash algorithm depends on the prime, p, defining the finite field:

- SHA-256 and 256 bits: when len(p) <= 2048
- SHA-384 and 384 bits: when 2048 < len(p) <= 3072
- SHA-512 and 512 bits: when 3072 < len(p)

5. Protocol Definition

PKEX is a balanced PAKE. The identical version of the password is used by both parties.

PKEX consists of two phases: exchange and commit/reveal. It is described using the popular protocol participants, Alice (an initiator of PKEX), and Bob (a responder of PKEX).

We denote Alice's role-specific element a Pi and Bob's as Pr. The password is pw. For simplicity, Alice's identity is "Alice" and

Bob's identity is "Bob". Alice's public key she wants to share with Bob is A and her private key is a, while Bob's public key he wants to share with Alice is B and his private key is b.

5.1. Exchange Phase

The Exchange phase is essentially the SPAKE2 key exchange. The peers derive ephemeral public keys, encrypt, and exchange them. Each party hashes a concatentation of his or her identity and the password and operates on the role-specific element to obtain a secret encrypting element. The group operation is then performed with the ephemeral key and the secret encrypting element to produce an encrypted ephmeral key.

```
      Alice:                                 Bob:
      ------                                 ----
  x, X = x*G                               y, Y = y*G
  Qi = H(Alice|pw)*Pi                      Qr = H(Bob|pw)*Pr
  M = X + Qa
                         M ------>
                                           Qi = H(Alice|pw)*Pi
                                           X' = M - Qi
                                           N = Y + Qr
                         <------ N
  Qr = H(Bob|pw)*Pr
  Y' = N - Qr
```

Both M and N MUST be verified to be valid elements in the selected group. If either one is not valid the protocol fails.

At this point in time the peers have exchanged ephemeral elements that will be unknown except by someone with knowledge of the password. Given our assumptions that means only Alice and Bob can know the elements X and Y.

The secret encrypting elements are irretrievably deleted at this point.

5.2. Commit/Reveal Phase

In the Commit/Reveal phase the peers commit to the particular public key they wish to exchange and then reveal it to the peer.

```
         Alice:                                Bob:
         ------                                ----
     ka = KDF-n(F(a*Y'), F(M) | F(N) |
              F(A) | F(Y') | pw)
     u = HMAC(ka, F(X) | F(Y') |
              F(A) | Alice | 0)
     z = KDF-n(F(x*Y'), F(M) | F(N) |
              F(X) | F(Y') | pw)

                          {A, u}z ------>

                                       z = KDF-n(F(y*X'), F(M) | F(N) |
                                           F(X') | F(Y) | pw)
                                       if (SIV-decrypt returns fail) fail
                                       if (A not valid element) fail
                                       ka' = KDF-n(F(y*A), F(M) | F(N) |
                                                 F(A) | F(Y) | pw)
                                       u' = HMAC(ka', F(X') | F(Y) |
                                                 F(A) | Alice | 0)
                                       if (u' != u) fail
                                       kb = KDF-n(F(b*X'), F(N) | F(M) |
                                                F(B) | F(X') | pw)
                                       v = HMAC(kb, F(Y) | F(X') |
                                                F(B) | Bob | 1)

                                <------ {B, v}z

     if (SIV-decrypt returns fail) fail
     if (B not valid element) fail
     kb' = KDF-n(F(x*B'), F(N) | F(M) |
               F(B') | F(X) | pw)
     v' = HMAC(kb', F(Y') | F(X) |
               F(B') | Bob | 1)
     if (v'!= v) fail
```

where 0 and 1 are single octets of the value zero and one, respectively, n is the key length from Section 4, and both the KDF and HMAC use the hash algorithm from Section 4.

If the parties didn't fail they have each other's public key, knowledge that the peer possesses the corresponding private key, and trust that the public key belongs to the peer's stated identity.

6. IANA Considerations

This memo could create a registry of the fixed public elements for a nice cross section of popular groups. Or not. If it ends up doing so there will be IANA Considerations here, otherwise there won't be.

7. Security Considerations

The encrypted shares exchanged in the Exchange phase MUST be ephemeral. Reuse of these keys, even with a different password, voids the security of the exchange.

The discrete logaritm of the fixed public elements MUST not be known. Knowledge of either of these values voids the security of the exchange.

The public keys exchanged in PKEX are never disclosed to an attacker, either passive or active. While they are, as the name implies, public, PKEX provides for secrecy of the exchanged keys for any protocol that might need such a capability.

PKEX has forward secrecy in the sense that exposure of the password used in a previous run of the protocol will not affect the security of that run.

There is no proof of security of PKEX at this time but the Exchange phase is SPAKE2 and the security proof for that protocol can be used to help prove the security of PKEX.

8. References

8.1. Normative References

[DSS] U.S. Department of Commerce/National Institute of Standards and Technology, "Digital Signature Standard (DSS)", Federal Information Processing Standards FIPS PUB 186-4, July 2013.

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

[RFC5297] Harkins, D., "Synthetic Initialization Vector (SIV) Authenticated Encryption Using the Advanced Encryption Standard (AES)", RFC 5297, DOI 10.17487/RFC5297, October 2008, <http://www.rfc-editor.org/info/rfc5297>.

[RFC5869] Krawczyk, H. and P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)", RFC 5869, DOI 10.17487/RFC5869, May 2010, <http://www.rfc-editor.org/info/rfc5869>.

[X9.62] American National Standards Institute, "X9.62-2005", Public Key Cryptography for the Financial Services Industry (ECDSA), 2005.

8.2. Informative References

[RFC7250] Wouters, P., Ed., Tschofenig, H., Ed., Gilmore, J., Weiler, S., and T. Kivinen, "Using Raw Public Keys in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", RFC 7250, DOI 10.17487/RFC7250, June 2014, <http://www.rfc-editor.org/info/rfc7250>.

[RFC7664] Harkins, D., Ed., "Dragonfly Key Exchange", RFC 7664, DOI 10.17487/RFC7664, November 2015, <http://www.rfc-editor.org/info/rfc7664>.

[RFC7670] Kivinen, T., Wouters, P., and H. Tschofenig, "Generic Raw Public-Key Support for IKEv2", RFC 7670, DOI 10.17487/RFC7670, January 2016, <http://www.rfc-editor.org/info/rfc7670>.

[hash2ec] Coron, J-S. and T. Icart, "An indifferentiable hash function into elliptic curves", Cryptology ePrint Archive Report 2009/340, 2009.

Appendix A. Appendix

Maybe show a sample PKEX exchange

Author's Address

Dan Harkins
HP Enterprise
1322 Crossman avenue
Sunnyvale, California 94089
USA

Phone: +1 415 997 9834
Email: dharkins@lounge.org