Network Working Group
Internet-Draft
Intended status: Informational
Expires: May 29, 2016

D. Miller
OpenSSH
S. Josefsson
SJD AB

November 26, 2015

The chacha20-poly1305@openssh.com authenticated encryption cipher
draft-josefsson-ssh-chacha20-poly1305-openssh-00

Abstract

This document describes the chacha20-poly1305@openssh.com authenticated encryption cipher supported by OpenSSH

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on May 29, 2016.

Table of Contents

1. Introduction

ChaCha20 is a stream cipher designed by Daniel Bernstein and described in [ChaCha]. It operates by permuting 128 fixed bits, 128 or 256 bits of key, a 64 bit nonce and a 64 bit counter into 64 bytes of output. This output is used as a keystream, with any unused bytes simply discarded.

Poly1305 [Poly1305], also by Daniel Bernstein, is a one-time Carter-Wegman MAC that computes a 128 bit integrity tag given a message and a single-use 256 bit secret key.

The "chacha20-poly1305@openssh.com" combines these two primitives into an authenticated encryption mode. The construction used is based on that proposed for TLS by Adam Langley in [I-D.agl-tls-chacha20poly1305], but differs in the layout of data passed to the MAC and in the addition of encyption of the packet lengths.

2. Negotiation

The chacha20-poly1305@openssh.com offers both encryption and authentication. As such, no separate MAC is required. If the chacha20-poly1305@openssh.com cipher is selected in key exchange, the offered MAC algorithms are ignored and no MAC is required to be negotiated.

3. Detailed Construction

The chacha20-poly1305@openssh.com cipher requires 512 bits of key material as output from the SSH key exchange. This forms two 256 bit keys (K_1 and K_2), used by two separate instances of chacha20.

The instance keyed by K_1 is a stream cipher that is used only to encrypt the 4 byte packet length field. The second instance, keyed by K_2, is used in conjunction with poly1305 to build an AEAD (Authenticated Encryption with Associated Data) that is used to encrypt and authenticate the entire packet.

Two separate cipher instances are used here so as to keep the packet lengths confidential but not create an oracle for the packet payload cipher by decrypting and using the packet length prior to checking the MAC. By using an independently-keyed cipher instance to encrypt the length, an active attacker seeking to exploit the packet input handling as a decryption oracle can learn nothing about the payload contents or its MAC (assuming key derivation, ChaCha20 and Poly1305 are secure).

The AEAD is constructed as follows: for each packet, generate a Poly1305 key by taking the first 256 bits of ChaCha20 stream output generated using K_2, an IV consisting of the packet sequence number encoded as an uint64 under the SSH wire encoding rules and a ChaCha20 block counter of zero. The K_2 ChaCha20 block counter is then set to the little-endian encoding of 1 (i.e. {1, 0, 0, 0, 0, 0, 0, 0}) and this instance is used for encryption of the packet payload.

4. Packet Handling

When receiving a packet, the length must be decrypted first. When 4 bytes of ciphertext length have been received, they may be decrypted using the K_1 key, a nonce consisting of the packet sequence number encoded as a uint64 under the usual SSH wire encoding and a zero block counter to obtain the plaintext length.

Once the entire packet has been received, the MAC MUST be checked before decryption. A per-packet Poly1305 key is generated as described above and the MAC tag calculated using Poly1305 with this key over the ciphertext of the packet length and the payload together. The calculated MAC is then compared in constant time with the one appended to the packet and the packet decrypted using ChaCha20 as described above (with K_2, the packet sequence number as nonce and a starting block counter of 1).

To send a packet, first encode the 4 byte length and encrypt it using K_1. Encrypt the packet payload (using K_2) and append it to the encrypted length. Finally, calculate a MAC tag and append it.

5. Rekeying

ChaCha20 must never reuse a {key, nonce} for encryption nor may it be used to encrypt more than 2^70 bytes under the same {key, nonce}. The SSH Transport protocol [RFC4253] recommends a far more conservative rekeying every 1GB of data sent or received. If this recommendation is followed, then chacha20-poly1305@openssh.com requires no special handling in this area.

6. Acknowledgements

Markus Friedl helped on the design.

7. References

7.1. Normative References

[RFC4253] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Transport Layer Protocol", RFC 4253, DOI 10.17487/RFC4253, January 2006, <http://www.rfc-editor.org/info/rfc4253>.

[ChaCha] Bernstein, J., "ChaCha, a variant of Salsa20", January 2008, <http://cr.yp.to/chacha/chacha-20080128.pdf>.

[Poly1305] Bernstein, J., "The Poly1305-AES message-authentication code", March 2005, <http://cr.yp.to/mac/poly1305-20050329.pdf>.

7.2. Informative References

[I-D.agl-tls-chacha20poly1305] Langley, A. and W. Chang, "ChaCha20 and Poly1305 based Cipher Suites for TLS", draft-agl-tls-chacha20poly1305-04 (work in progress), November 2013.

Appendix A. Copying conditions

Authors' Addresses

Damien Miller
OpenSSH

Simon Josefsson
SJD AB

Email: simon@josefsson.org