OpenPGP Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 21, 2016

B. Ford
EPFL

October 19, 2015

Modernizing the OpenPGP Message Format
draft-ford-openpgp-format-00

Abstract

This draft proposes and solicits discussion on methods of modernizing OpenPGP's encrypted message format to support more state-of-the-art authenticated encryption schemes, and optionally to protect format metadata as well as data via metadata encryption and judicious padding.

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 April 21, 2016.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Overview and Rationale

The current OpenPGP message format [RFC4880] has evolved periodically to support new cryptographic algorithms, but its structure embodies assumptions and imposes limitations that are overdue for reconsideration and modernization based on today's cryptographic best-practices and evolved threat models. This draft proposes, as a starting point for discussion, several of these issues and potential approaches to modernizing the OpenPGP format to address them.

2. Adopting Authenticated Encryption Schemes

The OpenPGP format currently handles symmetric-key encryption and integrity services as separate, orthogonal mechanisms. It is now widely accepted in the cryptographic community, however, that it is often advantageous to both performance and security to roll symmetric-key encryption and identity protection together into a single cryptographic abstraction, now commonly known as Authenticated Encryption with Additional Data or AEAD. An increasingly rich body of AEAD schemes is now available that considerably reduce computation cost with respect to the traditional approach of applying encryption and hash-based identity protection as separate, orthogonal steps.

Enhancing the OpenPGP format to support AEAD schemes will involve two

main updates to the OpenPGP format specification: (1) defining a suitable AEAD-based alternative encoding for the current Symmetrically Encrypted Integrity Protected Data packet (Tag 18, section 5.13 of [RFC4880]); and (2) defining at least one concrete AEAD scheme usable in this new data Encrypted Data packet format.

The sections below first propose a first-cut AEAD Data Packet format, then briefly point out several possible AEAD algorithms for consideration.

2.1. AEAD Protected Data Packet

The proposed AEAD Protected Data packet (tentatively Tag 20) contains data that is both encrypted and integrity-protected by a single AEAD algorithm, defined by the selected symmetric-key cipher. Symmetric-key AEAD algorithms occupy the same identifier space as traditional symmetric ciphers such as IDEA and Twofish (listed in section 9.2 of [RFC4880]), but require the use of AEAD Protected Data packets exclusively. That is, an OpenPGP message whose selected symmetric-key algorithm is an AEAD algorithm MUST use the AEAD Protected Data packet, while a message whose selected symmetric-key algorithm is not an AEAD algorithm MUST NOT use the AEAD Protected Data packet.

AEAD algorithms in general take as inputs: (1) a symmetric secret key, (2) a nonce or IV whose presence and size depends on the algorithm, (3) a variable-length body to be encrypted and identity protected, and (4) an "additional data" field to be identity protected but not encrypted (often as header and/or trailer metadata). The AEAD algorithm produces: (1) a ciphertext containing the encrypted content of the variable-length body, and (2) a fixed-length authenticator protecting the integrity of both the encrypted body and the additional data.

In OpenPGP's specific use of an AEAD algorithm, the symmetric secret key input is defined by the OpenPGP Session Key as conveyed in the message's public-key and/or symmetric-key ESK packets.

In the context of OpenPGP, there is no clear need for the "additional data" feature of AEAD schemes (in contrast with the uses of AEAD schemes to encrypt packets or datagrams), so we tentatively propose that the "additional data" field always be considered to be empty (0 bytes) in the context of OpenPGP. (DISCUSS: are we missing potential

uses of this that might warrant inclusion of some field or extension allowing the AD part to be nonempty?)

The AEAD Protected Data packet in an OpenPGP message contains the following octet sequences, directly concatenated: (1) the nonce or IV required by the AEAD algorithm, if any, encoded as a fixed-length header whose size is determined by the symmetric-key scheme; (2) the variable-length ciphertext representing the AEAD-encrypted body; and (3) the authenticator, as a fixed-length trailer whose size is determined by the symmetric-key scheme. Note that symmetric-key AEAD schemes MAY expand the size of the body during encryption (e.g., due to internal metadata and/or padding), but if so, MUST enable the

decrypting side to determine the true size of the original variable-length cleartext (e.g., by including any necessarily metadata within the encrypted ciphertext to indicate how much padding was added to the plaintext before encryption).

When an AEAD symmetric-key cipher is used, the Modification Detection Code Packet (Tag 19) MUST NOT be used. (DISCUSS: can anyone identify any benefit to placing the authenticator in a separate packet? The justification for the current proposal is that in all the AEAD schemes I'm aware of the authenticator is fixed-size and thus has no need for additional size metadata.)

DISCUSS: in nonce-based AEAD schemes, the nonce is technically not needed (or can be taken to be all 0's) if the symmetric-key is used only once, which is likely to be at least the common case for OpenPGP where the AEAD symmetric key is the one-time session key. Thus, we could save the size of the nonce provided there is only ever at most one encrypted data packet. The downside is the risk of a security disaster if any implementation ever (incorrectly) produces multiple AEAD Protected Data packets using the same key.

2.2. Concrete AEAD Schemes

The proposed AEAD enhancement will require the definition of at least one and perhaps multiple concrete AEAD schemes to be specified for use with OpenPGP. We propose the following choices as starting points for discussion, deferring for now the instantiation details of each:

2.2.1. AES-GCM

The AES cipher operated in Galois-Counter Mode (GCM) [GCM] has become a well-accepted AEAD scheme used in other Internet standards [RFC5288][RFC4106] and has no known serious cryptographic weaknesses. Thus, AES-GCM is likely to be a reasonable choice for inclusion in an AEAD extension to OpenPGP, even if it is does not necessarily represent the current state-of-the-art in performance or security.

2.2.2. ChaCha20-Poly1305

The ChaCha20 stream cipher used with the Poly1305 authenticator [RFC7539] has gained considerable traction as a practical alternative to AES-GCM providing high performance especially in tuned software implementations, believed to offer security comparable to or better than AES-GCM, and based on contrasting cryptographic foundations.

2.2.3. Keccak-based sponge scheme

The Keccak sponge function forms the cryptographic core of the recently-standardized SHA-3 family of hash algorithms [SHA3]. As a sponge construction, Keccak offers an attractive basis for AEAD schemes because the sponge construction can currently "absorb" bits for integrity protection and "produce" pseudorandom bits for encryption, while adding no significant overhead above the cost of one or the other. As SHA-3 has received substantial public attention and cryptanalysis, it represents a safe choice from a security perspective, and is based on substantially different cryptographic foundations from either of the above choices, offering further diversity. A particular Keccak-based AEAD construction would need to be selected, such as the well-known MonkeyDuplex [MONKEY] among other reasonable choices.

2.2.4. Future: CAESAR competition winner

The CAESAR competition [CAESAR] is in the process of selecting a new AEAD scheme for public recognition. The winner will not automatically become a formal standard per se but may become a "de facto" standard due to the extensive public cryptanalysis all the

competitors are currently undergoing, and as such will represent an obvious potential choice for future standardization in an AEAD-enhanced OpenPGP message format.

2.3. Metadata Leakage-Hardening the OpenPGP Format

The current OpenPGP format encodes a considerable amount of metadata about an OpenPGP-encrypted encrypted file "in the clear": for example, (1) the fact that it is an OpenPGP-encrypted file, (2) exactly which public-key and/or symmetric-key algorithms the file is encrypted with, (3) whether or not the file can be decrypted with a passphrase, (4) whether or not the file can be decrypted with a public/private keypair, and if so how many distinct keypairs can be used to decrypt the file, and (5) the length of the encrypted message. See [METADATA] for an illustration of this metadata.

While this unencrypted metadata was not thought to be privacy-sensitive when the OpenPGP format was first designed, the evolution of today's threats have called this assumption into question. For example, the very existence on a hard drive of a file that is readily identifiable as OpenPGP-encrypted can arouse suspicion and has been known to lead airport, border-control, and other authorities of some countries to demand passwords or decryption keys under threat of incarceration even if it is not clear that the holder of the device is in possession of the necessary decryption keys. Furthermore, as the state-of-the-art in cryptanalysis and brute-force attacks

gradually overtakes the security of older cryptographic schemes, the existence of a cryptographic scheme identifier in cleartext effectively acts as a "crack me!" flag, making it unnecessarily easy for an attacker to invest computational resources selectively into cracking ciphertexts known to use weak cryptographic schemes, while avoiding wasting compute resources attempting to crack ciphertexts encrypted under stronger schemes. The number of distinct public keys that can decrypt a file can serve to identify the group of people for which the file was encrypted. Finally, even the file's length can represent sensitive, possibly incriminating information especially in known-plaintext situations, e.g., when an attacker suspects but cannot otherwise prove that an OpenPGP file on a suspected whistleblower's or dissident's hard disk is an encryption of a particular document.

We therefore suggest for discussion two possible measures for the further evolution of the OpenPGP format to reduce this metadata leakage: encrypted metadata, and optional length padding.

2.3.1. Encrypting file format metadata

OpenPGP's current format makes the decryption process "easy" in the sense that it is immediately clear to the decryptor which cryptographic algorithms should be used to decrypt the file, at the cost of the metadata leaks above. It is readily feasible to define a new OpenPGP format in which no metadata is left unencrypted, leaving the encrypted file's contents appearing to be a "Uniformly Random Blob" or URB.

The obvious challenge such a format change presents is that the decryptor will not know a priori which encryption scheme(s) were used to encrypt a particular file, and hence would simply have to try in turn each of the schemes it supports. For files protected only by a single passphrase, implementing full metadata protection in this fashion is straightforward. While it may seem likely to incur significant cost, note that the decryptor need not attempt to decrypt the entire file using each scheme, but only a short header portion, before either successfully identifying the scheme in use (or giving up if the passphrase is wrong and/or the scheme is unsupported).

A fully-encrypted-metadata format is more challenging in the general case of files encrypted using a combination of one or more passphrases and/or one or more public keypairs, but still readily feasible, so as to require the decryptor to perform only one expensive "trial" public-key operation per scheme (not per key) on a file encrypted with any number of symmetric and/or public keys. Details will be expanded on if the WG decides this to be a direction potentially worth pursuing.

2.3.2. Intelligent padding to minimize size-based leakage

Even if the directly-encoded metadata of an OpenPGP file is encrypted as discussed above, the file's mere length can still represent significant leakage, likely immediately revealing the existence of a known plaintext on a hard drive for example. The only "perfect" solution from a security perspective - is to pad all encrypted files to a common length - is obviously impractical from an efficiency

perspective.

A second, more conceivable but still costly choice would be to pad files to (for example) the next power-of-two in size. This reduces the maximum possible information leakage from an N-byte file from O(log N) to O(log log N), but the up-to-100% expansion factor (50% expansion on average) is significant and likely to be a considerable deterrent against use.

A better choice would be to use a slightly more sophisticated padding scheme, which pads any encrypted file into "size buckets" chosen to limit maximum information leakage to O(log log N) - asymptotically equivalent to the simple next-power-of-two scheme - while ensuring that no file incurs more than about a 10% expansion and large files incur progressively smaller expansion factors (e.g., no more than 3% for files 1MB or larger). Details of this scheme will be expanded if the WG deems this direction potentially worth pursuing.

In combination with the above encrypted-metadata techniques, the resulting benefit is that (new) OpenPGP-encrypted messages or files would be substantially more "anonymous" than they are now, at least within the set of plaintexts whose ciphertext lengths fall into one of these padded "size buckets." Furthermore, since the padding scheme need not be specific to OpenPGP, the result would be that metadata-protected, encrypted files produced by any application designed to use the same padding scheme would produce objects cryptogrphically indistinguishable from others in the same "size bucket" across every application supporting a compatible padding scheme. Thus, the resulting "Padded Uniform Random Blobs" or PURBs could eventually provide metadata protection and some level of "encrypted file anonymity" not only within the context of one application (e.g., OpenPGP) but across different applications that produce PURBs in quite different ways.

3. Security Considerations

No new security considerations (beyond those that already apply to OpenPGP's existing message format) have been identified so far, but likely will be.

4. References

4.1. Normative References

[GCM] Dworkin, M., "Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC", NIST Special Publication 800-38D, November 2007, <http://csrc.nist.gov/publications/nistpubs/800-38D/SP-800-38D.pdf>.

[MONKEY] Bertoni, G., Daemen, J., Peeters, M., and G. Van Assche, "Permutation-based encryption, authentication and authenticated encryption", Directions in Authenticated Ciphers 2012, August 2015, <http://keccak.noekeon.org/KeccakDIAC2012.pdf>.

[RFC4106] Viega, J. and D. McGrew, "The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (ESP)", RFC 4106, DOI 10.17487/RFC4106, June 2005, <http://www.rfc-editor.org/info/rfc4106>.

[RFC4880] Callas, J., Donnerhacke, L., Finney, H., Shaw, D., and R. Thayer, "OpenPGP Message Format", RFC 4880, DOI 10.17487/RFC4880, November 2007, <http://www.rfc-editor.org/info/rfc4880>.

[RFC5288] Salowey, J., Choudhury, A., and D. McGrew, "AES Galois Counter Mode (GCM) Cipher Suites for TLS", RFC 5288, DOI 10.17487/RFC5288, August 2008, <http://www.rfc-editor.org/info/rfc5288>.

[RFC7539] Nir, Y. and A. Langley, "ChaCha20 and Poly1305 for IETF Protocols", RFC 7539, DOI 10.17487/RFC7539, May 2015, <http://www.rfc-editor.org/info/rfc7539>.

4.2. Informative References

[METADATA] Underwood, M., "The information leaked from a gpg encrypted file.", October 2015, <https://drive.google.com/file/d/0BwK1bcoczINteWVwVFN5UWNORW8/view?usp=sharing>.

Author's Address

Bryan Ford
EPFL
BC 210, Station 14
Lausanne CH-1015
Switzerland

Phone: +41 21 693 28 73
Email: bryan.ford@epfl.ch