

RFC 4880 : OpenPGP Message Format

Stéphane Bortzmeyer

<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 22 Novembre 2007

Date de publication du RFC : Novembre 2007

<http://www.bortzmeyer.org/4880.html>

Le logiciel PGP est synonyme de cryptographie pour beaucoup de gens. Un des plus anciens et des plus utilisés pour les fonctions de confidentialité mais aussi d'authentification, PGP est très célèbre mais on sait peu que ses messages sont normalisés, dans ce RFC.

PGP a vu son format de données normalisé pour la première fois en août 1996, dans le RFC 1991¹. Cette norme a été révisée par la suite, dans le RFC 2440 et notre RFC est la dernière version.

Cette normalisation permet à diverses mises en œuvre de PGP d'interopérer. La plus connue aujourd'hui est la seule libre, GNU Privacy Guard (qui n'existait pas encore au moment de la publication du premier RFC). Il ne faut donc plus confondre le **logiciel** PGP, écrit à l'origine par Phil Zimmermann, et qui est non-libre, avec le **format** PGP que décrit notre RFC et que des logiciels autres que PGP peuvent lire et écrire.

Le principe du chiffrement avec PGP est simple. Une **clé de session** (le terme est impropre puisqu'il n'y a pas de session au sens de TLS mais c'est celui utilisé par le RFC) est créée pour chaque destinataire, elle sert à chiffrer le message et cette clé est chiffrée avec la clé publique du destinataire (section 2.1 du RFC).

Pour l'authentification, c'est aussi simple conceptuellement. Le message est résumé et le résumé est chiffré avec la clé privée de l'émetteur (section 2.2 du RFC).

Le format PGP permet également la compression (qui améliore la sécurité en supprimant les redondances) et l'encodage en Base64 (RFC 4648), baptisé "*ASCII armor*", pour passer à travers des logiciels qui n'aiment pas le binaire (la section 6 détaille cet encodage).

1. Pour voir le RFC de numéro NNN, <http://www.ietf.org/rfc/rfcNNN.txt>, par exemple <http://www.ietf.org/rfc/rfc1991.txt>

La section 3 explique les éléments de base utilisés par le format PGP. L'un des plus importants est le concept d'entier de grande précision (MPI pour "*Multi-Precision Integers*"), qui permet de représenter des entiers de très grande taille, indispensables à la cryptographie, sous forme d'un doublet longueur + valeur.

Enfin les sections 4 et 5 expliquent le format lui-même. Un message PGP est constitué de **paquets** (rien à voir avec les paquets réseau). Chaque paquet a un type, une longueur et un contenu. Par exemple, un paquet de type 1 est une clé de session chiffrée, un paquet de type 2 une signature, un paquet de type 9 du contenu chiffré, etc.

La section 7 du RFC décrit un type de message un peu particulier, qui n'obéit pas à la syntaxe ci-dessus, les messages en clair mais signés. Ces messages ont l'avantage de pouvoir être lus sans avoir de logiciel PGP. Ils nécessitent donc des règles spéciales.

On notera que `gpg` permet d'afficher les paquets présents dans un message PGP, ce qui est pratique pour l'apprentissage ou le débogage. Voyons un exemple avec un fichier `test.txt` de 35 octets, signé mais non chiffré :

```
% gpg --list-packets test.gpg
:compressed packet: algo=1
:onepass_sig packet: keyid 4136479797D6D246
    version 3, sigclass 00, digest 2, pubkey 17, last=1
:literal data packet:
    mode b (62), created 1191502517, name="test.txt",
    raw data: 35 bytes
:signature packet: algo 17, keyid 4136479797D6D246
    version 3, created 1191502517, md5len 5, sigclass 00
    digest algo 2, begin of digest 0d e4
    data: [159 bits]
    data: [158 bits]
```

Malheureusement, `gpg` n'affiche pas les valeurs numériques des types, telles que listées par le RFC. Mais les noms qu'il utilise sont les mêmes que dans le RFC, on peut donc facilement trouver la section qui explique ce qu'est un "onepass_sig packet".

Avec un message chiffré, on obtient :

```
% gpg --list-packets test.txt.gpg
:pubkey enc packet: version 3, algo 16, keyid F3A0253D6C2A95F9
    data: [1022 bits]
    data: [1023 bits]
:pubkey enc packet: version 3, algo 16, keyid C0FE90B0F08F74D7
    data: [2044 bits]
    data: [2048 bits]

You need a passphrase to unlock the secret key for
user: "Stephane Bortzmeyer (Personl address) <stephane@bortzmeyer.org>"
2048-bit ELG-E key, ID F08F74D7, created 2000-03-31 (main key ID 97D6D246)

:encrypted data packet:
    length: 102
    mdc_method: 2
gpg: encrypted with 1024-bit ELG-E key, ID 6C2A95F9, created 2005-02-18
    "Kim Minh Kaplan <kaplan@kim-minh.com>"
gpg: encrypted with 2048-bit ELG-E key, ID F08F74D7, created 2000-03-31
    "Stephane Bortzmeyer (Personl address) <stephane@bortzmeyer.org>"
:compressed packet: algo=2
:literal data packet:
    mode b (62), created 1191502765, name="test.txt",
    raw data: 35 bytes
```

On notera que c'est après l'affichage des premiers paquets que gpg demande la phrase de passe pour lire la clé privée. En effet, les premiers paquets ne sont pas chiffrés, puisqu'ils indiquent la clé à utiliser pour déchiffrer le reste (PGP a une option pour masquer ces paquets, cf. section 5.1).

Malheureusement, notre RFC n'indique pas clairement les changements par rapport à la version précédente, le RFC 2440. Mais les changements ne sont pas radicaux, la version reste la même. Les principaux ajouts sont les suivants :

- Le précédent RFC contenait une note de l'IESG expliquant qu'il n'y avait pas de mécanisme d'extension, par exemple pour ajouter de nouveaux types ou sous-types de paquets. Ce n'est plus le cas et la section 10 détaille comment demander à l'IANA d'ajouter de nouveaux paramètres.
- Il y a moins de support des vieilles versions de PGP (l'obligation de les accepter est passée de MUST à SHOULD, cf. RFC 2119.)
- Il y a de nouveaux types de paquet comme le paquet "*User Attributes*", plus riche que le paquet "*User ID*", avec notamment la possibilité d'ajouter des images.
- Un nouveau format de compression est ajouté, bzip2.
- La section 14 (analyse de sécurité) est bien plus détaillée, avec description des attaques possibles.
- Parmi les nouvelles options, les signatures peuvent désormais être marquées comme confirmées par un tiers de confiance, via les "*Third-Party Confirmation signature*".