RFC 7366 : Encrypt-then-MAC for TLS and DTLS

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 16 septembre 2014. Dernière mise à jour le 22 février 2015

Date de publication du RFC : Septembre 2014

https://www.bortzmeyer.org/7366.html

Depuis ses débuts, le protocole de cryptographie TLS (héritier de SSL) protège l'intégrité des paquets transmis en calculant un MAC, qui est inclus dans les données avant le chiffrement, une technique qu'on nomme "*MAC-then-encrypt*". Plusieurs failles de sécurité ont été depuis identifiées dans cette technique et ce nouveau RFC normalise une extension à TLS qui permet de faire l'inverse, chiffrer avant de calculer le MAC, ce qui est aujourd'hui considéré comme plus sûr. On nomme cette méthode, logiquement, "*encrypt-then-MAC*". (Notez qu'il existe une méthode encore plus sûre, le chiffrement intègre, systématisé à partir de TLS 1.3.)

Lorsque le protocole SSL, ancêtre de TLS, avait été normalisé, au milieu des années 1990, le "*MAC-then-encrypt*" semblait tout à fait sûr. C'est ainsi que l'actuelle norme TLS, le RFC 5246 [1] (section 6.2.3.1), continue à utiliser "*MAC-then-encrypt*". Des études comme celle de M. Bellare et C. Namprempre, « "*Authenticated Encryption : Relations among notions and analysis of the generic composition paradigm*" <http://cseweb.ucsd.edu/~mihir/papers/oem.html> » ou celle de H. Krawczyk, « "*The Order of Encryption and Authentication for Protecting Communications (or : How Secure Is SSL?)*" <https://eprint.iacr.org/2001/045> » ont depuis montré que "*MAC-then-encrypt*" est vulnérable à un certain nombre d'attaques cryptanalytiques, dans certaines conditions d'utilisation (je simplifie : la question de la vulnérabilité exacte de "*MAC-then-encrypt*" est plus compliquée que cela). Il est désormais recommandé d'utiliser plutôt "*encrypt-then-MAC*" (je simplifie encore : le RFC note par exemple que les algorithmes qui ne séparent pas chiffrement et calcul du MAC n'ont pas besoin de cette méthode).

Pour sélectionner la méthode recommandée, le groupe de travail TLS <https://tools.ietf.org/wg/tls> de l'IETF a choisi (section 2 de notre RFC) d'utiliser une extension à TLS (RFC 5246, section 7.4.1.4). Le client ajoute dans son message de bienvenue (`client_hello`) l'extension `encrypt_then_mac` (le `extension_type` de `encrypt_then_mac` vaut 22 et est désormais dans le registre

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc5246.txt

IANA <https://www.iana.org/assignments/tls-extensiontype-values/tls-extensiontype-valu
xhtml#tls-extensiontype-values-1>). Si le serveur TLS est d'accord, il mettra cette extension dans son `server_hello`.

Une autre solution (que d'utiliser le mécanisme d'extensions) aurait été de définir de nouveaux algorithmes de chiffrement et elle a fait l'objet de nombreux débats au sein du groupe de travail. Mais cela menait à une explosion du nombre d'algorithmes ("*ciphers*"), il aurait fallu presque doubler le nombre d'algorithmes concernés, pour avoir une version "*MAC-then-encrypt*" (l'actuelle) et une version "*encrypt-then-MAC*". Encore une autre solution aurait été de créer une nouvelle version de TLS, la 1.3, où "*encrypt-then-MAC*" aurait été le comportement standard. Quand on sait que la version actuelle de TLS, la 1.2, est toujours minoritaire dans la nature, on voit qu'une telle solution aurait sérieusement retardé le déploiement. Au contraire, avec la solution choisie, il « suffira » de changer quelques dizaines de lignes de code dans les bibliothèques TLS actuelles. Un tel changement a des chances de se retrouver plus rapidement sur le terrain.

Comme le mécanisme d'extension est apparu avec TLS (il n'existait pas dans SSL, même dans sa dernière version, la 3), "*encrypt-then-MAC*" ne sera pas utilisable pour les logiciels qui utilisent encore SSL v3 (le cas principal semblant être la version 6 d'Internet Explorer). Comme SSL est abandonné depuis quinze ans, des logiciels aussi vieux ont probablement des tas d'autres failles de sécurité plus graves que l'utilisation de "*MAC-then-encrypt*". (Au passage, c'est une des raisons pour lesquelles ce modeste blog n'accepte pas SSLv3 <https://www.bortzmeyer.org/https-blog.html>.)

Arrêtons la discussion sur les alternatives qui auraient été possibles et revenons à la nouvelle extension (section 3 de notre RFC). Une fois qu'elle est acceptée par le client et le serveur, le traitement des paquets par TLS passera de l'ancien comportement `encrypt (data || MAC || pad)` (où `||` désigne la concaténation) au nouveau `encrypt (data || pad) || MAC`. Le MAC est calculé comme avant, sauf qu'il part du texte chiffré (`TLSCiphertext`) et plus du texte en clair (`TLSCompressedtext`, cf. RFC 5246, section 6.2.3.1). La structure de données TLS qui était (RFC 5246, section 6.2.3) :

```
struct {
          ContentType type;
          ProtocolVersion version;
          uint16 length;
          GenericBlockCipher fragment; /* Après moultes discussions,
  le groupe de travail a décidé de n'appliquer
  encrypt-then-MAC qu'aux "block ciphers". */
      } TLSCiphertext;
```

va devenir :

```
struct {
          ContentType type;
          ProtocolVersion version;
          uint16 length;
          GenericBlockCipher fragment;
          opaque MAC;  /* Ce champ supplémentaire est la seule
  nouveauté */
      } TLSCiphertext;
```

Pour le déchiffrement, on fait les opérations en sens inverse, on vérifie le MAC d'abord, on jette immédiatement le paquet si le MAC est incorrect (en renvoyant un message `bad_record_mac`), sinon on déchiffre. Ainsi, un attaquant qui modifie les données en vol ne peut rien apprendre sur les clés

cryptographiques utilisées : toute modification invalidera le MAC et tous les calculs d'un MAC incorrect prendront le même temps, ne permettant pas d'attaque par mesure du temps.

À noter que TLS permet d'utiliser un MAC raccourci (RFC 6066) et cela reste possible avec "*encrypt-then-MAC*", dans les mêmes conditions.

Comme avec toute extension qui améliore la sécurité, il y a un risque qu'un attaquant actif perturbe la négociation (section 4 du RFC) pour amener les deux parties à ne pas choisir la nouvelle extension (attaque par repli), par exemple en forçant une utilisation de vieilles versions du protocole, n'ayant pas d'extensions (et donc pas de "*encrypt-then-MAC*"). La seule protection est de refuser le repli, ce qui peut provoquer des problèmes d'interopérabilité avec certains vieux logiciels (voir par exemple le ticket #1084025 chez Mozilla <https://bugzilla.mozilla.org/show_bug.cgi?id=1084025>).

Il y a un serveur TLS public qui gère cette extension, si vous voulez tester votre code client : <https://eid.vx4.net>. Au 14 septembre 2014, la gestion de cette extension ne figure pas encore dans la version publiée d'OpenSSL, la 1.0.1i mais est dans la version de développement (voir le commit <http://git.openssl.org/gitweb/?p=openssl.git;a=commit;h=5e3ff62c345c976cd1ffbcc5e6042f55264977f5>. Par contre, encore rien dans GnuTLS (la version publiée est la 3.3.7).

Merci à Florian Maury pour sa relecture et à Manuel Pégourié-Gonnard pour avoir corrigé une faute sur l'attaque par repli..

Une bonne explication technique de ces problèmes est celle de Moxie Marlinspike <http://www.thoughtcrime.org/blog/the-cryptographic-doom-principle/>.