RFC 9750 : The Messaging Layer Security (MLS) Architecture

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 23 avril 2025

Date de publication du RFC : Avril 2025

https://www.bortzmeyer.org/9750.html

On le sait, un des problèmes stratégiques de l'Internet est l'absence d'un protocole de messagerie instantanée standard et largement répandu. Autant le courrier électronique fonctionne relativement bien entre logiciels différents, autant la messagerie instantanée est un monde de jardins clos, incompatibles entre eux, et qui nécessite d'installer vingt applications différentes sur son ordiphone. L'interopérabilité reste un objectif lointain. Il n'y a pas de solution simple à ce problème mais le système MLS ("*Messaging Layer Security*") vise à au moins fournir un mécanisme de sécurité pour les groupes, mécanisme qui peut ensuite être utilisé par un protocole complet, comme XMPP (RFC 6120 [1]). MLS est normalisé dans le RFC 9420 et cet autre RFC, le RFC 9750, décrit l'architecture générale.

Les services de sécurité que doit offrir MLS sont les classiques (confidentialité, notamment) et, bien sûr, doivent être assurés de bout en bout (l'opérateur du service ne doit **pas** pouvoir casser la sécurité et, par exemple, ne doit pas pouvoir lire les messages). Quand il n'y a que deux participants, c'est relativement facile mais MLS est prévu pour des groupes, allant de deux à potentiellement des centaines de milliers de participant·es.

Le protocole, on l'a vu, est spécifié dans le RFC 9420. Il repose sur l'envoi de messages de maintenance du groupe, comme le message `Proposal`, qui demande un changement dans la composition du groupe, ou `Commit`, qui va créer une nouvelle "*epoch*", c'est-à-dire un nouvel état du groupe (la ressemblance avec le "*commit*" d'un VCS comme git est volontaire). Les clés effectivement utilisées pour le chiffrement sont liées à cet état, et changent avec lui.

Le protocole MLS repose sur deux services, qui doivent être fournis pour qu'il puisse fonctionner (voir aussi la section 7) :

1. Pour voir le RFC de numéro NNN, `https://www.ietf.org/rfc/rfcNNN.txt`, par exemple `https://www.ietf.org/rfc/rfc6120.txt`

— Un service d'authentification des utilisateurices, l'AS ("*Authentication Service*", section 4 du RFC), qui permettre d'obtenir les clés publiques, avec la garantie qu'elles sont bien liées à l'identificateur utilisé par l'utilisateurice,
— Un service de livraison, le DS ("*Delivery Service*", section 5), qui va se charger de porter les messages.

Il n'y a pas besoin de faire confiance au DS, tous les messages étant chiffrés. La seule possibilité d'action d'un DS malveillant serait de ne pas transmettre les messages. (Par contre, un AS malveillant pourrait casser toute la sécurité, en envoyant des clés publiques mensongères, permettant par exemple l'espionnage.)

MLS n'impose pas une forme particulière à l'AS et au DS. Par exemple, il pourrait utiliser une PKI (RFC 5280) traditionnelle comme AS. On l'a dit, MLS n'est pas une solution complète de messagerie instantanée, une telle solution nécessite aussi l'AS, le DS et plein d'autres choses. Parmi les systèmes existants, on peut noter que PGP repose sur les serveurs de clés et le "*Web of Trust*" comme AS et sur le courrier comme DS. Signal repose sur un système centralisé comme DS et sur une vérification des clés par les utilisateurices comme AS (voir aussi la section 8.4.3.1 de notre RFC, sur l'authentfication des utilisateurices).

Le RFC rappelle que MLS ne met pas de contrainte à ces messages de changement du groupe. Si on veut le faire (par exemple créer des groupes fermés au public), cela doit être fait dans le protocole complet, qui inclut MLS pour la partie « chiffrement au groupe ». Par contre, MLS impose que tous les membres du groupe connaissent tous les autres membres. Il n'y a pas de possibilité d'avoir des membres cachés (cela serait difficilement compatible avec le chiffrement de bout en bout). Et le DS peut, selon les cas, être capable de trouver ou de déduire la composition d'un groupe (voir section 8.4.3.2).

Mais alors, que fait MLS ? Il gère la composition des groupes et ses changements dans le temps, et le matériel cryptographique associé. Via le DS, tous les membres du groupe génèrent les clés secrètes nécessaires et les messages peuvent être chiffrés pour tout le monde. MLS ne comprend pas le format des messages, il chiffre des octets, sans avoir besoin de connaitre l'application au-dessus.

Ah, et un peu de sécurité (section 8 du RFC). Il est très recommandé de faire tourner MLS sur un transport chiffré, comme QUIC (RFC 9000) ou TCP+TLS (RFC 8446) car, même si certains messages sont chiffrés, des métadonnées ne le sont pas. MLS fonctionne sur l'hypothèse que tout ce qui n'est pas une extrêmité du réseau (une des machines qui communiquent) est un ennemi (RFC 3552).

Autre question de sécurité, les "*push tokens*", qui ont fait parler d'eux lors de discussions sur la sécurité de certaines messageries instantanées. Ils sont utilisés pour la distribution de contenu, par exemple pour prévenir les utilisateurices qu'un nouveau message est arrivé. Sans système de "*push*", les machines clients devraient périodiquement interroger les serveurs, ce dont leurs batteries se plaindraient certainement. Mais les "*push tokens*" ont l'inconvénient de pouvoir être liés à des informations permettant d'identifier un·e utilisateurice <https://blog.davidlibeau.fr/push-notifications-are-a-p>. Cela ne compromet pas le contenu des messages (qui est chiffré) mais donne accès à des métadonnées bien révélatrices. Et les États utilisent cette possibilité <https://arstechnica.com/tech-policy/2023/12/apple-admits-to-secretly-giving-governments-push-notification-data/>. Le problème est compliqué car il n'y a pas de solution de remplacement, à part l'attente active. C'est ainsi que Proton a noté <https://www.reddit.com/r/IAmA/comments/18czv7w/comment/kcdyv6s/> : « "*That said, we will continue to use Apple and Google push notifications when the services are available on the device because unfortunately they are favored heavily by the operating system in terms of performance and battery life.*" » Mais Tuta a fait un autre choix <https://tuta.com/fr/blog/open-source-email-fdroid>. Notre RFC suggère quelques pistes comme l'introduction de délais aléatoires.

La section 8 donne beaucoup d'autres conseils de sécurité et sa lecture est recommandée. Notez par exemple que le protocole MLS du RFC 9420 a fait l'objet de plusieurs analyses de sécurité (la liste figure dans la section 8.6).

Il existe un site Web du projet MLS <https://messaginglayersecurity.rocks/> mais il a très peu de contenu. Question mises en œuvre de MLS, une liste est disponible <https://github.com/mlswg/mls-implementations>, citant, par exemple, la bibliothèque BouncyCastle ou bien OpenMLS <https://openmls.tech/> (avec une documentation très complète <https://book.openmls.tech/>).