RFC 9954 : Hybrid key exchange in TLS 1.3

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 16 juillet 2026

Date de publication du RFC : Juillet 2026

https://www.bortzmeyer.org/9954.html

Dans le monde merveilleux de la cryptographie post-quantique (RFC 9958[1]), un problème se pose : si en voulant éviter le Charybde des calculateurs quantiques, on tombe dans le Scylla d'un algorithme, certes post-quantique, mais pour lequel une attaque cryptanalytique est trouvée ? N'aurait-on pas laché la proie pour l'ombre ? La solution, que ce RFC décrit pour le cas de l'échange de clés <https://www.bortzmeyer.org/echange-cles.html> dans TLS, est la cryptographie hybride : on utilise à la fois un algorithme traditionnel et un post-quantique (RFC 9794). Un éventuel attaquant devrait alors casser les deux pour réussir.

Si vous n'êtes pas à l'aise avec la terminologie de la cryptographie post-quantique, la section 1.2 de notre RFC la rappelle. Notamment, un algorithme **traditionnel** est un de ceux qu'on utilisait avant la menace des calculateurs quantiques, comme par exemple RSA ou ECDH. Un algorithme post-quantique est un algorithme dont on a de bonnes raisons de penser qu'il résistera aux futurs CRQC ("*Cryptographically-Relevant Quantum Computer*", les calculateurs quantiques capables de traiter des problèmes de cryptographie réels). Un algorithme de **nouvelle génération** est un de ceux récemment créés et dont on n'est pas complètement sûrs de la sécurité. Cela inclut certains algorithmes post-quantiques, par exemple ML-KEM. Un algorithme **hybride** combine un traditionnel et un post-quantique. On parle de clé composée quand elle est faite de deux clés, une traditionnelle et une post-quantique, gérées ensemble comme un seul algorithme. Mais les termes ne sont pas universellement adoptés : comme « hybride » a un autre sens en cryptographie (un protocole qui utilise à la fois la cryptographie symétrique et l'asymétrique, comme TLS ou OpenPGP), certains préfèrent dire « composé » pour tout système PQ/T (post-quantique et traditionnel).

« Nouvelle génération » pour parler (entre autres) des algorithmes post-quantiques est inhabituel mais le but est de montrer que la technique décrite dans ce RFC ne se limite pas au post-quantique : elle

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc9958.txt

s'applique dès qu'on n'a aucun algorithme parfait et qu'on en compose deux, que ces algorithmes soient post-quantiques ou pas.

Et puisqu'on pinaille sur le terminologie, le RFC note que TLS utilise le terme de « groupe » pour parler des algorithmes d'échange de clé alors que tous ces algorithmes ne font pas référence à un groupe. Mais l'habitude s'est prise.

Pourquoi créer des systèmes hybrides, PQ/T, plutôt que, par exemple, attendre tranquillement que les algorithmes post-quantiques soient mieux testés, et qu'on leur fasse autant confiance qu'à, par exemple, RSA ? Avoir un seul algorithme serait plus simple. Mais le but des hybrides est de permettre aux adopteurs précoces de se lancer tout de suite dans le post-quantique, en étant sûrs de ne pas diminuer leur sécurité actuelle. (D'autant plus que l'utilisation des algorithmes traditionnels peut être une obligation réglementaire, par exemple pour respecter FIPS.) Ces adopteurs précoces peuvent être motivés, entre autres, par la crainte d'un décryptage rétroactif, lorsqu'un attaquant qui n'a pas de CRQC aujourd'hui enregistre quand même les communications, pour les décrypter dès qu'il pourra acheter un CRQC sur AliExpress. Si vous gérez des secrets qui ont une longue durée de vie (secrets d'État…), vous ne devez pas attendre qu'un CRQC existe réellement, même s'il n'arrive que dans 20 ans, certains secrets durent plus longtemps que ça.

Bon, maintenant, au boulot ; ce RFC sur les échanges de clé hybrides dans TLS 1.3 (RFC 9846) ne va pas vous dire quel algorithme post-quantique utiliser, et il ne va pas parler d'authentification (vu la façon dont fonctionne TLS, le risque d'attaque rétroactive n'existe pas, donc on a le temps de voir venir). Par contre, il va dire comment faire que les deux parties aient la clé (clé pour la cryptographie symétrique, celle qui va chiffrer la communication une fois la session TLS établie) avec une configuration hybride, PQ/T. Parmi ses objectifs, il y a évidemment la compatibilité avec le TLS existant (un client récent doit interopérer avec un serveur ancien et réciproquement, sans compter les exaspérantes "*middleboxes*" entre les deux), et de préférence sans exiger des aller-retours supplémentaires, car il ne faut pas dégrader la latence. Il faut aussi être rapide (ne pas imposer des calculs démesurés, voir « "*Benchmarking Post-Quantum Cryptography in TLS*" <https://eprint.iacr.org/2019/1447.pdf> » et « "*Post-quantum confidentiality for TLS*" <https://www.imperialviolet.org/2018/04/11/pqconftls.html> »). Et il ne faut pas trop augmenter la taille des paquets : bien sûr, TLS fontionne sur TCP et n'a pas les problèmes que la fragmentation des paquets pose à UDP mais quand même. Or, les algorithmes post-quantiques ont souvent des clés et des signatures bien plus grosses qu'avec les algorithmes traditionnels.

Plus spécifiquement qu'un général échange de clés <https://www.bortzmeyer.org/echange-cles.html>, ce RFC parle de KEM ("*Key Encapsulation Mechanism*"). Si vous ne vous souvenez pas bien de ce qu'est un KEM, la section 2 du RFC vous le résume. Un KEM est utilisé lorsque la clé est générée par une seule des deux parties, et communiquée à l'autre. Le but d'un KEM est que les deux parties qui communiquent utilisent la même clé secrète pour le chiffrement en cryptographie symétrique. Un KEM permet se communiquer de manière sécurisée même face à un attaquant actif qui peut modifier les paquets à sa guise (au pire, il fera un déni de service mais ne pourra jamais lire les messages).

Maintenant, les détails pratiques, en section 3. Le nom du mécanisme hybride va être annoncé au début de la session TLS (dans l'extension `supported_groups` et rappelez-vous que ce ne sont pas forcément des groupes). Le nom, même si on y retrouve les noms des deux algorithmes (le traditionnel et le post-quantique) désigne le mécanisme hybride, quand on fera du post-quantique pur, il y aura un autre nom. Un exemple de nom (RFC pas encore publié) est `X25519MLKEM768` (le traditionnel X25519 et le post-quantique ML-KEM). Ensuite, on fait simple : on fait un double échange de clefs (un traditionnel et un post-quantique) et on dérive ensuite les clefs de session à partir des deux. Plus précisément, on concatène les messages traités avec chacun des deux algorithmes du mécanisme hybride. Même chose

pour le secret généré par chacun des deux algorithmes. La concaténation de ces secrets sera le point d'entrée du calcul de la clé.

La sécurité du système (section 6) vient du fait qu'un attaquant devrait réussir à casser les deux algorithmes : il lui faudrait un CRQC **et** une attaque (aujourd'hui inconnue) contre l'algorithme quantique. C'est l'avantage de ces systèmes hybrides. Mais si vous voulez creuser la cryptographie qui est derrière ces mécanismes hybrides, le RFC recommande « "*KEM Combiners*" <https://link.springer.com/chapter/10.1007/978-3-319-76578-5_7> » et « "*Hybrid Key Encapsulation Mechanisms and Authenticated Key Exchange*" <https://link.springer.com/chapter/10.1007/978-3-030-25510-7_12> ». Et si vous voulez en apprendre plus sur le calcul quantique, l'annexe A du RFC cite quelques bonnes lectures.

La section 4 discute quelques problèmes avec cette approche (ou, parfois, avec la cryptographie post-quantique en général). Par exemple, les clés sont grandes. Bon, qu'on fasse de l'hybride ou du post-quantique pur, ça ne change pas grand'chose, c'est la clé post-quantique qui est responsable de la grande majorité des octets : Classic McEliece a des clés d'au moins 200 kilo-octets ! Une telle taille se heurterait à des limites de TLS (65 536 octets pour la clé publique) mais heureusement les algorithmes post-quantiques normalisés par le NIST <https://www.bortzmeyer.org/nist-pq.html> ont des clés plus petites, et qui tiennent dans les messages TLS (mais pas forcément dans un seul paquet IP). L'encodage de notre RFC aggrave les choses si on annonce deux hybrides, chacun utilisant la même clé post-quantique, qui sera alors envoyée deux fois.

Les noms des mécanismes hybrides seront mis dans le registre IANA <https://www.iana.org/assignments/tls-parameters/tls-parameters.xml#tls-parameters-8> (X25519MLKEM768 et quelques autres y sont déjà) au fur et à mesure de leur spécification.

Côte mises en œuvre, les expérimentations sont anciennes : CECPQ2 <https://www.imperialviolet.org/2018/12/12/cecpq2.html> ou OQS <https://github.com/open-quantum-safe/openssl/tree/OQS-OpenSSL_1_1_1-stable>, par exemple, et des tests de performance ont été faits <https://link.springer.com/chapter/10.1007/978-3-030-44223-1_5>. Aujourd'hui, le mécanisme hybride est possible avec Chrome, Firefox, OpenSSL, wolfSSL, Cloudflare, Google, BoringSSL, Rustls <https://github.com/rustls/rustls>…(Je n'ai pas testé Firefox.)

La marche vers ce RFC a été longue. Les premiers brouillons sur un mécanisme hybride pour TLS, draft-schanck-tls-additional-keyshare, draft-kiefer-tls-ecdhe-sidh et draft-whyte-qsh-tls13 datent de 2017-2018. Et cet article que vous lisez a été commencé en 2020, me dit mon VCS. Il y a d'ailleurs eu quelques débats vigoureux <https://keymaterial.net/2025/11/27/ml-kem-mythbusting/>.