RFC 9958 : Post-Quantum Cryptography for Engineers

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 20 juin 2026

Date de publication du RFC : Juin 2026

https://www.bortzmeyer.org/9958.html

Tout le monde parle <https://www.afnic.fr/observatoire-ressources/papier-expert/en-route-vers-la-journee-du-conseil-scientifique-2025/> des calculateurs quantiques et du risque qu'ils font peser sur la cryptographie traditionnelle. La solution ? Remplacer ces algorithmes par des algorithmes post-quantiques, c'est-à-dire résistant aux calculateurs quantiques. Ces algorithmes existent déjà, plusieurs sont normalisés et beaucoup ont déjà été mis en œuvre dans des programmes. Mais les intégrer dans les protocoles de l'Internet n'est pas trivial. Ce RFC vise à guider les ingénieur·es qui vont appliquer ces algorithmes post-quantiques à des logiciels et des protocoles du monde de l'Internet.

Il ne s'agit pas ici d'expliquer le fonctionnement des algorithmes de cryptographie post-quantiques. Le RFC les considère acquis et se demande comment on va les utiliser. (Note personnelle : de toute façon, je ne comprends rien à la cryptographie, post-quantique ou pas. Mais ce n'est pas forcément grave, ce RFC est conçu pour des ingénieur·es normaux·les, le doctorat en cryptographie n'est pas indispensable.) Notez aussi, question terminologie, que « post-quantique » ne fait pas l'unanimité. On pourrait dire (section 2 du RFC), « après quantique » ou bien « résistant au quantique » ou encore « prêt pour le quantique ». Chaque terme a ses qualités et ses défauts : « résistant au quantique » serait inapproprié si, finalement, on trouvait un algorithme quantique pour casser un algorithme qui se prétendait sûr face aux calculateurs quantiques.

D'abord, résumons le problème : la quantique est aujourd'hui un domaine très bien connu et dont les bases théoriques sont solides et établies. Un calculateur quantique peut effectuer **certains** calculs plus vite qu'un ordinateur classique. Parmi ces calculs, il y a les problèmes mathématiques sous-jacents aux algorithmes de cryptographie courants, comme RSA et ECDSA. Aujourd'hui, certes, les calculateurs quantiques sont très loin de pouvoir être utilisés en pratique contre la cryptographie. Il faudrait des millions de qubits physiques ou des milliers de qubits logiques (ceux qui ont un système de correction d'erreurs). On n'en est qu'à quelques centaines de qubits physiques (malgré certaines annonces sensationnalistes), et la difficulté de fabrication et de fonctionnement augmente très vite avec le nombre de qubits. Mais, à force de progrès, on aura peut-être un jour des CRQC ("*Cryptographically Relevant*

Quantum Computer" <https://en.wiktionary.org/wiki/CRQC>, prononcez « cric » <https://mastodon.gougere.fr/@bortzmeyer/114008857061964694>, cf. section 2 du RFC), des calculateurs quantiques utilisables dans le monde réel.

Ce jour-là, on pourra, par exemple, trouver une clé privée RSA ou ECDSA à partir de la clé publique, cassant ainsi effectivement ces algorithmes. Et, comme le déploiement des algorithmes post-quantiques va prendre du temps, il ne faut pas attendre que des CRQC soient disponibles pour commencer.

On pourrait croire que le problème, pour les protocoles IETF, est simple : tous ont la propriété d'agilité cryptographique (RFC 7696 [1]), qui permet de remplacer un algorithme par un autre sans changer le protocole et le code. Mais ça serait trop simple : les algorithmes post-quantiques ont des propriétés qui font qu'il n'est pas toujours possible de les utiliser tels quels, sans adaptation du protocole. Par exemple, leurs clés et signatures sont souvent bien plus grandes, et ne rentrent pas dans les protocoles existants.

Donc, quelques points à garder en tête (j'ai déjà lu des erreurs sur ces sujets, dans divers articles) :

— Un algorithme post-quantique n'a rien de quantique, et ne nécessite pas de calculateur quantique pour tourner. Nous travaillons sur la défense (la cryptographie post-quantique), pas sur l'attaque (les calculateurs quantiques). Pas besoin donc de s'y connaitre en quantique pour comprendre ces algorithmes.
— Les algorithmes « quantiques » ne sont pas 100 % quantiques. Ils ont tous une partie classique et un calculateur quantique ne sera donc pas utilisé seul, mais comme une partie d'un ordinateur mixte (section 3 du RFC).
— Un calculateur quantique n'est pas un ordinateur : il n'est rapide que pour certains problèmes, et c'est cela qui permet de concevoir des algorithmes dont on pense qu'ils ne seront pas vulnérables aux futurs CRQC. (Le RFC fait une analogie avec les GPU qui ne sont pas plus rapides que les CPU dans l'absolu, seulement plus rapides pour certains problèmes.)
— Un algorithme post-quantique peut, si on trouve une faille dans sa conception ou dans sa mise en œuvre, être cassé par un ordinateur classique. C'est d'autant plus gênant que ces algorithmes sont plus récents que RSA ou ECDSA, et ont été moins testés au feu.

Donc (section 1 du RFC), aujourd'hui, des gens construisent des calculateurs quantiques et ceux-ci sont de plus en plus perfectionnés, capable d'attaquer des problèmes plus gros. Pas mal d'argent et d'efforts sont investis <https://www.bortzmeyer.org/rapport-forteza-quantique.html> dans ces travaux. Mais on reste loin du CRQC ("*Cryptographically Relevant Quantum Computer*"), les calculateurs connus du public ne peuvent casser que des clés RSA ou ECDSA ridiculement courtes <https://sam-jaques.appspot.com/quantum_landscape_2024>. Il est très difficile de prévoir quand on aura un CRQC. (Lorsque j'ai fait mon exposé à Pas Sage En Seine <https://www.bortzmeyer.org/pas-sage-en-seine-quantique.html>, plusieurs personnes m'ont dit que les CRQC seraient disponibles « bientôt ». C'était en 2018 et on ne les a toujours pas.) Le problème est difficile et il ne suffit pas d'extrapoler les prototypes actuels, le monde quantique est difficile et, plus on rassemble de composants, plus il est difficile de maintenir les propriétés quantiques qui nous intéressent. Des prédictions sensationnalistes sur l'arrivée « prochaine » de calculateurs quantiques ont déjà été faites et se sont montrées fausses. Ceci dit, il faut noter qu'il n'y a pas que la recherche publique : on peut toujours imaginer que, dans la zone 51, la NSA fait tourner un CRQC utilisant des technologies obtenues auprès des extra-terrestres…

Mais le fait que les CRQC n'arrivent pas n'est pas une raison pour attendre : le déploiement effectif des algorithmes post-quantiques va prendre beaucoup de temps (il s'agit d'une migration très complexe) et il ne faut donc pas rester les bras croisés en attendant qu'un CRQC soit réellement disponible (d'autant plus qu'il existe toujours la possibilité de percées soudaines dans la recherche scientifique). Mais, en raison de l'incertitude, la question est délicate (cf. mon exposé (en ligne sur https:

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc7696.txt

//www.bortzmeyer.org/files/post-quantique-decision.pdf) à NetGouv 26 <https://netgouv.hypotheses.org/1551>).

Euclide concevait des algorithmes bien avant qu'on ait des ordinateurs et Lovelace écrivait des programmes alors que l'ordinateur à qui ils étaient destinés n'existait pas (et n'a finalement pas existé). De même, des chercheurs écrivent des algorithmes pour les calculateurs quantiques sans avoir de calculateurs quantiques. C'est notamment le cas de l'algorithme de Shor, qui résout le problème de la décomposition en facteurs premiers qui est à la base de RSA. Et une variante de l'algorithme résout le problème du logarithme discret qui est à la base des algorithmes à courbes elliptiques comme ECDSA (RFC 6090).

Il existe aussi un algorithme, l'algorithme de Grover, qui permet de s'attaquer aux clés utilisées en cryptographie symétrique mais il est loin de l'efficacité spectaculaire de l'algorithme de Shor et notre RFC estime donc (section 3.1 pour les détails) que les éventuels futurs calculateurs quantiques n'auront que peu d'impact sur la cryptographie symétrique, notamment parce que cet algorithme n'est pas facilement parallélisable. Et il y a des raisons de penser <https://arxiv.org/abs/quant-ph/9711070> que Grover ne sera sans doute pas amélioré. Un algorithme comme AES-128 est donc a priori sûr (et c'est encore plus vrai pour AES-256 <https://cyber.gouv.fr/sites/default/files/document/follow_up_position_paper_on_post_quantum_cryptography.pdf>). Le RFC se concentre donc sur l'asymétrique.

Pour la cryptographie asymétrique, la situation est moins rose (section 3.2). Des algorithmes très courants comme RSA et les algorithmes à courbes elliptiques, mais aussi des algorithmes moins connus comme ElGamal (RFC 6090) ou Schnorr (RFC 8235) seront cassables dès qu'on aura un CRQC. En quelques heures, voire en quelques secondes, celui-ci pourra casser une clé RSA de 2 048 bits, comme étudié dans « "*Circuit for Shor[Caractère Unicode non montré [2]]s algorithm using 2n+3 qubits*" <https://arxiv.org/pdf/quant-ph/0205095.pdf> », « "*How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits*" <https://arxiv.org/abs/1905.09749> » ou bien « "*Breaking RSA Encryption - an Update on the State-of-the-Art*" <https://www.quintessencelabs.com/blog/breaking-rsa-encryption
(Rappel : tous ces articles supposent l'existence d'un CRQC, et on n'en a pas encore.) Bon, après, vous pouvez toujours faire des clés RSA d'un téra-octet <https://cr.yp.to/papers/pqrsa-20170419.pdf> mais ça ne serait pas pratique.

La section 4 du RFC détaille les services pour lesquels on utilise de la cryptographie asymétrique :

— Transport de clé symétrique. C'était la façon traditionnelle de chiffrer. Pour des raisons de performance, on ne chiffre pas avec les algorithmes de cryptographie asymétrique, qui sont trop lents. Une des parties génère une clé de cryptographie symétrique, qui doit donc être connue des deux parties, et on la chiffre avec l'algorithme de cryptographie asymétrique avant de l'envoyer à l'autre partie. Cette méthode ne fournit pas de confidentialité persistante donc on ne l'utilise plus beaucoup.
— Accord sur une clé. C'est la façon la plus courante aujourd'hui pour que les deux parties aient la même clé de chiffrement symétrique. Un exemple pour réaliser cet accord est la méthode Diffie-Hellman.
— Signature de documents et authentification d'une autre partie.

Les algorithmes courants utilisés en cryptographie asymétrique sont vulnérables aux futurs éventuels CRQC. Ce problème étant connu depuis longtemps, plusieurs algorithmes pour lesquels on ne connait pas d'algorithme quantique susceptible de les casser ont été développés. On peut donc les qualifier de « post-quantiques ». Suite à un concours lancé en 2016 <https://www.bortzmeyer.org/nist-pq.html>, le NIST a normalisé plusieurs de ces algorithmes <https://www.nist.gov/news-events/news/2024/08/nist-releases-first-3-finalized-post-quantum-encryption-standards>

2. Car trop difficile à faire afficher par LaTeX

(et d'autres le seront dans le futur <https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization>). Mais, en général, ces algorithmes ne peuvent pas tout simplement remplacer les algorithmes traditionnels. Par exemple, RSA et ECDSA peuvent servir de KEM ("*Key Encapsulation Mechanism*", cf. section 9) **et** faire des signatures alors que les algorithmes post-quantiques existants ne savent faire qu'un des deux. Et puis ils ne s'utilisent pas de la même façon et la transition vers ces algorithmes sera donc plus complexe que l'a été celle depuis RSA vers ECDSA.

Quels sont ces algorithmes normalisés par le NIST (section 5) ? Pour le KEM, ce sont :

— ML-KEM (autrefois nommé Kyber), normalisé dans la norme FIPS-203 <https://csrc.nist.gov/pubs/fips/203/final>.
— HQC, approuvé mais pas encore normalisé.

Et pour les signatures :

— ML-DSA (autrefois nommé Dilithium), normalisé dans FIPS-204 <https://csrc.nist.gov/pubs/fips/204/final>. (Voir aussi la section 10.2.)
— SLH-DSA (autrefois nommé SPHINCS+), normalisé dans FIPS-205 <https://csrc.nist.gov/pubs/fips/205/final>.
— FN-DSA, qui sera apparemment normalisé dans FIPS-206.

D'autres algorithmes sont en cours de normalisation à l'ISO (section 6) :

— FrodoKEM.
— ClassicMcEliece <https://classic.mceliece.org/>.
— NTRU.

OK, on a plein d'algorithmes résistants aux calculateurs quantiques. Maintenant, de quel délai dispose t-on pour faire la transition, avant que les CRQC ne fichent toute la cryptographie traditionnelle en l'air ? La section 7 de notre RFC explore cette difficile question. Je divulgâche tout de suite : on ne sait pas. Le progrès des calculateurs quantiques ne dépend pas de progrès d'ingénierie relativement prévisibles mais de percées scientifiques encore inconnues. Mais d'un autre côté, attendre n'est pas non plus une bonne solution. Il y a sans doute des attaquants qui enregistrent des messages aujourd'hui, dans l'espoir qu'un CRQC permettra de les décrypter un jour (ce qu'on nomme HNDL pour "*Harvest Now, Decrypt Later*"). Si certains secrets sont à courte durée de vie, d'autres peuvent être encore sensibles dans des dizaines d'années (la France ne publie toujours pas complètement les archives étatiques sur la guerre d'Algérie). Pour de tels secrets, il serait peut-être plus prudent de passer au post-quantique tout de suite.

Pour les signatures, le RFC prend l'exemple de l'authentification : les signatures utilisées dans l'établissement d'une connexion TLS ne sont sensibles que pendant le très court temps entre leur génération et leur vérification. Par contre, la signature du microcode d'un objet connecté peut rester critique pendant toute la durée de vie de l'objet.

Pour analyser le délai dont on dispose, on utilise souvent le modèle de Mosca <https://globalriskinstitute.org/publications/quantum-threat-timeline-report-2020/>. Dans ce modèle, X est le temps pendant lequel les secrets doivent rester secrets (rappelez-vous que, pour certains secrets, cela peut être de nombreuses années), Y est la durée de la transition (rappelez-vous qu'en informatique, les transitions complètes prennent **toujours beaucoup** plus de temps que prévu, la publication par le NIST d'une norme ne suffit pas) et Z est le temps qu'il nous reste avant que les CRQC soient disponibles (c'est la durée la plus incertaine des trois, d'autant plus qu'on ne peut pas exclure la possibilité de progrès soudains). Si Z ¿ X + Y, tout va bien. Sinon, on a un trou pendant lequel certains usages vont être vulnérables. Notons aussi que Z va être plus court pour les États (qui ont sans doute quelques années d'avance sur la recherche publique en matière de calculateurs quantiques) que pour le pirate de base, qui doit attendre qu'un CRQC soit en vente sur AliExpress.

Quant à Y, son estimation varie beaucoup. S'il faut juste faire une mise à jour d'un logiciel, cela peut être rapide. Mais de la cryptographie est aussi présente dans bien des matériels, qu'on ne remplace pas du jour au lendemain, comme les puces accélératrices des opérations de cryptographie ou les HSM. Et certaines clés doivent rester stables pendant longtemps comme la clé DNSSEC de la racine du DNS.

Et il y a aussi le temps de programmer, de tester, de faire valider, de déployer, etc. Bref, entre la publication d'une norme comme FIPS-203 <https://csrc.nist.gov/pubs/fips/203/final> et sa généralisation, il faut compter de nombreuses années. C'est une des raisons qui justifie que le travail de transition vers la cryptographie post-quantique n'ait pas attendu la sortie du premier CRQC.

Si vous voulez apprendre un peu de cryptographie, vous pouvez aussi lire la section 8 du RFC, qui expose les principales catégories d'algorithmes post-quantiques :

— Réseaux euclidiens (non, ne me demandez pas d'expliquer) : c'est la technique sous-jacente à ML-KEM, ML-DSA et FrodoKEM, donc la technique « principale » à l'heure actuelle. Clés et surtout signatures sont bien plus grandes qu'avec ECDSA ou même RSA mais quand même plus courtes qu'avec les autres catégories. (La cryptographie post-quantique n'est pas écologique.)
— Condensation : c'est la technique derrière SLH-DSA. Il y a même des algorithmes spécifiés dans un RFC, les RFC 8391 et RFC 8554. Attention : bien des algorithmes de cette catégorie sont à état (il ne faut pas signer deux fois avec le même état) et sont donc difficiles à utiliser de manière sûre (et pas du tout adapté aux protocoles comme DNSSEC, qui sont habitués à un environnement sans état).
— Codes correcteurs : c'est là qu'on trouve HQC et ClassicMcEliece.

La façon la plus courante d'utiliser la cryptographie sur l'Internet est de combiner de la cryptographie symétrique, avec une clé générée pour chaque utilisation, et de la cryptographie asymétrique, qui sert à authentifier et à faire en sorte que la clé de cryptographie symétrique soit partagée par les deux parties (cf. RFC 9180 mais attention au fait que le terme « hybride » peut désigner autre chose en cryptographie post-quantique). La façon la plus simple (mais pas la plus sûre !) de réaliser ce partage est qu'une des deux parties génère la « clé de session », celle utilisée pour la cryptographie symétrique, puis la chiffre en cryptographie asymétrique avec la clé publique de l'autre partie. (On combine cryptographie symétrique et asymétrique, alors qu'on pourrait assurer le même service uniquement avec l'asymétrique, car, si l'asymétrique est souple, la symétrique est beaucoup plus rapide.) Ce transport de la clé d'une partie à l'autre est une des façons de faire en sorte que les deux parties aient la même clé, mais il en existe d'autres (ne me demandez pas un exposé détaillé, et encore moins un avis !). Et ces différentes façons ont typiquement des API différentes, ce qui complique le remplacement d'une façon par une autre. La section 9, consacrée aux KEM, détaille cette question. Après l'avoir lue, vous comprendrez les concepts d'AKE et de NIKE [Caractère Unicode non montré].

La section 10 du RFC, consacrée aux signatures, intéressera entre autres les gens qui font du DNSSEC, comme moi. Un point amusant (section 10.4) et qui illustre bien qu'on ne peut pas toujours remplacer purement et simplement un algorithme traditionnel par un post-quantique : ML-DSA incorpore dans son calcul la condensation de la donnée à signer, ce qui le rend plus résistant à certaines attaques. Les protocoles qui calculent un condensat eux-mêmes avant de signer (comme DNSSEC) ne bénéficient donc pas de cette sécurité accrue et, idéalement, devraient être modifiés.

Un autre exemple du fait qu'on ne peut pas brancher un algorithme post-quantique et espérer que tout soit pareil qu'avant est le problème des performances. Ce n'est pas juste que les algorithmes post-quantiques sont plus lents, c'est qu'ils sont parfois tellement lents et avec des clés tellement grosses qu'il faut changer sa façon de travailler. Les tableaux de la section 12 donnent une idée du problème. Alors que des clés traditionnelles font, par exemple, 65 et 32 octets (pour la clé publique et la clé privée), les plus petites clés ML-KEM font respectivement 800 et 1632 octets (et c'est pire pour ML-DSA). Pour des protocoles comme TLS, qui fonctionne sur des connexions TCP ou QUIC, ce n'est pas très grave, OK, ce n'est pas écologique, mais l'augmentation de données due aux clés n'est probablement qu'une faible partie de ce que transportera la connexion. À part avec des "*middleboxes*" boguéees qui auraient une limite stupide à la taille des paquets TLS, ça devrait marcher (mais elles ont d'autres problèmes <https://github.com/golang/go/issues/79626>). Mais pour des protocoles comme IKE ou le DNS, qui tournent sur UDP et ont des limites bien plus strictes, c'est ennuyeux <https://labs.ripe.net/author/eline-stehouwer/dnssec-and-pqc-practical-impact-of-increased-tcp-in-dns/>. Et le même problème se pose pour les objets contraints et leurs protocoles spécifiques (cf. section 14, qui

leur est dédiée). Il existe bien des solutions (cf. RFC 9242 et RFC 9370 pour IKE) mais qui compliquent et ralentissent le protocole.

Autre problème, la sécurité des algorithmes post-quantiques face aux ordinateurs **traditionnels**. C'est bien joli, de résister aux CRQC mais, si l'algorithme peut être cassé par de la cryptanalyse traditionnelle, cela ne sert à rien (cf. sections 15.1 et 15.4 du RFC). Lors de la dernière grande transition cryptographique sur l'Internet, celle depuis RSA vers ECDSA, il y avait de très bonnes raisons de penser que le nouvel algorithme était plus sûr et qu'on pouvait donc simplement remplacer l'ancien par le nouveau. Ce n'est pas le cas avec les algorithmes post-quantiques <https://keymaterial.net/2025/12/13/a-very-unscientific-guide-to-the-security-of-various-pqc-algorithms/>, dont on ne peut pas être certains de leur résistance (regardez par exemple l'attaque KyberSide, contre ML-KEM <https://eprint.iacr.org/2022/1452>). C'est d'autant plus vrai que les programmes qui les mettent en œuvre sont, eux aussi, récents et pas forcément testés longuement au feu. Et, comme les algorithmes traditionnels sont vulnérables aux CRQC, que faire ? Que choisir ? L'idée dominante aujourd'hui est d'avoir les deux, un système **hybride** (ou « PQ/T », pour « Post-quantique et Traditionnel », cf. RFC 9794). Par exemple, lors d'un échange de clé symétrique, on va créer la clé en concaténant deux clés qui ont été obtenues, l'une par un algorithme traditionnel, l'autre par un post-quantique (c'est ce que fait le futur RFC draft-ietf-tls-hybrid-design). Ainsi, la défaillance d'un des deux algorithmes ne permettra pas à l'attaquant d'obtenir la clé. On peut aussi appliquer la fonction de dérivation autant de fois qu'on a de clés dans la structure hybride (RFC 9370). Pour l'authentification, on peut authentifier avec les deux algorithmes et exiger que les deux donnent le même résultat (draft-ietf-lamps-pq-composite-sigs).

Pour faciliter la tâche des programmeur·ses, il vaut mieux que les deux clés (la PQ et la T) soient manipulées ensemble, dans une seule structure de données ("*composite*"). On évitera ainsi de compliquer le protocole comme cela serait le cas si, par exemple, on exigeait qu'il y ait deux certificats. Un travail de normalisation va être nécessaire pour le format de ces clés composées (cf. draft-bonnell-lamps-chameleon-cer
Il faudra aussi définir quelles paires PQ/T ont du sens, question sécurité, et ne pas juste accepter n'importe quel algorithme post-quantique avec n'importe quel algorithme traditionnel.

Voilà, maintenant, si vous voulez sérieusement apprendre la cryptographie post-quantique, le RFC recommande le livre « "*Serious Cryptography*" <https://nostarch.com/serious-cryptography-2nd-editi
deuxième édition), de Jean-Philippe Aumasson. Si vous voulez voir du code post-quantique, il y a le projet OQS <https://openquantumsafe.org/> (avec une très bonne FAQ pour les débutants <https://openquantumsafe.org/faq.html> et un dépôt Github <https://github.com/open-quantum-
>). Et si vous vous demandez ce que fait l'IETF en matière de normalisation de la cryptographie post-quantique dans ses protocoles, il y a une liste maintenue à jour <https://github.com/ietf-wg-pquip/state-of-protocols-and-pqc>. Je rajoute « "*awesome-post-quantum*" <https://github.com/veorq/awesome-post-quantum> », qui maintient une liste de ressources sur la cryptographie post-quantique, Cloudflare avait fait un article détaillé sur les finalistes du concours NIST <https://blog.cloudflare.com/nist-post-quantum-surprise/>, qui expliquait bien les différentes questions techniques, et enfin lisez le texte de position <https://messervices.cyber.gouv.fr/guides/en-follow-position-paper-post-quantum-cryptography> de l'ANSSI.

Bon, mais je vais quand même vous montrer des exemples d'utilisation d'algorithmes post-quantiques dans des protocoles Internet. D'abord, avec SSH :

```
% ssh -v autre-machine-debian
...
debug1: kex: algorithm: mlkem768x25519-sha256
```

Oui, OpenSSH sait faire du ML-KEM pour l'échange de clés, depuis la version 10 <https://www.openssh.com/txt/release-10.0>. Celui-ci ne permet pas de faire de l'authentification donc OpenSSH ne sait apparemment pas générer des clés de machines avec des algorithmes post-quantiques.

Pour TLS, voici un échange de clés <https://www.bortzmeyer.org/echange-cles.html> hybride (ML-KEM et traditionnel) vu par Firefox (« Outils de développement Web » puis « Réseau » puis « Sécurité », tout au bout) : .

Pour DNSSEC, il existe une zone d'exemple <https://pq-dnssec.dedyn.io/> expérimentale signée avec ML-DSA :

```
% dig +dnssec dilithium2.pdns.pq-dnssec.dedyn.io DNSKEY
;; Truncated, retrying in TCP mode.
...
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 55771
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1
...
;; ANSWER SECTION:
dilithium2.pdns.pq-dnssec.dedyn.io. 3598 IN DNSKEY 257 3 18 (
    XVrPO18btCpMLjXFiZYJNyMbQLXB7oOwk6ZXEmxhm8PP
    EKiKP1+t/j7pwBUNYXp3s0U+3AFp3moviOf4cnl4K4MH
                                ...
    ) ; KSK; alg = 18 ; key id = 47389
dilithium2.pdns.pq-dnssec.dedyn.io. 3598 IN RRSIG DNSKEY 18 5 3600 (
    20260129000000 20260108000000 47389 dilithium2.pdns.pq-dnssec.dedyn.io.
    aGU3rLJ8vb1AYln+y3bxcOO0RHboWfh03i8H1XhLd9f/
    TxhGJNwsTCoRGOnJPjdB2ZDHMB7EnfUfgOitDjvcvcsd
                                ...
;; Query time: 0 msec
;; SERVER: 127.0.0.1#53(127.0.0.1) (TCP)
;; WHEN: Tue Jan 20 15:16:45 CET 2026
;; MSG SIZE  rcvd: 3877
```

Notez la taille de la réponse, et la nécessité de se rabattre sur TCP. D'autre part, l'algorithme de numéro 18, utilisé ici, n'est pas enregistré <https://www.iana.org/assignments/dns-sec-alg-numbers/dns-sec-alg-numbers.xml#dns-sec-alg-numbers-1>. Aujourd'hui, le monde du DNS semble plutôt attentiste, il n'y a pas d'algorithme de signature post-quantique qui convienne vraiment au DNS. Une synthèse <https://www.icann.org/octo-031-en.pdf> de l'ICANN est relativement sceptique.

Et merci à Magali Bardet pour sa relecture attentive. (Et, naturellement, les erreurs restantes en crypto sont de moi, pas d'elle.)