RFC 7165 : Use Cases and Requirements for JSON Object Signing and Encryption (JOSE)

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 15 avril 2014

Date de publication du RFC : Avril 2014

https://www.bortzmeyer.org/7165.html

Il est traditionnel de classer les mécanismes de sécurité de l'Internet en deux, ceux fondés sur la **sécurité du canal** et ceux fondés sur la **sécurité de l'objet**. Pour les premiers, on garantit certaines propriétés de l'objet pendant son transit dans un canal donné. Pour les seconds, l'objet se déplace en permanence dans une armure cryptographique qui le protège, quelle que soit la sécurité du canal où il voyage. Le groupe de travail JOSE <https://tools.ietf.org/wg/jose> de l'IETF cherche à procurer une sécurité de l'objet aux textes encodés en JSON. Son premier RFC est consacré à décrire les cas d'usage et le cahier des charges de cette sécurité.

JSON (RFC 7159 [1]) est un format très répandu, notamment sur le Web. Il peut être transporté dans des protocoles qui assurent la sécurité du canal comme HTTPS. Mais, comme toujours avec la sécurité du canal (IPsec, TLS, etc), elle ne garantit pas de sécurité de bout en bout. Par exemple, si le serveur d'origine est piraté, et les textes en JSON modifiés, HTTPS ne protégera plus. Même chose si on utilise certains relais, chose courante sur le Web : HTTPS ne protège pas contre les intermédiaires. Il existe à l'IETF des mécanismes assurant la sécurité du message (de l'objet) comme CMS (RFC 5652) pour l'ASN.1/DER. Mais pour le JSON ? Il y a là un manque à combler. Car la sécurité du message est nécessaire pour permettre la manipulation du messages par des intermédiaires à qui on ne fait pas forcément confiance. Pour le courrier électronique, la sécurité du canal est fournie par SMTP sur TLS - RFC 3207, et la sécurité du message par PGP - RFC 4880 - ou S/MIME - mais le RFC, curieusement, ne cite pas PGP. Par contre, il n'existe pas encore de solution propre pour JSON transporté sur HTTP. Les gens de XML ont choisi XML Signature et XML Encryption, les gens de JSON travaillent donc activement à leur propre solution.

Ce RFC parle de sécurité et utilise donc largement la terminologie standard du RFC 4949. Il peut être utile aussi de lire le RFC 5652 sur CMS car ce RFC le cite souvent.

Le groupe de travail JOSE ("*JSON Object Signing and Encryption*") est chargé de développer trois formats :

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc7159.txt

— Un format assurant la confidentialité du texte JSON par chiffrement, JWE ("*JSON Web Encryption*"),
— Un format assurant l'intégrité et l'authenticité du texte JSON par signature numérique, JWS ("*JSON Web Signature*"),
— Un format pour représenter les clés cryptographiques utilisées, JWK ("*JSON Web Key*").

Rappelez-vous que ce RFC 7165 n'est que le premier RFC du groupe, il ne donne pas encore de solutions, il explore le problème et précise les exigences du travail à effectuer (la liste officielle et numérotée de ces exigences figure en section 6).

Le cahier des charges débute en section 3 avec un résumé des exigences « de base » :

— Le format pour l'objet chiffré doit permettre l'utilisation de la cryptographie symétrique **et** de la cryptographie asymétrique.
— Le format pour l'objet signé doit permettre une vérification de l'intégrité par MAC pour le cas où les deux parties partagent une clé (cryptographie symétrique), et par signature avec de la cryptographie asymétrique.
— Comme les textes JSON protégés ne seront pas forcément traités dans le contexte d'une communication synchrone avec possibité de négociation ou d'échange, il faut (le RFC prévoit des exceptions très encadrées) que tous les paramètres cryptographiques nécessaires (à part les clés) soient inclus dans l'objet. À noter que la liste des paramètres dépend de l'algorithme cryptographique utilisé (qui doit donc être indiqué clairement).
— Conséquence de l'exigence précédente, les applications qui traitent les textes JSON protégés doivent avoir un moyen simple de déterminer si elles sont en possession de la clé nécessaire, et de produire un message d'erreur clair si ce n'est pas le cas.

La section 4 rappelle que l'application qui utilisera les textes JSON protégés a aussi des responsabilités, par exemple c'est elle qui décidera quels algorithmes sont acceptables, comment seront échangées les clés, etc.

La liste complète des exigences est en section 6 du RFC. Mais, avant cela, la section 5 contient les études de cas. La première est celle des jetons ("*security tokens*"), ces petits textes qu'on s'échange pour authentifier ou autoriser un tiers dans une communication (« il a le doit de lire les fichiers, laisse-le y accéder »). C'est par exemple le cas des assertions de SAML, de Persona, et d'OpenID Connect. Certains utilisent XML mais un autre groupe de travail IETF développe un format JSON, JWT ("*JSON Web token*"), déjà utilisé dans des systèmes comme Persona. Ce format a besoin des techniques JOSE pour être sécurisé. Le RFC note qu'il faudra aussi que le format final puisse être mis dans un URL, ce qui nécessite qu'il doit très compact. Le fait qu'une permission soit donnée peut être parfois confidentiel et le format de signature peut donc ne pas suffire. Même dans ce cas d'usage, il faudra donc aussi penser à la confidentialité. Autre exemple de jeton, dans OAuth (RFC 6749). Ici, la norme impose un transport par HTTPS donc la confidentialité est normalement assurée, il ne reste que l'authentification.

Ces jetons de sécurité sont également utilisés dans les cas de fédérations d'identité. Ainsi, OpenID Connect, déjà cité, repose sur OAuth et JSON et est un des gros demandeurs des fonctions JOSE.

Autre étude de cas, pour le protocole XMPP (RFC 6120), surtout connu pour la messagerie instantanée. Un instant, vous allez me dire, le X dans XMPP rappelle qu'il a XML comme format. Que vient-il faire dans un RFC parlant de JSON ? C'est parce que la sécurité de XMPP, aujourd'hui, est uniquement une sécurité du canal : les communications entre clients et serveurs, ou bien entre les serveurs, sont protégées par TLS (et de nombreux acteurs du monde XMPP se sont engagés à ce que le chiffrement soit systématisé `<https://github.com/stpeter/manifesto>` avant mai 2014). Mais cela ne fournit pas de sécurité de bout en bout, ce qui est d'autant plus gênant que les sessions XMPP sont rarement établies directement entre deux pairs, mais passent par des serveurs qui sont la plupart du temps gérés par une organisation différente. XMPP, tel qu'utilisé aujourd'hui, est donc vulnérable à l'espionnage par le fournisseur. Le problème est identifié depuis longtemps, mais la seule solution standard, décrite

dans le RFC 3923, n'a jamais été adoptée (comme la plupart des techniques S/MIME). Actuellement, la solution la plus commune pour résoudre ce problème de sécurité est OTR.

Une autre solution est donc en cours de développement dans la communauté XMPP, fondée sur JSON et JOSE. C'est certes bizarre de transporter des textes JSON dans les flux XML de XMPP et cela complique un peu les choses (il faut être sûr que le texte JSON ne contienne pas de caractères qui sont spéciaux pour XML, ou alors il faut tout mettre dans un bloc `CDATA` ou encore encoder tout en Base64, au prix d'un accroissement de taille).

Autre système qui aura besoin de JOSE, ALTO (RFC 6708). Ce système permet à un client de s'informer, auprès d'un serveur, sur le pair le plus « proche » pour les cas où plusieurs pairs peuvent rendre le même service. Dans son mode le plus simple, ALTO est simplement un protocole requête/réponse, où les échanges se font en JSON sur HTTP. HTTPS est suffisant pour sécuriser ce mode. Mais les futures versions d'ALTO pourraient avoir des objets d'information relayés entre plusieurs parties, et JOSE deviendrait alors utile pour les sécuriser.

Continuons la riche liste de cas d'école possibles pour JOSE, avec les systèmes d'alerte. Il y a des travaux en cours à l'IETF <http://tools.ietf.org/wg/atoca> sur l'utilisation de l'Internet pour diffuser des alertes, par exemple concernant une catastrophe naturelle proche. Une exigence absolument critique de ces systèmes est l'impossibilité de fabriquer une fausse alerte. Si c'était possible, une grave panique pourrait être déclenchée à volonté. Et, sans même parler des effets de la panique, la confiance dans le système disparaitrait rapidement. Or, les alertes doivent être diffusées très vite, à beaucoup de gens, quel que soit le mécanisme par lequel ils sont actuellement connectés. Elles passent donc par des intermédiaires, cherchant à toucher tout le monde. Une protection du canal ne suffit donc pas, une protection du message (du texte JSON) est nécessaire.

Dernier cas que je vais citer, l'API Web Cryptography <http://www.w3.org/TR/WebCryptoAPI/>. Cette API normalisée permet notamment au code JavaScript s'exécutant dans un navigateur de demander au navigateur de réaliser des opérations cryptographiques. L'un des intérêts (par rapport à une mise en œuvre complètement en JavaScript) est la sécurité : les clés peuvent rester dans le navigateur, JavaScript n'y a pas accès. Ceci dit, dans certains cas, la capacité d'exporter une clé (par exemple pour la copier vers un autre appareil) est utile. L'API prévoit donc cette fonction (avec, on l'espère, de stricts contrôles et vérifications) et cela implique donc un format standard pour ces clés (publiques ou privées). Au moins pour les clés privées, la confidentialité de ce format est cruciale. JOSE (son format JWK) va donc être utilisé <http://www.w3.org/TR/WebCryptoAPI/#jose>.

La section 6 du RFC liste les exigences précises du projet JOSE. Chacune, pour faciliter les références ultérieures, porte un identificateur composé d'une lettre et d'un nombre. La lettre est F pour les exigences fonctionnelles, S pour celles de sécurité et D si c'est une simple demande, pas une exigence. Par exemple, F1 est simplement l'exigence de base d'un format standard permettant authentification, intégrité et confidentialité. F2 et F3 couvrent le format des clés (dans le cas de la cryptographie symétrique comme dans celui de l'asymétrique). F4 rappelle que le format doit être du JSON, et F5 qu'il faut une forme compacte, utilisable dans les URL.

Parmi les simples désirs, D1 rappelle l'importance de se coordonner avec le groupe de travail WebCrypto du W3C <http://www.w3.org/2012/webcrypto/>, D2 souhaite qu'on n'impose pas (contrairement à beaucoup d'autres normes de cryptographie comme XML Signature) de canonicalisation du contenu, D3 voudrait qu'on se focalise sur le format, pas sur les opérations, de manière que les formats JOSE soient utilisables par des applications très différentes, utilisant des protocoles de cryptographie bien distincts.

À noter que la section 7 revient sur la canonicalisation en reconnaissant que, comme il n'existe pas de moyen automatique simple de déterminer si deux textes JSON représentent la même information, cela peut avoir des conséquences sur la sécurité de JOSE.

Merci à Virginie Galindo pour sa relecture.