RFC 9230 : Oblivious DNS over HTTPS

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 9 juin 2022

Date de publication du RFC : Juin 2022

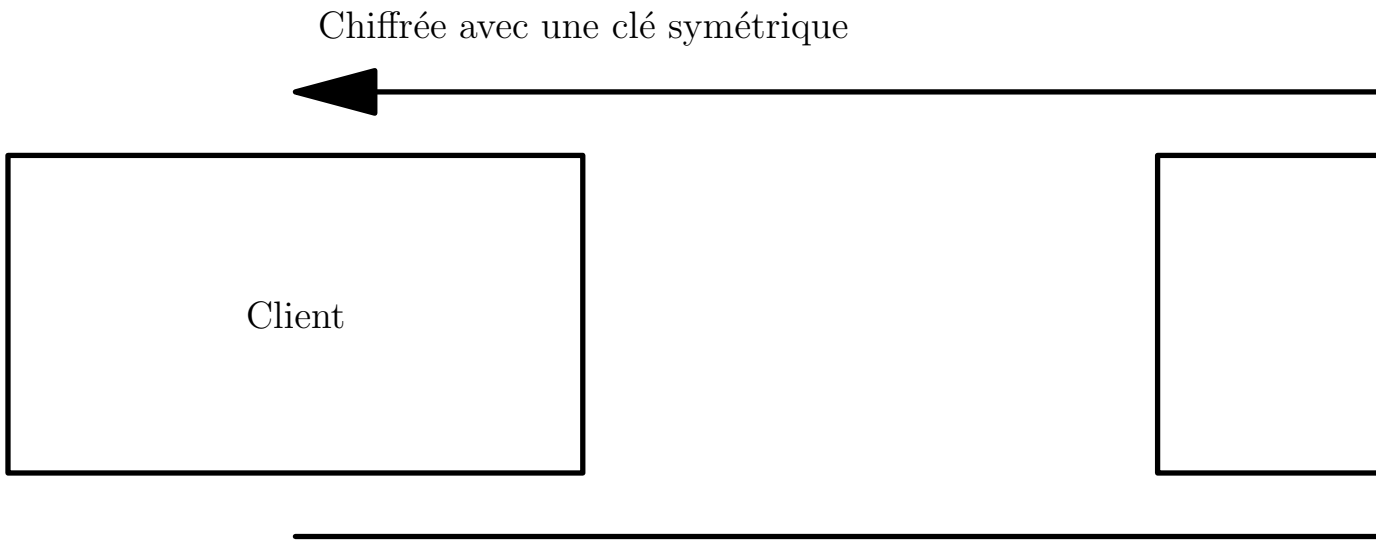
https://www.bortzmeyer.org/9230.html

Des protocoles de DNS sécurisés comme DoH protègent la liaison entre un client et un résolveur DNS <https://www.bortzmeyer.org/resolveur-dns.html>. Mais le résolveur, lui, voit tout, aussi bien le nom de domaine demandé que l'adresse IP du client. Ce nouveau RFC décrit un mécanisme expérimental de relais entre le client et le serveur DoH, permettant de cacher l'adresse IP du client au serveur, et la requête au relais. Une sorte de Tor minimum.

Normalement, DoH (normalisé dans le RFC 8484 [1]) résout le problème de la surveillance des requêtes DNS. Sauf qu'évidemment le serveur DoH lui-même, le résolveur <https://www.bortzmeyer.org/resolveur-dns.html>, voit le nom demandé, et la réponse faite. (Les utilisateurices de l'Internet oublient souvent que la cryptographie, par exemple avec TLS, protège certes contre le tiers qui écoute le réseau mais pas du tout contre le serveur avec qui on parle.) C'est certes nécessaire à son fonctionnement mais, comme il voit également l'adresse IP du client, il peut apprendre pas mal de choses sur le client. Une solution possible est d'utiliser DoH (ou DoT) sur Tor (exemple plus loin). Mais Tor est souvent lent, complexe et pas toujours disponible (cf. annexe A du RFC). D'où les recherches d'une solution plus légère, "*Oblivious DoH*", décrit dans ce nouveau RFC. L'idée de base est de séparer le routage des messages (qui nécessite qu'on connaisse l'adresse IP du client) et la résolution DNS (qui nécessite qu'on connaisse la requête). Si ces deux fonctions sont assurées par deux machines **indépendantes**, on aura amélioré la protection de la vie privée.

Les deux machines vont donc être :

— L'"*Oblivious Proxy*", le relais qui connait l'adresse IP du client mais ne voit passer qu'une requête DNS chiffrée ; ce n'est pas un relais HTTP complet, il est spécialisé dans DoH.
— Et l'"*Oblivious Target*", le serveur DoH (le résolveur, donc) ; il a une clé publique avec laquelle le client chiffrera les requêtes à envoyer.



Un schéma simplifié :

Il est important de répéter que, si le relais et le serveur sont complices, "*Oblivious DoH*" perd tout intérêt. Ils doivent donc être gérés par des organisations différentes.

La découverte du relais, et de la clé publique du serveur, n'est pas traité dans ce RFC. Au moins au début, elle se fera « manuellement ».

Le gros morceau du RFC est sa section 4. Elle explique les détails du protocole. Le client qui a une requête DNS à faire doit la chiffrer avec la clé publique du serveur. Ce client a été configuré avec un gabarit d'URI (RFC 6570) et avec l'URL du serveur DoH, serveur qui doit connaitre le mécanisme "*Oblivious DoH*". Le gabarit doit contenir deux variables, `targethost` (qui sera incarné avec le nom du serveur DoH) et `targetpath` (qui sera incarné avec le chemin dans l'URL du serveur DoH). Par exemple, un gabarit peut être `https://dnsproxy.example/{targethost}/{targetpath}`. Le client va ensuite faire une requête (méthode HTTP `POST`) vers le relais. La requête aura le type `application/oblivious-dns-mess` (la réponse également, donc le client a intérêt à indiquer `Accept: application/oblivious-dns-message`).

En supposant un serveur DoH d'URL `https://dnstarget.example/dns`, la requête sera donc (en utilisant la syntaxe de description de HTTP/2 et 3) :

```
:method = POST
:scheme = https
:authority = dnsproxy.example
:path = /dnstarget.example/dns
accept = application/oblivious-dns-message
content-type = application/oblivious-dns-message
content-length = 106

[La requête, chiffrée]
```

1. Pour voir le RFC de numéro NNN, `https://www.ietf.org/rfc/rfcNNN.txt`, par exemple `https://www.ietf.org/rfc/rfc8484.txt`

Le relais enverra alors au serveur :

```
:method = POST
:scheme = https
:authority = dnstarget.example
:path = /dns
accept = application/oblivious-dns-message
content-type = application/oblivious-dns-message
content-length = 106

[La requête, chiffrée]
```

Le relais ne doit pas ajouter quelque chose qui permettrait d'identifier le client, ce qui annulerait tout l'intérêt d'"*Oblivious DoH*". Ainsi, le champ `forwarded` (RFC 7239) est interdit. Et bien sûr, on n'envoie pas de "*cookies*", non plus.

La réponse utilise le même type de médias :

```
:status = 200
content-type = application/oblivious-dns-message
content-length = 154

[La réponse, chiffrée]
```

Le relais, en la transmettant, ajoutera un champ `proxy-status` (RFC 9209).

Les messages de type `application/oblivious-dns-message` sont encodés ainsi (en utilisant le langage de description de TLS, cf. RFC 8446, section 3) :

```
struct {
      opaque dns_message<1..2^16-1>;
      opaque padding<0..2^16-1>;
} ObliviousDoHMessagePlaintext;
```

Bref, un message DNS binaire, et du remplissage. Le remplissage est nécessaire car TLS, par défaut, n'empêche pas l'analyse de trafic. (Lisez le RFC 8467.) Le tout est ensuite chiffré puis mis dans :

```
struct {
      uint8  message_type;
      opaque key_id<0..2^16-1>;
      opaque encrypted_message<1..2^16-1>;
} ObliviousDoHMessage;
```

Le `key_id` servant à identifier ensuite le matériel cryptographique utilisé.

J'ai parlé à plusieurs reprises de chiffrement. Le client doit connaitre la clé publique du serveur DoH pour pouvoir chiffrer. (Le RFC ne prévoit pas de mécanisme pour cela.) Comme avec la plupart des systèmes de chiffrement, le chiffrement sera en fait réalisé avec un algorithme symétrique. La transmission ou la génération de la clé de cet algorithme utlisera le mécanisme HPKE ("*Hybrid Public Key Encryption*", spécifié dans le RFC 9180).

Outre l'analyse de trafic, citée plus haut, une potentielle faille d'"*Oblivious DoH*" serait le cas d'un serveur indiscret qui essaierait de déduire des informations de l'établissement des connexions DoH vers lui. Le RFC recommande que les relais établissent des connexions de longue durée et les réutilisent pour plusieurs clients, rendant cette éventuelle analyse plus complexe.

Notez aussi que le relais est évidemment libre de sa politique. Les relais peuvent par exemple ne relayer que vers certains serveurs DoH connus.

Le serveur DoH ne connait pas l'adresse IP du client (c'est le but) et ne peut donc pas transmettre aux serveurs faisant autorité <https://www.bortzmeyer.org/serveur-dns-faisant-autorite.html> les informations qui peuvent leur permettre d'envoyer une réponse adaptée au client final. Si c'est un problème, le client final peut toujours utiliser ECS ("*EDNS Client Subnet*", RFC 7871), en indiquant un préfixe IP assez générique pour ne pas trop en révéler. Mais cela fait évidemment perdre une partie de l'intérêt d'"*Oblivious DoH*".

Bon, et questions mises en œuvre de ce RFC ? Il y en a une dans DNScrypt <https://dnscrypt.info/> (et ils publient une liste de serveurs et de relais <https://github.com/DNSCrypt/dnscrypt-resolver
list-of-odoh-servers-and-relays>).

"*Oblivious DoH*" est en travaux depuis un certain temps (voyez par exemple cet article de Cloudflare <https://blog.cloudflare.com/fr-fr/oblivious-dns-fr-fr/>). Notez que la technique de ce RFC est spécifique à DoH. Il y a aussi des travaux en cours à l'IETF pour un mécanisme plus général, comme par exemple un "*Oblivious HTTP*" <https://datatracker.ietf.org/doc/draft-ietf-ohai-ohttp/>, qui serait un concurrent « "*low cost*" » de Tor.

À propos de Tor, j'avais dit plus haut qu'on pouvait évidemment, si on ne veut pas révéler son adresse IP au résolveur DoH, faire du DoH sur Tor. Voici un exemple de requête DoH faite avec le client kdig et le programme torsocks pour faire passer les requêtes TCP par Tor :

```
% torsocks kdig +https=/ @doh.bortzmeyer.fr ukraine.ua
;; TLS session (TLS1.3)-(ECDHE-SECP256R1)-(RSA-PSS-RSAE-SHA256)-(AES-256-GCM)
;; HTTP session (HTTP/2-POST)-(doh.bortzmeyer.fr/)-(status: 200)
...
;; ANSWER SECTION:
ukraine.ua.          260 IN A 104.18.7.16
ukraine.ua.          260 IN A 104.18.6.16
...
;; Received 468 B
;; Time 2022-06-02 18:45:15 UTC
;; From 193.70.85.11@443(TCP) in 313.9 ms
```

Et ce résolveur DoH ayant un service `.onion`, on peut même :

```
% torsocks kdig +https=/ @lani4a4fr33kqqjeiy3qubhfx2jewfd3aeaepuwzxrx6zywp2mo4cjad.onion ukraine.ua
;; TLS session (TLS1.3)-(ECDHE-SECP256R1)-(RSA-PSS-RSAE-SHA256)-(AES-256-GCM)
;; HTTP session (HTTP/2-POST)-(lani4a4fr33kqqjeiy3qubhfx2jewfd3aeaepuwzxrx6zywp2mo4cjad.onion/)-(status: 200)
...
;; ANSWER SECTION:
ukraine.ua.          212 IN A 104.18.6.16
ukraine.ua.          212 IN A 104.18.7.16
...
;; Received 468 B
;; Time 2022-06-02 18:46:00 UTC
;; From 127.42.42.0@443(TCP) in 1253.8 ms
```

Pour comparer les performances (on a dit que la latence <https://www.bortzmeyer.org/latence.html> de Tor était franchement élevée), voici une requête non-Tor vers le même résolveur :

```
% kdig +https=/ @doh.bortzmeyer.fr ukraine.ua
;; TLS session (TLS1.3)-(ECDHE-SECP256R1)-(RSA-PSS-RSAE-SHA256)-(AES-256-GCM)
;; HTTP session (HTTP/2-POST)-([2001:41d0:302:2200::180]/)-(status: 200)
...
;; ANSWER SECTION:
ukraine.ua.          300 IN A 104.18.6.16
ukraine.ua.          300 IN A 104.18.7.16
...
;; Received 468 B
;; Time 2022-06-02 18:49:44 UTC
;; From 2001:41d0:302:2200::180@443(TCP) in 26.0 ms
```