Chiffrement de messagerie quasi instantanée : à quel protocole se vouer ?

Florian Maury – ANSSI/SDE/ST/LRP

Mars 2017

Résumé

Telegram, Whatsapp, Signal, OTR...et autant de protocoles de messagerie quasi instantanée, de modèles de sécurité et de protocoles cryptographiques : lesquels choisir ? Et si la solution idéale n'était pas dans la liste précédente ? Cet article évoque les limites de plusieurs de ces solutions, et présente le cœur cryptographique de Signal, Whatsapp et du protocole OMEMO. Il met finalement en exergue, par une analyse comparative, certaines limites de Signal et des qualités d'OMEMO.

Chiffrement des messages de bout en bout ou chiffrement du canal, qualité des algorithmes cryptographiques et de l'authentification des pairs, confidentialité persistante (*Perfect Forward Secrecy* ou PFS) ou non, fiabilité de l'équipement faisant tourner la solution, limitation de l'exposition des métadonnées, fuite du carnet d'adresses, localisation des serveurs, capacité à dénier l'envoi d'un message : des critères rarement considérés par les utilisateurs de messagerie. Leurs véritables critères de sélection sont souvent la taille du réseau social joignable, la facilité d'utilisation, l'accessibilité de la solution sur l'équipement de l'utilisateur ou encore la liste des services annexes. Un constat compréhensible, puisque la première liste est formée de critères abscons et rarement absolus, tandis que la seconde liste affecte l'usage quotidien.

Certains utilisateurs restent néanmoins soucieux de leur vie privée ou sont contraints par la nature de leurs activités en ligne à un niveau de sécurité plus élevé. Ils disposent alors d'une myriade d'options, dont Telegram, Whatsapp, Signal, ou Apple iMessage. Ils peuvent aussi se tourner vers des solutions de messagerie instantanées traditionnelles augmentées du protocole Off-the-Record (OTR) [otr] ou encore des emails augmentés grâce à OpenPGP [opg].

Se pose alors la question de séparer le bon grain de l'ivraie ; une tâche qui est simplifiée par la forte concentration des applications autour des protocoles X3DH [x3d] et Double Ratchet [dr]. Ces deux protocoles, spécifiés récemment dans le domaine public par les auteurs de Signal, sont employés par plusieurs vendeurs, dont Signal et Whatsapp. En outre, la communauté de XMPP, un protocole de messagerie quasi instantanée, a également choisi X3DH+Double Ratchet, afin de remplacer leur usage d'OpenPGP et OTR.

La combinaison X3DH+Double Ratchet n'est cependant qu'une partie de la solution pour sécuriser les communications. Plus spécifiquement, ces protocoles permettent, respectivement, la négociation des clés et leur rafraîchissement. L'utilisation de ces clés, afin de créer des sessions cryptographiques entre les utilisateurs, est dévolue à d'autres composants : le protocole Signal, dans le cas de Signal et de Whatsapp, et le protocole OMEMO [ome], dans le cas de XMPP.

X3DH+Double Ratchet, OMEMO et Signal sont étudiés dans la suite de cet article. Pour le lecteur intéressé, la sécurité de Telegram a fait l'objet de discussions [tela] et d'études [telb] et l'université de Johns Hopkins a étudié celle d'Apple iMessage [ims]. WeChat ou encore Slack sont hors sujet, puisqu'ils ne proposent pas de chiffrement de bout en bout.

1 Les protocoles inadéquats pour la messagerie quasi instantanée

1.1 OpenPGP

OpenPGP est un format de stockage de messages et de clés cryptographiques, spécifié dans sa version la plus récente en 2007 [opg]. Il est notamment employé pour sécuriser des messages électroniques. Ce format étant agnostique vis-à-vis de la nature des messages, il convient pour des réseaux de messagerie décentralisés ou lorsque le destinataire est hors-ligne (protocole non interactif). Il permet également d'assurer de la protection de bout en bout, puisque ce sont les messages qui sont chiffrés et non le canal de transport de ces derniers.

En outre, il est possible d'utiliser OpenPGP pour envoyer des messages à un groupe d'utilisateurs. Pour ce faire, la clé de chiffrement du message est chiffrée avec les clés publiques de chacun des destinataires. Les clés publiques ainsi utilisées doivent être préalablement créées, et stockées dans des certificats OpenPGP. Ces certificats permettent d'associer des identités – éventuellement des pseudonymes – à ces clés publiques. Ces certificats sont transmis ponctuellement aux autres utilisateurs grâce à des annuaires ou par une remise en main propre.

OpenPGP présente cependant des limites, en regard de solutions alternatives spécialisées pour le chiffrement de messagerie quasi instantanée. Ainsi, il n'offre pas, de manière inhérente, de confidentialité persistante du fait de la transmission ponctuelle des certificats : le rafraîchissement des clés est du ressort de l'utilisateur. En outre, pour la protection en intégrité des messages, l'utilisateur n'a le choix qu'entre des solutions imparfaites. L'une est sujette à des attaques par dégradation du niveau de sécurité (*downgrade attack*) [for] et l'autre use de signatures cryptographiques non répudiables, ce qui n'est pas toujours souhaitable.

1.2 Off-the-Record

Le protocole *Off-the-Record* a été spécifié, pour la première fois, en 2004 ; sa version la plus récente date de 2012. Ce mécanisme permet l'échange de clés et de messages sécurisés de bout en bout. L'établissement de la session protégée est effectué entre deux parties de manière interactive. Autrement dit, il est nécessaire que les participants soient en ligne simultanément pour l'établissement de

cette session. Cette propriété exclut un usage asynchrone, et restreint donc ce protocole à la messagerie instantanée. Une variante, appelée mpOTR [mpo], permet d'échanger des messages au sein d'un groupe d'utilisateurs.

Avec OTR, les utilisateurs sont identifiés par des clés publiques à long terme. Les clés à long terme servent à signer/authentifier des échanges de clés Diffie-Hellman (DH) éphémères. Ces clés éphémères servent, à leur tour, à générer les clés protégeant les messages. Il convient de noter que les signatures émises par les clés à long terme affectent la vie privée ; elles constituent, en effet, une preuve non répudiable qu'une conversation a eu lieu entre deux parties, même si le contenu de la conversation reste inconnu d'un observateur.

Une fois la négociation de clés initiale accomplie, de nouvelles biclés DH sont introduites à chaque nouveau message. Cela permet ainsi de rafraîchir les secrets et d'apporter la confidentialité persistante. En effet, la compromission d'une clé privée DH n'affecte la confidentialité que d'un unique message ; de même, la compromission de la clé à long terme n'affecte que les futurs messages.

Le protocole *Off-the-Record* présente également une propriété de sécurité qui serait favorable à la vie privée. Ainsi, *Off-the-Record* permettrait à un expéditeur de dénier le contenu d'un message, tout en garantissant au destinataire la légitimité et l'intégrité du message reçu. Pour ce faire, les clés utilisées pour calculer des motifs d'intégrité de messages sont divulguées en clair après réception et vérification de ces messages. Conjuguée à un mode de chiffrement malléable et à un clair connu, cette méthode peut permettre à un observateur de forger un message chiffré et intègre arbitraire. Cet observateur serait cependant incapable de prouver à un tiers l'authenticité d'un message qu'il détient. L'efficacité de ce mécanisme fait néanmoins débat parmi les experts, car il n'a jamais été éprouvé dans le cadre d'un procès, afin de décrédibiliser une conversation enregistrée et utilisée comme preuve incriminante.

Malgré ses nombreux avantages, le protocole *Off-the-Record*, tel que spécifié dans sa version la plus récente (3.4), souffre d'un problème d'obsolescence cryptographique. En effet, l'absence d'un mécanisme de négociation des algorithmes fait d'*Off-the-Record* un musée des algorithmes cryptographiques des années 2000. Sont notamment employés l'algorithme de signature DSA, des empreintes cryptographiques avec SHA-1, ou encore des échanges DH sur corps entiers de 1536 bits avec un groupe fixé par la spécification. L'usage de cette cryptographie datée est contraire aux bonnes pratiques actuellement reconnues.

2 X3DH+Double Ratchet

Les protocoles X3DH et Double Ratchet ont été inventés par Trevor Perrin et Moxie Marlinspike. En 2013, il s'agissait, en fait, d'un seul et même protocole connu sous le nom d'Axolotl. Ce n'est qu'en 2016 qu'Axolotl fut divisé et que ses parties furent renommées afin de mettre fin à des confusions fréquentes entre Axolotl et le protocole Signal. Il faut dire que le protocole Signal, qui fait usage d'une variante d'Axolotl, n'a, à ce jour, jamais été spécifié ou documenté et que la frontière entre les deux protocoles était donc pour le moins floue. Les spécifications complètes de X3DH et Double Ratchet ont finalement été publiées en novembre 2016. Cette publication a également permis de mettre fin à de récurrentes menaces judiciaires que les auteurs de Signal ont pu proférer contre des vendeurs prétendant implémenter le protocole Signal, alors qu'ils utilisaient

réellement Double Ratchet [leg].

X3DH est responsable de la négociation de clés cryptographiques. Celle-ci prend place lors d'une phase initiale. Les clés évoluent ensuite par dérivations, selon le protocole Double Ratchet. Ce dernier rafraîchit les clés à l'aide de cryptographie symétrique, ainsi que par l'apport régulier de nouveaux éléments secrets asymétriques.

Pour effectuer ces opérations sur les clés, les deux protocoles emploient de la cryptographie moderne : XEdDSA, une extension à EdDSA, sur les courbes elliptiques curve25519 ou curve448 [xds], SHA2 et HKDF [hkd].

2.1 X3DH

Chaque utilisateur du protocole X3DH doit générer et publier un ensemble de biclés cryptographiques. Ces clés doivent être compatibles avec les fonctions X25519 ou X448 de XEdDSA.

La première biclé est appelée clé à long terme. Elle sert dans le cadre d'échanges DH, mais elle est également employée pour signer d'autres biclés, appelées clés à moyen terme. Des biclés à usage supposément unique sont également générées en grande quantité ; Signal et Conversations en génèrent ainsi une centaine. Générer autant de clés à usage unique permet qu'un grand nombre de sessions puissent être établies avec la confidentialité persistante dès le premier message, et ce alors que le destinataire n'est pas en ligne.

La génération de ces trois types de biclés (à long et moyen termes et à usage unique) doit être répétée pour chacun des périphériques avec lesquels un utilisateur est susceptible d'accéder à ses messages. L'utilisateur disposant d'un PC, d'une tablette et d'un téléphone portable se retrouve ainsi rapidement avec plusieurs centaines de biclés associées à son identifiant. Seule la clé à long terme de chaque équipement nécessite cependant une vérification d'authenticité par les autres utilisateurs.

Ces biclés sont utilisées afin d'établir des sessions entre un expéditeur et l'ensemble des périphériques des destinataires. Ces périphériques peuvent être possédés par un même destinataire ou par plusieurs destinataires, dans le cadre d'une discussion de groupe. La liste des périphériques destinataires peut même contenir les équipements de l'expéditeur, afin de permettre la synchronisation des messages émis entre équipements.

Autant de sessions sont créées qu'il y a d'équipements destinataires. Cette étape n'a cependant besoin de se produire qu'une seule fois, lors de la première conversation entre deux périphériques. Ces sessions ont, en effet, une durée de vie illimitée.

Pour établir une session, la première étape consiste à récupérer les clés publiques des périphériques destinataires. La manière dont elles sont publiées et récupérées est laissée ici volontairement abstraite ; elle varie d'une implémentation à l'autre, comme le détaillera la section 3 de cet article.

Une fois les clés publiques des destinataires en possession de l'expéditeur, ce dernier effectue les mêmes étapes avec les clés de chaque périphérique pour lequel une session doit être établie. La première étape consiste à générer une nouvelle biclé DH éphémère. Trois à quatre échanges DH sont ensuite effectués, entre les clés de l'équipement expéditeur et celles de l'équipement destinataire. L'appariement des clés publiques DH est détaillé dans la figure 1.

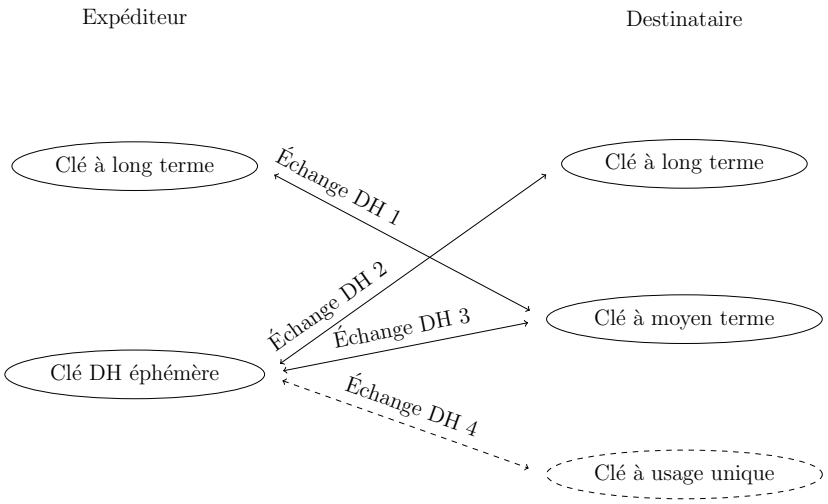

FIGURE 1 – Illustration des échanges de clés DH effectués dans le cadre de X3DH. Le trait en pointillé représente un échange optionnel, qui n'a lieu que si une clé à usage unique est disponible pour l'équipement destinataire.

La variabilité du nombre d'échanges DH résulte de la capacité à récupérer une des clés à usage unique pour l'équipement destinataire. Certains dépôts de clés tiennent, en effet, une comptabilité afin d'assurer qu'une clé à usage unique n'est bien distribuée qu'une seule fois. Si toutes les clés ont été distribuées, aucune n'est fournie à l'expéditeur et seuls trois échanges DH sont opérés. Ceci peut affecter la confidentialité persistante, car le destinataire ne fournit alors que des clés qui sont partagées entre plusieurs sessions. Dans la section 2.2 traitant de Double Ratchet, il sera détaillé comment cette faiblesse est cicatrisée dès la réception d'un message de la part de l'équipement destinataire.

L'ensemble des secrets résultant des échanges DH sont ensuite concaténés et passés à travers la fonction HKDF pour former une valeur secrète, appelée secret racine de la session.

Cet échange de clés a la particularité de négocier un secret tout en préservant la capacité des deux parties de nier avoir tenu une conversation ensemble. Cette propriété est dérivée de l'hypothèse de difficulté calculatoire de DH (*Computational DH Assumption*). La signature XEdDSA des clés à moyen terme, qui elle est non répudiable, ne prévient pas cette propriété puisqu'elle est totalement décorrélée de l'échange de clés et n'intervient que pour « certifier » la clé à moyen terme.

2.2 Double Ratchet

Double Ratchet repose sur deux mécanismes de rafraîchissement des clés. Ces deux mécanismes confèrent son nom à cet algorithme, puisqu'ils utilisent tous deux, à l'instar d'une roue à rochet, des fonctions cryptographiques à sens unique pour faire « évoluer » des secrets.

Le premier, représenté sur fond jaune dans la figure 2, utilise exclusivement la cryptographie symétrique. Il permet de générer les clés secrètes protégeant

les messages. Avec ce mécanisme, chaque message bénéficie d'une clé secrète à usage unique. Cette clé de protection d'un message est obtenue par dérivation d'une clé, tirée d'un ensemble appelé chaîne de clés. Cette chaîne est formée par des dérivations successives de secrets. Le secret initial de cette chaîne est le secret racine actuel. La notion d'actualité du secret racine provient du second mécanisme de rafraîchissement des clés.

Ce second mécanisme, représenté sur fond vert dans la figure 2, utilise de la cryptographie asymétrique. Il vise à faire évoluer la clé racine qui a été négociée initialement, par exemple avec X3DH. Avec ce mécanisme, une nouvelle biclé DH est tirée aléatoirement chaque fois qu'un périphérique s'apprête à envoyer un message consécutif à la réception d'un message par un autre périphérique. La clé publique de cette nouvelle biclé DH est jointe à l'ensemble des messages envoyés par ce périphérique jusqu'à la réception d'un message de la part d'un autre périphérique. Cette gymnastique est représentée dans la figure 3.

La nouvelle clé DH fraîchement tirée est utilisée dans un échange DH en conjonction avec les clés publiques les plus récentes reçues dans des messages émis par les autres périphériques. Le résultat de cet échange est ensuite « mélangé » avec le secret racine actuel à l'aide de la fonction HKDF. Le résultat de cette opération devient le nouveau secret racine actuel.

L'utilisation de ces deux mécanismes de rafraîchissement de clés permet de bénéficier de clés secrètes uniques pour chaque message envoyé, y compris lorsqu'un des participants se lance dans un monologue de plusieurs messages. La compromission d'une clé symétrique ne mène alors qu'à la compromission d'un seul message. La réception d'un message de la part de l'autre participant permet ensuite de rafraîchir le secret racine. Ceci permet ainsi de prévenir la compromission de plus d'un monologue en cas de compromission d'une clé asymétrique.

Outre ces propriétés, les protocoles de messageries sécurisées reposant sur Double Ratchet peuvent également se montrer tolérants vis-à-vis de la perte de messages ou de la livraison de messages dans le désordre. Il suffit à ces applications de conserver les différentes chaînes de clés, et de « sauter » les messages encore non reçus, en appliquant plusieurs fois de suite la fonction HMAC-SHA256 avec la « constante 2 » de la figure 2. Il convient néanmoins de noter que conserver ainsi les clés, au lieu de les supprimer dès que possible, peut mettre en péril la confidentialité persistante, en cas de compromission d'un équipement.

2.3 Intégration de X3DH+Double Ratchet dans OMEMO

La première version d'OMEMO a été spécifiée par Andreas Straub en 2015, avec l'aide de Daniel Gultsch, développeur principal du client XMPP Conversations. En décembre 2016, OMEMO a été officiellement acceptée comme extension expérimentale du protocole XMPP (XEP-0384).

Avec XMPP, chaque utilisateur est identifié par un Jabber ID (JID). Il s'agit d'un identifiant qui ressemble fort à une adresse email, mais il est suivi d'une barre oblique (*slash*) et du nom d'un équipement ou d'un logiciel. *florian@im.x-cli.eu/phone* est, par exemple, le JID de l'auteur de cet article lorsqu'il est connecté avec son téléphone. À l'instar des adresses email, la partie précédant l'arobase désigne un utilisateur local, tandis que la partie suivant l'arobase et jusqu'à la barre oblique désigne le serveur sur lequel est hébergé cet utilisateur.

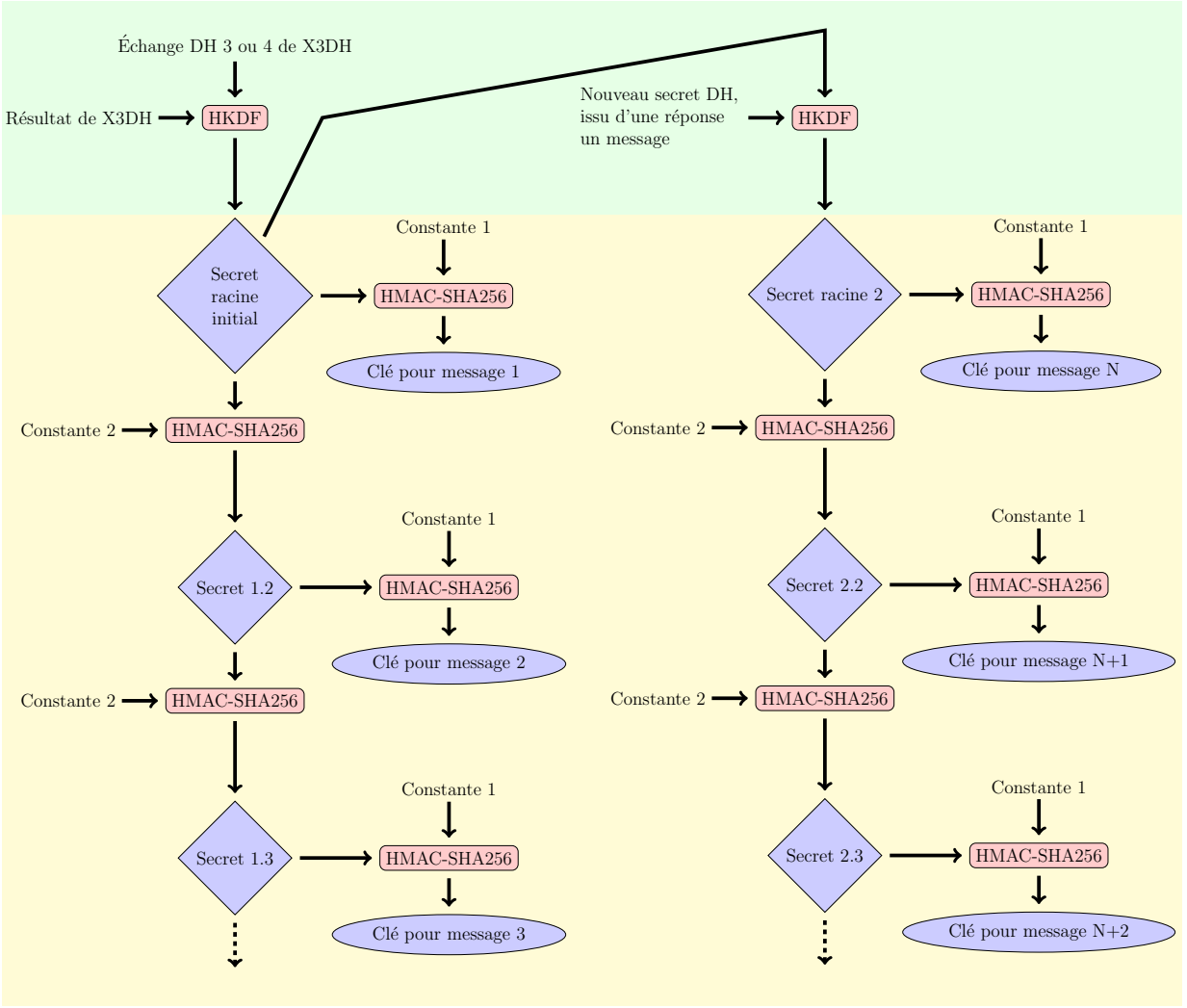

FIGURE 2 – Illustration de l'algorithme Double Ratchet. Les clés composant la chaîne de clés sont représentées dans des diamants. Les clés de messages sont représentées dans des ellipses. Les algorithmes sont dessinés dans des boîtes roses aux bords arrondis. Les composants sur fond vert représentent la roue à rochet asymétrique. Les composants sur fond jaune représentent la roue à rochet symétrique.

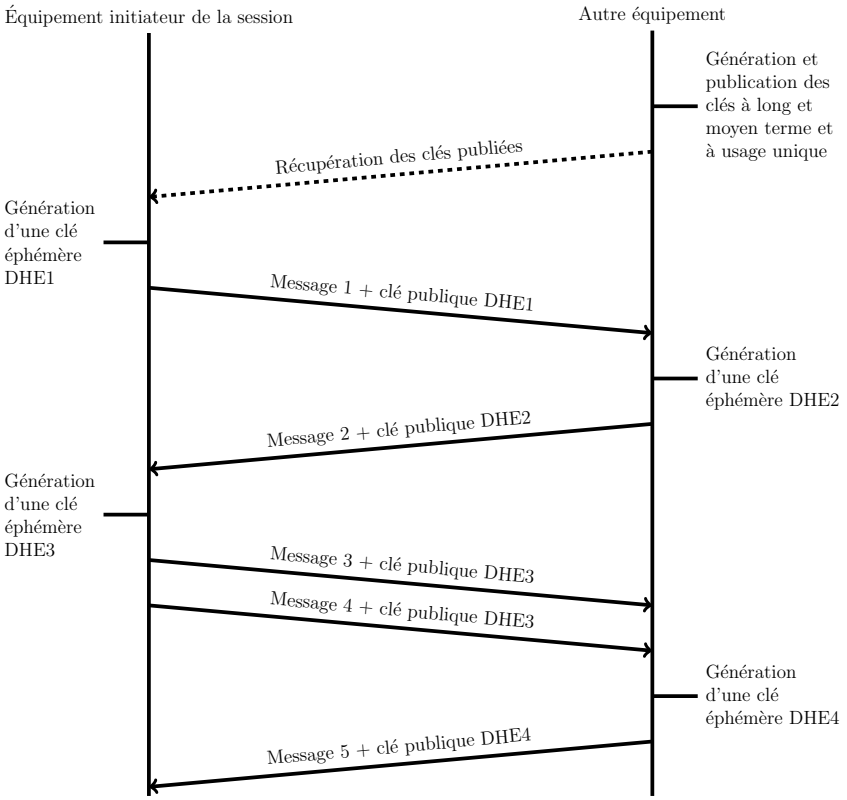

FIGURE 3 – Illustration d'un enchaînement de messages avec un protocole de messagerie employant Double Ratchet. De nouvelles clés asymétriques sont générées juste avant l'envoi d'un message faisant suite à une réponse. Cette clé asymétrique est répétée dans tous les messages suivants.

XMPP est donc un système fédéré, où chaque utilisateur choisit son fournisseur de service.

Les messages à destination d'un utilisateur, désigné par son *bare JID* (c.-à-d. son JID sans le nom de l'équipement), sont délivrés au dernier périphérique actif de cet utilisateur. Ce fonctionnement peut être altéré grâce aux extensions Carbon Messages (XEP-0280) et Message Archive Management (XEP-0313) afin que tous les équipements d'un utilisateur aient accès à tous les messages reçus. En outre, si un utilisateur est hors-ligne, ses messages peuvent être stockés sur le serveur responsable du compte de l'utilisateur. Ces messages lui seront alors délivrés lors de sa prochaine connexion. Par ailleurs, des informations relatives à un utilisateur peuvent être stockées, en clair, sur le serveur responsable de son compte, grâce à l'extension XMPP XEP-0163, appelée Personal Eventing Protocol (PEP). Ce mécanisme, lui-même extensible, permet ainsi à un utilisateur connecté ou hors-ligne de mettre à la disposition d'autres utilisateurs diverses informations comme son avatar, son dernier message de statut, ou encore sa « carte de visite ».

OMEMO utilise ces différentes extensions XMPP pour offrir une méthode de chiffrement de bout en bout à l'ergonomie moderne. Chaque équipement, lorsqu'il active OMEMO, génère sa clé à long terme, sa clé à moyen terme et d'une vingtaine à une centaine de clés à usage « unique ». Les jeux de clés de chaque équipement sont stockés dans le profil PEP utilisateur. Lorsqu'un expéditeur souhaite établir une nouvelle session, il récupère toutes les clés auprès du serveur responsable du compte du destinataire. Ensuite, il effectue un échange de clés en suivant le protocole X3DH. Par équipement avec lequel une session doit être établie, il sélectionne aléatoirement une clé à « usage unique ». Cette sélection aléatoire permet de s'affranchir du risque qu'un serveur malveillant dégrade la confidentialité persistante en diffusant sciemment une clé à usage unique déjà employée par ailleurs. Le risque de dégradation de la confidentialité persistante n'est cependant pas totalement évité. Il est, en effet, possible que plusieurs sessions soient établies pendant qu'un périphérique est hors-ligne, et que la sélection aléatoire provoque une collision. Plus un périphérique tarde à remplacer les clés utilisées, et plus le risque de collision grandit. Cette collision a une portée limitée puisqu'elle n'a un impact que sur tout ou partie du premier monologue d'un utilisateur. Si l'équipement, dont l'une des clés à usage unique a été réutilisée, se connecte et détecte cette situation, il est alors en mesure de rétablir la confidentialité persistante. Il lui suffit d'envoyer un message « de service » dont le seul objet est de rafraîchir le secret racine. Cette situation semble, dans tous les cas, préférable à l'absence totale du quatrième échange DH.

3 Des divergences entre Signal et le protocole OMEMO

Signal et le protocole OMEMO sont par certains aspects très similaires. Certaines implémentations d'OMEMO utilisent, en effet, la bibliothèque cryptographique libsignal [lib] d'Open Whisper Systems, la société éditrice de Signal. Ses usages sont cependant cantonnés aux échanges de clés X3DH et à la maintenance des secrets avec Double Ratchet. Les points de divergences entre Signal

et les implémentations d'OMEMO se situent donc sur les points suivants : les identifiants et l'architecture réseau, l'infrastructure de gestions de clés, l'usage qui est fait de ces clés, l'existence de documentation de qualité et la capacité à auditer le protocole.

3.1 Les identifiants et l'infrastructure réseau

Alors qu'OMEMO utilise une infrastructure répartie, où chaque utilisateur choisit son fournisseur de services, grâce au réseau fédéré XMPP, Signal préfère une infrastructure centralisée. Les auteurs de Signal ont, en effet, exprimé une opinion très négative sur les notions de fédération et d'interopérabilité, les qualifiant de causes génératrices de l'ossification des protocoles [nof]. Au diable donc l'Internet ouvert, le succès de HTTP ou encore celui des courriers électroniques. Moxie Marlinspike a même demandé à des clones (*fork*) de Signal de ne pas employer les serveurs opérés par Open Whisper Systems, quand bien même le protocole employé était le même [nok]. Signal est donc un système clos ; la porte de la fédération a été fermée à clé, clé qui est maintenant au fond d'un lac.

Signal utilise les numéros de téléphone des utilisateurs comme identifiants. Cette pratique a certainement pour origine l'usage des SMS/MMS comme première méthode de transport des messages de Signal. En 2015, les développeurs ont cependant décidé arbitrairement d'arrêter de prendre en charge le transport par SMS/MMS, et d'utiliser, à la place, exclusivement les serveurs d'Open Whisper Systems, situés aux États-Unis. Cette décision a fait réagir une fraction de la communauté qui a créé un clone de l'application, désormais appelé Silence [sln].

Le choix d'Open Whisper Systems d'arrêter la prise en charge des SMS/MMS n'a cependant pas causé l'arrêt de l'usage des numéros de téléphone comme identifiants. Il en résulte un problème de vie privée, dû au fait que les métadonnées relatives aux expéditeurs et destinataires ne sont pas chiffrées par le protocole Signal. En effet, ces identifiants pourraient permettre aux opérateurs d'Open Whisper Systems de tracer le graphe social des utilisateurs, ou encore de les géolocaliser, grâce au réseau SS7 [ss7].

Le choix des numéros de téléphone comme identifiant est également fortement discutable depuis la publication de Signal Desktop. Ce logiciel permet aux utilisateurs de converser depuis des PC avec les utilisateurs Signal ; il n'est cependant pas possible d'employer ce logiciel sans s'être enrôlé préalablement sur téléphone portable !

Pour finir, cette centralisation du service présente également des risques topologiques. En effet, la géolocalisation aux États-Unis de la société Open Whisper Systems et de ses serveurs signifie qu'elle est soumise à l'arsenal légal américain. Celui-ci a d'ailleurs déjà été mis en œuvre à l'encontre de Signal [pac]. En outre, la centralisation facilite la censure administrative du service, comme cela s'est déjà produit plusieurs fois au Brésil pour Whatsapp [bra], et en Égypte pour Signal [egy]. Une technique expérimentale [dom], inspirée du module *meek* pour Tor et appelée *Domain Fronting*, a été déployée en réponse à ce dernier blocage. Son déploiement hâtif laisse cependant en suspens des questions de vie privée, de confidentialité et d'intégrité, eu égard à l'absence de protection de bout en bout de certaines (méta-)données transitant par Google AppEngine lors du *Domain Fronting*.

En comparaison, les JID de XMPP/OMEMO peuvent être des pseudonymes

n'offrant aucune corrélation avec un utilisateur particulier. Les utilisateurs les plus prudents peuvent même se connecter à XMPP au travers de Tor, ou utiliser des *hidden services* XMPP sur Tor. En outre, les identifiants ne sont pas liés à un type d'équipements. Ils peuvent ainsi être utilisés sur PC ou téléphone exclusivement ou sur un mélange des deux. Finalement, pour la géolocalisation des serveurs, l'infrastructure répartie de XMPP permet de jongler à volonté avec la localisation administrative et juridique des serveurs.

3.2 L'infrastructure de gestion de clés

Les serveurs gérés par Open Whisper Systems sont responsables du stockage des clés publiques de tous les utilisateurs, et de distribuer ces clés aux nouveaux utilisateurs. Ainsi, lorsqu'un nouvel utilisateur installe Signal, le logiciel prélève les numéros de téléphone de l'intégralité du carnet d'adresses de l'utilisateur, et les envoie aux serveurs de Signal, qui retournent en échange les clés des utilisateurs connus [lea]. Les utilisateurs figurant dans le carnet de contacts sont également notifiés qu'un de leurs contacts vient d'installer Signal.

En vue de protéger la vie privée des utilisateurs, les numéros de téléphone des contacts sont hachés avec une fonction cryptographique et le résultat est tronqué ; cette technique s'avère cependant insuffisante et une recherche exhaustive permet de recouvrer ces numéros de téléphone.

Pour chaque utilisateur du carnet d'adresses ayant Signal, les serveurs d'Open Whisper Systems retournent donc une clé à long terme, une clé à moyen terme et optionnellement une clé à usage unique. Cette méthode de distribution centralisée des clés exige de faire confiance aux serveurs. Ces derniers peuvent, en effet, dégrader sciemment la confidentialité persistante en ne retournant pas de clé à usage unique. Ils peuvent, en outre, tout simplement fournir de fausses clés, en vue d'effectuer une interception de messages. Cette éventualité peut être contrée si les utilisateurs effectuent une vérification des clés retournées et les valident. L'usage des clés préalable à cette vérification n'est cependant pas empêché, et nombre d'utilisateurs font donc probablement une confiance aveugle aux serveurs de Signal.

Les utilisateurs méfiants qui voudraient effectuer cette vérification ne sont malheureusement pas dotés d'outils adaptés. Ainsi, s'il était auparavant possible de vérifier la clé du destinataire d'un message, les développeurs de Signal ont dégradé cette possibilité en novembre 2016. Au nom d'études sur l'expérience utilisateur, ils ont ainsi remplacé la vérification d'empreintes cryptographiques des clés par la comparaison de « nombres de sûretés » (*safety number*) [saf], supposément plus faciles à comparer. Cette opération a ainsi réduit la sécurité de l'empreinte de 256 bits à 100 bits. Plus incompréhensible encore, l'empreinte a également été réduite dans le QRCode utilisé pour la comparaison des clés, alors qu'il ne peut y avoir d'impact sur l'expérience utilisateur, que l'on photographie un QRCode de 100 ou 256 bits de sécurité ! Pour finir, la vérification des empreintes est impossible lors d'une conversation de groupe [nos].

Pour enfoncer le dernier clou, Signal envisage de réviser à la baisse son mécanisme de sécurité concernant le signalement d'un changement de clés d'un pair [saf]. Auparavant, lorsqu'un utilisateur changeait de clé à long terme (une opération rarissime), une notification était affichée et une confirmation manuelle était exigée. Avec la nouvelle option, dont il est envisagé qu'elle soit activée par défaut, seule une petite ligne sera affichée au milieu de la conversation.

En comparaison, les clés OMEMO sont récupérées auprès d'un serveur au choix du destinataire. Avec le logiciel Conversations, par défaut depuis la version 1.15 de novembre 2016, les clés peuvent employées sans avoir été vérifiées ; un indicateur visuel différencie cependant les échanges avec les clés vérifiées et les échanges sans vérification [ctr]. De plus, dès qu'une première clé d'un utilisateur est vérifiée, seules ses clés vérifiées sont utilisables. Dans le cas où plusieurs périphériques seraient destinataires, chaque clé doit être individuellement validée au premier usage, à l'aide d'une empreinte cryptographique de 256 bits. Il existe un risque d'utiliser plusieurs fois une clé à usage unique, mais le protocole prévoit une contre-mesure pour raccourcir la durée de l'incident.

3.3 Usage des clés symétriques

Signal chiffre les messages à l'aide d'AES256 en mode CBC et ses motifs d'intégrité sont calculés avec HMAC-SHA256 dont le résultat est tronqué à 64 bits.

Ces choix peuvent faire débat. Ainsi, bien que Signal use d'une composition chiffrement/intégrité satisfaisante (*Encrypt-then-Mac*), l'implémentation incorrecte du mode CBC s'est révélée à l'origine de nombreuses vulnérabilités au fil des années. Cela a été notamment le cas dans TLS. En conséquence, l'existence de meilleures alternatives, tant en performances qu'en sécurité, a valu à ce mode d'être déconseillé par les auteurs du RFC de HTTP/2 [h2]. La prochaine version de TLS ne la prend même tout simplement pas en charge [tls].

En outre, si l'algorithme HMAC-SHA256 est, à ce jour, irréprochable, la troncature du motif d'intégrité à 64 bits affaiblit son efficacité de manière significative. Ce choix tient peut-être sa justification dans l'usage des SMS comme méthode originelle de transport des messages. À l'heure actuelle, tous les messages de Signal sont notifiés à l'aide de *Firebase Cloud Messaging* (anciennement *Google Cloud Messaging*), et transportés sur le canal de données des téléphones portables. Les problèmes de bande passante utile ne peuvent donc plus constituer une justification. En fait, la grande bande passante désormais disponible joue même à l'encontre de cette troncature, rendant plus facile le bombardement de messages frauduleux jusqu'à réussir à forger un motif correct.

En comparaison, OMEMO emploie AES256 avec le mode de chiffrement authentifié GCM, considéré à l'état de l'art.

3.4 Documentation et audits

Signal est une cible mouvante. Jusqu'alors dénué de documentation, par une volonté affichée de ses développeurs, ce n'est que sur la pression croissante de la communauté que les algorithmes XEdDSA, X3DH et Double Ratchet ont été finalement spécifiés et publiés dans le domaine public. Si des études formelles vont désormais pouvoir être menées sur ces trois protocoles, effectuer ce même type d'études sur le protocole Signal reste encore un défi. Quelques-uns s'y sont néanmoins essayés, documentant leurs découvertes du protocole par « ingénierie inverse du code source », afin de dégager des propriétés de sécurité [etua, etub]. Aucune attaque réellement significative n'a été découverte, à ce jour.

En comparaison, le protocole OMEMO est documenté et standardisé dans une extension expérimentale du protocole XMPP. Il a récemment subi un audit, ayant révélé divers points d'attention [aud], qui ont été rapidement pris en

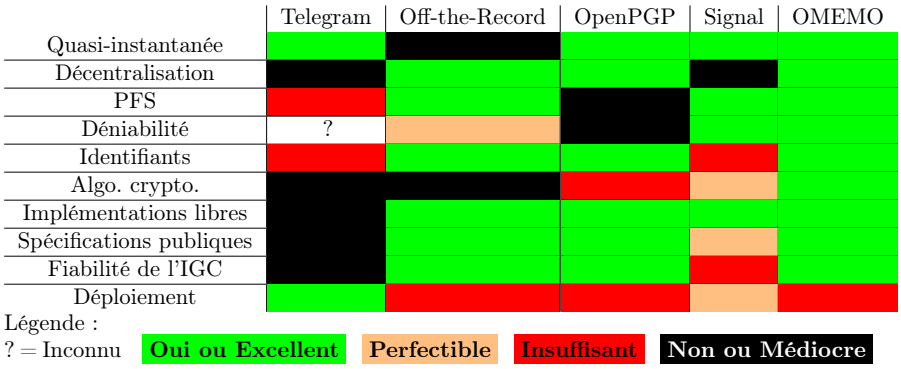

FIGURE 4 – Synthèse des points forts et faibles de plusieurs protocoles dans le cadre de la messagerie (quasi) instantanée. Les justifications ou références ont été apportées dans l'article.

compte par le protocole et au moins certaines implémentations.

4 Conclusion

Cet article a détaillé les atouts et inconvénients de plusieurs protocoles permettant la sécurisation de messageries quasi instantanées. La figure 4 reprend les observations de manière synthétique.

Les utilisateurs souhaitant protéger leur messagerie quasi instantanée de bout en bout disposent d'options valables, comme Signal, Whatsapp, ou encore OMEMO. Ces outils ont en commun un cœur cryptographique formé des protocoles X3DH et Double Ratchet, dont il ressort de plusieurs études indépendantes qu'ils seraient fiables.

Malgré cela, ces différentes solutions offrent un niveau de protection de la vie privée et de la confidentialité des messages qui varie de manière significative. Ainsi, cet article a rappelé, à l'instar d'une conférence lors de la conférence 33c3 [ccc], que les numéros de téléphone sont à la fois des identifiants de compte pratiques, puisque déjà enregistrés dans le téléphone, mais aussi une donnée personnelle sensible. Outre leur usage éventuel pour géolocaliser des utilisateurs, ils sont nécessairement transmis en clair, en tant que métadonnées de tout message : une situation préoccupante lorsque les messages du réseau doivent passer par une infrastructure centralisée. Cette dernière est, en effet, en mesure d'observer le graph social de ses utilisateurs et la fréquence de leurs échanges, quand bien même leurs concepteurs s'en défendent [pac]. Par ailleurs, comme cet article l'a présenté, les serveurs de Signal et Whatsapp sont en charge de la délivrance des clés publiques des contacts d'un utilisateur. Pour cela, un dérivé cryptographique des numéros de tous les contacts d'un utilisateur est envoyé aux serveurs, qui répondent avec des clés publiques associées. Cette dérivation cryptographique est hélas aisément réversible, et il est possible de retrouver la liste des contacts d'un utilisateur de ces applications. En outre, il est à la charge des utilisateurs de vérifier l'authenticité des clés remises par les serveurs, une étape probablement rarement effectuée et dont les mécanismes de vérification ont été récemment dégradés dans Signal, parfois de façon inexplicable. Pour What-

sapp, ce mécanisme de distribution de clés a même été à l'origine d'un tumulte, en janvier 2017, lorsque le journal Guardian a rapporté qu'une vulnérabilité publique depuis huit mois et non corrigée permet l'interception de messagesen clair [gua]. Finalement, cet article a détaillé le protocole OMEMO, qui utilise le réseau XMPP pour la distribution des clés et des messages. Le réseau XMPP utilise des serveurs répartis et des identifiants indépendants de l'identité propre de l'utilisateur. Chaque utilisateur de XMPP est libre d'employer le serveur de son choix, dont la sécurité peut être catastrophique ou excellente. Une sécurité serveur excellente n'exempte cependant pas les utilisateurs de la nécessité de vérifier les clés cryptographiques.

Heureusement, grâce à la publication récente dans le domaine public des spécifications de X3DH et de Double Ratchet par les auteurs de Signal, de nombreuses applications peuvent s'équiper ce coeur cryptographique robuste tout en faisant des choix d'infrastructures plus respectueux de la vie privée que ne le sont Signal ou Whatsapp.

Les arguments de l'éditeur de Signal concernant l'agilité d'un écosystème fermé, soumis aux décisions unilatérales des développeurs, sont certainement fondés. À l'instar du régime politique démocratique, les réseaux fédérés, comme XMPP, nécessitent des négociations, des ententes et des compromis. Le résultat peut cependant, au long cours, se montrer supérieur à la somme des idées exprimées par les différents intervenants de l'écosystème.

Ainsi, OMEMO, avec la stabilité de sa spécification, sa licence ouverte, ses primitives cryptographiques à l'état de l'art et son architecture répartie, offre aux utilisateurs de XMPP une méthode de communication sécurisée satisfaisante, et potentiellement durable.

> Pour le lecteur intéressé, l'application libre Conversations implémente OMEMO et des extensions permettant l'économie de la batterie du périphérique le faisant tourner [con]. Un compte est optionnellement fourni à tout utilisateur faisant l'acquisition de l'application au travers du Google Play Store. Les utilisateurs d'iOS peuvent utiliser ChatSecure [chs]. Les utilisateurs PC peuvent, quant à eux, se tourner vers Gajim et son implémentation expérimentale d'OMEMO. Pour les utilisateurs souhaitant effectuer un autohébergement de leur serveur XMPP, Prosody [pro] implémente la partie serveur des optimisations permettant des économies de batterie. Enfin, la Quadrature du Net fournit un service XMPP ouvert à tous [lqd].

5 Remerciements

Je tiens à remercier mes relecteurs : Piotr Chmielnicki, François Contat, Arnaud Ebalard, Sarah De Haro, Olivier Levillain, Mickaël Salaün, et Guillaume Valadon. Les idées exprimées dans cet article ne sauraient les engager.

6 Références

Références

[aud] https://conversations.im/omemo/audit.pdf.

[bra] https://techcrunch.com/2016/07/19/whatsapp-blocked-in-brazil-again/.

[ccc] https://media.ccc.de/v/33c3-8062-a_look_into_the_mobile_messaging_black_box.

[chs] https://chatsecure.org/blog/chatsecure-v4-released/.

[con] https://conversations.im.

[ctr] https://gultsch.de/trust.html.

[dom] https://www.bamsoftware.com/papers/fronting/.

[dr] https://whispersystems.org/docs/specifications/doubleratchet/.

[egy] https://whispersystems.org/blog/doodles-stickers-censorship/.

[etua] https://eprint.iacr.org/2014/904.pdf.

[etub] https://eprint.iacr.org/2016/1013.pdf.

[for] https://www.ssi.gouv.fr/uploads/2015/05/format-Oracles-on-OpenPGP.pdf.

[gua] https://www.theguardian.com/technology/2017/jan/13/whatsapp-backdoor-allows-snooping-on-encrypted-messages.

[h2] https://www.rfc-editor.org/rfc/rfc7540.txt.

[hkd] https://www.rfc-editor.org/rfc/rfc5869.txt.

[ims] https://isi.jhu.edu/~mgreen/imessage.pdf.

[lea] https://whispersystems.org/blog/contact-discovery/.

[leg] https://moderncrypto.org/mail-archive/messaging/2016/002275.html.

[lib] https://github.com/WhisperSystems/libsignal-protocol-java.

[lqd] https://jabber.lqdn.fr.

[mpo] https://www.cypherpunks.ca/~iang/pubs/mpotr.pdf.

[nof] https://whispersystems.org/blog/goodbye-encrypted-sms/.

[nok] https://github.com/LibreSignal/LibreSignal/issues/37.

[nos] https://support.whispersystems.org/hc/en-us/articles/213134107-How-do-I-verify-the-person-I-m-sending-messages-to-is-who-they-say-they-a

[ome] https://xmpp.org/extensions/xep-0384.html.

[opg] https://www.rfc-editor.org/rfc/rfc4880.txt.

[otr] https://otr.cypherpunks.ca/.

[pac] https://whispersystems.org/bigbrother/eastern-virginia-grand-jury/.

[pro] https://prosody.im.

[saf] https://whispersystems.org/blog/safety-number-updates/.

[sln] https://silence.im/.
[ss7] https://www.youtube.com/watch?v=lQ0I5tl0YLY.
[tela] https://news.ycombinator.com/item?id=6913456.
[telb] https://cs.au.dk/~jakjak/master-thesis.pdf.
[tls] https://tools.ietf.org/html/draft-ietf-tls-tls13-18.
[x3d] https://whispersystems.org/docs/specifications/x3dh/.
[xds] https://whispersystems.org/docs/specifications/xeddsa/.