Das Signal-Protokoll

Ausarbeitung zur Vorlesung Kryptographie

Marius Wagner
Universität Leipzig
mw13voqo@studserv.uni-leipzig.de

14. Juni 2017

1 Einleitung

In den letzten Jahren ist das Bewusstsein für den Schutz der eigenen Privatsphäre sowie das Verlangen nach einer sicheren elektronischen Kommunikation gestiegen. Eine aktuell zum Einsatz kommende Technik ist das Signal-Protokoll. In dieser Ausarbeitung soll ein Überblick über die Entstehung, die Komponenten des Protokolls sowie deren Anwendung gegeben werden.

2 Geschichte

Die erste Version des Signal-Protokolls wurde 2013 von Trevor Perrin und Moxin Marlinspike entwickelt und hatte noch keinen eigenen Namen. Es kam zuerst in der Anwendung TextSecure des Unternehmens Open Whisper Systems zum Einsatz. Als Grundlage dient das Off-the-record-Protokoll (OTR), das vor allem bei der Kommunikation mittels XMPP eingesetzt wurde.

Im folgenden Jahr erschien die zweite Version, die den Namen Axolotl-Protokoll erhielt, eine Anspielung auf die starke Selbstheilungsfähigkeit des gleichnamigen Lebewesens. Diese Version erlaubte im Gegensatz zu OTR auch asynchronen Nachrichtenaustausch, das heißt es müssen nicht beide Kommunikationspartner gleichzeitig online sein.

Die dritte Version bestand im Wesentlichen aus einer Überarbeitung der kryptographischen Grundlagen und der Regelung des Ablaufs von Übertragungen.

Im Jahr 2015 wurde die Entwicklung von TextSecure beendet und in die Anwendung Signal überführt, deren Entwicklung schon einige Zeit vorher begann. Damit einher ging auch die Umbenennung in Signal-Protokoll aus Gründen der Vereinheitlichung.

Die aktuelle Version des Protokolls stammt vom Oktober 2016.

3 Komponenten

Das Protokoll setzt sich aus drei individuellen Komponenten zusammen. Als Erstes ein Verfahren zur Erzeugung von Signaturen [Per16b], eine angepasste Variante des Diffie-Hellmann-Schlüsselaustausches [MP16] und ein Algorithmus zum sicheren Austausch von Nachrichten [PM16]. Das eigentliche Zusammenspiel der einzelnen Bestandteile sowie ein Konzept zur Sitzungsverwaltung ist durch das Sesame-Protokoll festgehalten [Per16a]. Nachfolgend werden die einzelnen Bestandteile beschrieben.

3.1 Signaturerzeugung

Für die Erzeugung des Hauptschlüssels und dessen Austausch ist es notwendig, auf Basis einer Nachricht und eines öffentlichen Schlüssels eine Signatur zu erzeugen und verifizieren zu können[MP16, 2]. Als Verfahren dafür wird XEdDSA oder die erweiterte Variante VXEdDSA definiert.

Diese Verfahren können mit einer der beiden Diffie-Hellmann-Funktionen X25519 oder X448 [LHT16] verwendet werden. Diese sind so entworfen, dass sie immun gegenüber üblichen Rechenzeitangriffen sind [LHT16, 10].

Um die elliptischen Kurven für (V)XEdDSA zu verwenden, sind bestimmte Parameter definiert[Per16b, 3].

Weiterhin benötigen die Verfahren eine Hash-Funktion, dabei ist SHA-512 als Standard empfohlen [Per16b, 5].

Damit bietet XEdDSA die Möglichkeit, Nachrichten zu signieren und zu überprüfen, ob eine Signatur tatsächlich zu einer bestimmten Nachricht gehört [Per16b, 7]. VXEdDSA unterscheidet sich insofern von XEdDSA, als es neben der Signatur noch eine Beweisausgabe erzeugt wird. Mit diesem Beweis kann und dem öffentlichen Schlüssel kann verifiziert werden, ob die Signatur korrekt erzeugt wurde, ohne dabei den privaten Schlüssel zu kompromittieren. [Per16b, 8]

3.2 Schlüsselerzeugung

Damit der eigentliche Austausch von Nachrichten mit Hilfe des Signal-Protokolls stattfinden kann, müssen sich die beiden Teilnehmer auf einen gemeinsamen initialen Privatschlüssel einigen. Dafür definiert das Protokoll das X3DH-Verfahren („Extended Triple Diffie-Hellman“)[MP16]. Dieses verfügt über perfekte vorwärts gerichtete Geheimhaltung, das heißt selbst wenn ein Angreifer („Mallory“) Zugriff auf den Schlüssel erhält, wird es

ihm nicht möglich sein, alle Nachrichten zu entschlüsseln, da die Schlüssel für die einzelnen Nachrichten aufeinander aufbauen, jedoch nach Verwendung immer wieder so schnell wie möglich gelöscht werden. Eine weitere Eigenschaft ist die kryptographische Bestreitbarkeit.

Damit Anwendungen Gebrauch von X3DH machen können, müssen sie sich für eine der beiden Diffie-Hellmann-Funktionen X255519 oder X448 entscheiden, einen Hash-Algorithmus mit Ausgabelänge von 256 bzw. 512 auswählen, sowie eine Zeichenkette als Identifikator der Anwendung bereitstellen [MP16, 2].

X3DH erlaubt asynchrone Kommunikation, allerdings ist es dafür notwendig, dass ein Benutzer („Bob“) gewisse Informationen veröffentlicht haben muss, damit ein anderer Benutzer („Alice“) mit ihm einen gemeinsamen Privatschlüssel aushandeln und verschlüsselte Informationen an ihn senden kann, während Bob offline ist[MP16, 2]. Dafür bietet es sich an, einen Server zu verwenden, auf dem die Informationen der einzelnen Benutzer hinterlegt sind und außerdem Nachrichten für den Empfänger zwischengespeichert werden, bis dieser wieder online ist und sie abrufen kann [MP16, 3].

Die notwendigen Informationen von Bob umfassen:

- Identitätsschlüssel IK_B
- signierter PreKey SPK_B
- Signatur des PreKey $Sig(IK_B, Encode(SPK_B))$
- Menge von einmal verwendbaren PreKeys $({OPK_B}^1, {OPK_B}^2, ...)$

Damit Alice damit beginnen kann Nachrichten an Bob zu senden, werden zuerst die Informationen des Adressaten vom Server geholt. Damit kann eine Überprüfung der Signatur erfolgen und bei einem Scheitern kommt es zum Abbruch. Anschließend werden die folgenden Berechnungen durchgeführt:

- $DH1 = DH(IK_A, SPK_B)$
- $DH2 = DH(EK_A, IK_B)$
- $DH3 = DH(EK_A, SPK_B)$
- $DH4 = DH(EK_A, OPK_B)$
- $SK = KDF(DH1||DH2||DH3||DH4))$

Beim berechneten SK handelt es sich um den gemeinsamen Geheimschlüssel. Weiterhin generiert Alice aus den beiden Identitätsschlüsseln eine Byte-Sequenz „AD“. Damit kann nun eine erste Nachricht mit den folgenden Bestandteilen versendet werden:

- IK_A
- EK_A
- Identifikator des verwendeten einmaligen PreKey
- Geheimtext, der per AEAD-Schema verschlüsselt wurde

AEAD bedeutet Authenticated Encryption with Associated Data. Dabei handelt es sich um ein Verschlüsselungsverfahren, das neben der Vertraulichkeit des verschlüsselten Textes auch dessen Integrität und Authentizität gewährleistet. Dies wird realisiert, indem zum Beispiel nicht nur der Klartext verschlüsselt wird, sondern außerdem ein Hash, der sogenannte Nachrichtenauthentifizierungscode, davon erzeugt wird und beide übertragen werden. [McG08]

Nachdem die Nachricht von Bob empfangen wurde, führt er die gleichen Diffie-Hellman-Berechnungen wie Alice durch, nur mit vertauschten Schlüsseln, um somit auch den Geheimschlüssel zu erzeugen. Da er ebenfalls über die Identitätsschlüssel verfügt, erzeugt auch er die Byte-Sequenz „AD“. Mit diesen beiden Werten versucht er, den Geheimtext zu entschlüsseln. Sollte dies scheitern, bricht Bob den Vorgang ab. Wenn Bob erfolgreich war, sind nun beide im Besitz des gemeinsamen Geheimschlüssels.

Damit können nun weitere Nachrichten ausgetauscht werden oder der Schlüssel dient als Ausgangspunkt für den im Folgenden beschriebenen Double-Ratchet-Algorithmus. [MP16]

3.3 Nachrichtenaustausch

Der eigentliche Austausch von Nachrichten erfolgt nach dem Double-Ratchet-Algorithmus. Das grundlegende Konzept für diesen Algorithmus ist eine sogenannte KDF-Kette. Dabei erzeugt eine Key-Derivation-Funktion (KDF) mit Hilfe eines geheimen und zufälligen Schlüssels zusammen mit Eingabedaten mehrere Ausgabewerte. Eines der Ergebnisse ist ein weiterer Schlüssel, der als erneuter Eingabewert für die KDF dient und somit eine Kette bildet. [PM16, 3]

Der Double-Ratchet-Algorithmus verwendet drei KDF-Ketten. Eine Wurzelkette, die mit Hilfe von Diffie-Hellmann-Berechnungen neue Ausgabeschlüssel erzeugt. Die so erhaltenen Schlüssel werden als KDF-Schlüssel für die Nachrichtenketten, also einer Sende- und einer Empfangs-Kette, verwendet. Diese beiden verwenden einen symmetrischen Schlüssel für die notwendigen Berechnungen. [PM16, 5]

Mit Hilfe der Nachrichtenketten wird für jede einzelne Nachricht ein Schlüssel erzeugt, um diese damit zu verschlüsseln. Diese Kombination der Wurzel-Kette mit ihren Ausgaben als Eingabewerte für die Nachrichtenkette wird als Double-Rachet bezeichnet.

Das Double-Ratchet-Verfahren bietet auch Unterstützung, um Nachrichten nicht in der korrekten chronologischen Reihenfolge zu empfangen. Dazu enthält der Nachrichtenkopf Informationen, wie viele Nachrichten die vorherige Sendekette enthält und an welcher Position die Nachricht in der aktuellen Kette steht. Damit lässt sich beim Erhält einer Nachricht ermitteln, wie viele Nachrichten seit dem letzten Erhalt noch nicht übermittelt wurden, und der Empfänger kann die Schlüssel für diese bis zum Empfang aufbewahren.

Eine erweiterte Variante des Algorithmus ermöglicht es, zusätzlich auch den Nachrichtenkopf verschlüsselt zu übertragen. Dazu verwenden beide Teilnehmer einen symmetrischen Schlüssel für die aktuelle Nachrichtenkette und halten einen bereit für die nächste Nachrichtenkette. Diese beiden Schlüssel für den Nachrichtenkopf werden ähnlich wie die Schlüssel für die Nachrichtenkette aus der Wurzelkette erzeugt.

3.4 Sitzungsverwaltung

Die letzte Komponente des Signal-Protokolls ist der Sesame-Algorithmus, der die Sitzungsverwaltung regelt. Dies ist notwendig, damit ein asynchroner Nachrichtenaustausch funktioniert und Teilnehmer mehrere Geräte verwenden können. [Per16a, 2]

Der Algorithmus setzt einen Server voraus, auf dem sich die Daten der Teilnehmer befinden und der auch die Nachrichten aufbewahren kann, bis diese von den Teilnehmern empfangen werden. Ebenso gibt es Benutzer, die über eine eindeutige Identifikation verfügen. Diese Benutzer können eine beliebige Anzahl von Geräten besitzen, die sich ebenfalls eindeutig identifizieren lassen. Auf dem Server befindet sich für jedes Gerät ein Briefkasten, in dem die Nachrichten bis zur Auslieferung aufbewahrt werden. Geräte können Nachrichten auch an andere Briefkästen senden, um den Nachrichtenverlauf zu synchronisieren. Den letzten Bestandteil bilden die Sitzungen, die ebenfalls über eine eindeutige Identifikation verfügen. Eine verschlüsselte Nachricht kann nur mit der korrekten Sitzung entschlüsselt werden. [Per16a, 3-4]

Damit Nachrichten ausgetauscht werden können, sind auf den Geräten die Informationen der Zielteilnehmer zusammen mit deren Geräteinformationen aufbewahrt. Geräteinformationen können auch Sitzungen enthalten. Damit kann dann eine Nachricht für die entsprechende Sitzung verschlüsselt und an den Server gesendet werden. Empfängt ein Teilnehmer eine Nachricht, so erhält er auch die Information, von welchem Benutzer und Gerät die Nachricht stammt. Damit kann die korrekte Sitzung zum Entschlüsseln der Nachricht ausgewählt werden. [Per16a, 6-9]

4 Anwendung

Verwendung fand das Signal-Protokoll erstmalig 2013 in der Anwendung TextSecure, die genau wie das Protokoll auch von Open Whisper Systems entwickelt wurde.

Im April 2016 wurde der Nachrichtenaustausch der Anwendung WhatsApp auf das Signal-Protokoll umgestellt. [Lom16] Weitere Anwendungen, bei denen das Protokoll teilweise auch in angepasster Form zum Einsatz kommt, sind zum Beispiel Viber und Wire. Im Dezember wurde eine Erweiterung unter der Bezeichnung Omemo für XMPP, die auf dem Double-Ratchet-Algorithmus basiert, durch die XMPP Standards Foundation genehmigt. [Fou16]

Der Entwickler Open Whisper Systems stellt Implementierungen des Signal-Protokolls als Bibliotheken für verschiedene Programmiersprachen zur Verfügung. [Sys17]

5 Sicherheitsbedenken

Auch wenn mit dem Signal-Protokoll versucht wurde, ein möglichst sicheres Verfahren für den Datenaustausch zu entwerfen, so existieren doch einige potentielle Schwachstellen.

Bei der Erzeugung von Signaturen mittels XEdDSA oder VXEdDSA muss als Parameter eine zufällige Bytefolge mit übergeben werden. Dabei ist darauf zu achten, dass bei jeder Verwendung eine neue Bytefolge zum Einsatz kommt, da es sonst möglich ist, Rückschlüsse auf den privaten Schlüssel zu ziehen [Per16b, 11].

Jeder Teilnehmer besitzt einen öffentlichen Identifizierungsschlüssel, mit dessen Hilfen sich die beiden Parteien authentifizieren sollten, ansonsten kann die kryptographische Sicherheit der nachfolgenden Kommunikation nicht sichergestellt werden [MP16, 7]. Weiterreichende Probleme können entstehen, wenn die privaten Schlüssel eines Teilnehmers kompromittiert werden. Dies gibt die Möglichkeit zur Nachahmung der Identität und dem boshaften Erzeugen von Nachrichtensitzungen mit anderen Teilnehmern. Dies kann teilweise durch die Verwendung von signierten PreKeys unterbunden werden. Allerdings verhindert der Einsatz des Double-Ratchet-Algorithmus teilweise die Rekonstruktion bereits ausgetauschter Nachrichten. [MP16, 9]

6 Fazit

Das Signal-Protokoll bietet mit seinen einzelnen Komponenten eine verständliche Möglichkeit, um sichere Kommunikation zu gewährleisten. Mittlerweile ist es ein gern angewendetes Verfahren, was man auch daran sieht, dass auch außerhalb des Programms Signal zum Einsatz kommt und als Inspiration für weitere Verfahren wie zum Beispiel Omemo diente. Diese Verbreitung wurde auch ermöglicht durch die Bereitstellung von quelloffenen Bibliotheken für Entwickler in verschiedenen Sprachen. Dennoch hat auch dieses Protokoll etwaige problematische Bereiche, über die man sich bewusst sein sollte und die bei einem entsprechenden Einsatz berücksichtigen werden sollten.

Literatur

[Fou16] XMPP Standards Foundation. Xep-0384. omemo encryption. `https://xmpp.org/extensions/xep-0384.html`, 2016.

[LHT16] A. Langley, M. Hamburg, and S. Turner. Elliptic Curves for Security. RFC 7748 (Informational), January 2016.

[Lom16] Natasha Lomas. Whatsapp completes end-to-end encryption rollout. `https://techcrunch.com/2016/04/05/whatsapp-completes-end-to-end-encryption-rollout/`, 2016.

[McG08] D. McGrew. An Interface and Algorithms for Authenticated Encryption. RFC 5116 (Proposed Standard), January 2008.

[MP16] Moxie Marlinspike and Trevor Perring. The x3dh key agreement protocol. 2016.

[Per16a] Moxie Marlinspike Trevor Perring. The sesame algorithm. session management for asynchronous message encryption. 2016.

[Per16b] Trevor Perring. The xeddsa and xedddsa signature schemes. 2016.

[PM16] Trevor Perring and Moxie Marlinspike. The double ratchet algorithm. 2016.

[Sys17] Open Whisper Systems. Open whisper system repositories. `https://github.com/WhisperSystems`, 2017.