SelfLinux-0.13.1

Konzepte

Autor: Mike Ashley ()
Formatierung: Matthias Hagedorn (*matthias.hagedorn@selflinux.org*)
Lizenz: GFDL

GnuPG verwendet mehrere kryptographische Verfahren wie beispielsweise **symmetrische Verschlüsselung**, **Public-Key-Verschlüsselung** und **Einweg-Hashing**. Natürlich können Sie GnuPG auch ohne tiefere Kenntnis dieser Konzepte benutzen, doch wenn Sie GnuPG effektiv einsetzen möchten, sollten Sie ein wenig Hintergrundwissen haben.

Dieses Kapitel führt in die grundlegenden kryptographischen Konzepte ein, wie sie von GnuPG benutzt werden. Andere Bücher behandeln diese Themen viel detaillierter. Empfehlenswerte Bücher zum tieferen Studium sind beispielsweise Bruce Schneiers "Angewandte Kryptographie" oder Reinhard Wobsts "Abenteuer Kryptologie". Weitere Literaturhinweise finden sich im Anhang B.

Inhaltsverzeichnis

1 Symmetrische Verschlüsselung

2 Public-Key-Verschlüsselung

3 Hybride Verschlüsselungsverfahren

4 Digitale Unterschriften

5 Fußnoten

1 Symmetrische Verschlüsselung

Eine symmetrische Verschlüsselung benutzt zum Ver- und Entschlüsseln denselben Schlüssel. Zwei Korrespondenzpartner, die eine symmetrische Verschlüsselung benutzen, müssen sich vorher über den Schlüssel einigen. Mit diesem Schlüssel verschlüsselt der Absender die Nachricht und schickt sie an den Empfänger, der sie unter Benutzung desselben Schlüssels wiederherstellt. Nach diesem Prinzip funktionierte beispielsweise die deutsche `Enigma`. Die jeweiligen Tages-Schlüssel wurden als Code-Bücher ausgegeben, und jeden Tag konsultierte dann ein Funker seine Kopie des Code-Buchs, um den aktuellen Tagesschlüssel zu ermitteln, mit dem der Funkverkehr für den betreffenden Tag dann ver- und entschlüsselt wurde. Zu den modernen Beispielen für symmetrische Verschlüsselungen gehören z.B. Blowfish und IDEA.

Ein gutes Verschlüsselungverfahren legt den Schwerpunkt der Sicherheit auf die Geheimhaltung des Schlüssels und nicht auf die Geheimhaltung des verwendeten Algorithmus. Mit anderen Worten, es ist keine Hilfe für einen Angreifer, wenn das Verschlüsselungsverfahren bekannt ist, solange er nicht im Besitz des Schlüssels selbst ist. Die von GnuPG benutzten Verschlüsselungsverfahren beruhen auf diesen Prinzipien.

Da die gesamte Sicherheit auf dem Schlüssel beruht, ist es wichtig, dass der Schlüssel mit verfügbaren Mitteln nicht zu erraten ist. Daraus folgt, dass der Vorrat an möglichen Schlüsseln, der sogenannte **key space**, möglichst groß sein muß. Während seiner Zeit in Los Alamos war der Nobelpreisträger Richard Feynman berühmt für seine Fähigkeit, Safes zu knacken. Um es noch geheimnisvoller zu machen, schleppte er einen Satz von Werkzeugen mit, zu denen ein altes Stethoskop gehörte. In Wirklichkeit wandte er jedoch eine ganze Reihe von Tricks an, um die Zahl der Kombinationen, die er ausprobieren mußte, zu reduzieren; dann fing er an zu raten, bis er die richtige Kombination fand. Mit anderen Worten, er verringerte die Größe des **key space**.

Die Briten benutzten im 2. Weltkrieg Maschinen, um Schlüssel zu erraten. Die deutsche `Enigma` hatte einen sehr großen **key space**, doch die Briten bauten spezialisierte Rechenmaschinen, `Bombes` genannt, um systematisch alle Schlüssel auszuprobieren, bis der jeweilige Tagesschlüssel gefunden war. Manchmal fanden sie den Tagesschlüssel innerhalb der Benutzungsdauer des neuen Schlüssels, an manchen Tagen fanden sie den richtigen Schlüssel überhaupt nicht.

Heute können Computer sehr schnell Schlüssel erraten, und eben deshalb ist in modernen Verschlüsselungsverfahren die Schlüsselgröße äußerst wichtig. Die DES-Verschlüsselung zum Beispiel benutzt einen 56-Bit-Schlüssel; das bedeutet, dass es 256, also genau 72.057.594.037.927.936 mögliche Schlüssel gibt (das sind **mehr als 72 Billiarden**). Obwohl das eine sehr große Zahl ist, kann ein normaler Mehrzweckcomputer den gesamten **key space** innerhalb von Tagen prüfen. Ein spezialisierter Computer braucht hierfür möglicherweise nur ein paar Stunden. Die moderneren Verschlüsselungsverfahren wie beispielsweise Blowfish und IDEA benutzen sämtlich 128-Bit-Schlüssel, was bedeutet, dass es 2128 (340.282.366.920.938.463.463.374.607.431.768.211.456!!!) mögliche Schlüssel gibt. Dies sind so unglaublich viel mehr Kombinationen als bei einer 56-Bit-Verschlüsselung, dass sogar selbst dann, wenn man alle Computer der Welt zusammen arbeiten ließe, das bisherige Alter des Universums noch eine zu kurze Zeit sein könnte, um den richtigen Schlüssel zu finden.

2 Public-Key-Verschlüsselung

Das Hauptproblem bei symmetrischen Verschlüsselungen ist nicht die Sicherheit der eingesetzten Verfahren, sondern der Austausch der Schlüssel. Wenn zwei Kommunikationspartner einmal die Schlüssel ausgetauscht haben, kann der betreffende Schlüssel für sicheren Datenaustausch benutzt werden. Die Frage ist nur, auf welchem sicheren Wege der Schlüsselaustausch stattgefunden hat. Wahrscheinlich wäre es für einen Angreifer viel leichter, den Schlüssel abzufangen, als alle möglichen Schlüssel im **key space** auszuprobieren (eine Erfahrung, die die Deutschen mit ihrer Enigma auch machen mußten). Ein weiteres Problem ist die Anzahl der insgesamt benutzten Schlüssel. Wenn die Zahl der Leute, die miteinander kommunizieren wollen, **n** beträgt, so werden insgesamt **n(n-1)/2** Schlüssel (also beispielsweise 45 Schlüssel bei 10 Leuten) benötigt. Dies mag für eine geringe Personenzahl noch angehen, läßt sich aber bei großen Personengruppen nicht mehr handhaben.

Der Sinn von Verschlüsselungsverfahren mit öffentlichem Schlüssel besteht darin, dass das Sicherheitsrisiko beim gegenseitigen Schlüsselaustausch gänzlich vermieden wird. Jeder hat ein Schlüsselpaar mit einem **öffentlichen** und einem **geheimen** Schlüssel. Zum Verschlüsseln einer Nachricht benutzt man den öffentlichen Schlüssel des Empfängers, und nur dieser kann sie mit seinem geheimen Schlüssel wieder entschlüsseln.

Dadurch löst man das Problem des Schlüsselaustausches bei symmetrischer Verschlüsselung. Sender und Empfänger brauchen sich nicht auf einen Schlüssel zu einigen. Erforderlich ist nur, daß der Absender eine Kopie des öffentlichen Schlüssels des Empfängers besitzt. Dieser eine öffentliche Schlüssel kann von jedem benutzt werden, der mit dem Empfänger kommunizieren will. Somit sind dann insgesamt nur **n** Schlüsselpaare notwendig, wenn **n** Leute verschlüsselt miteinander kommunizieren wollen.

Die Verschlüsselung mit öffentlichem Schlüssel beruht auf sogenannten Falltür-Algorithmen bzw. Einweg-Hashes. Das sind Funktionen, die leicht zu berechnen sind, doch umgekehrt ist es praktisch unmöglich, aus dem Ergebnis dieser Hash-Funktionen wieder den Ausgangswert zu berechnen. So ist es z.B. leicht, zwei Primzahlen miteinander zu multiplizieren, um eine Nichtprimzahl zu erhalten, es ist aber schwer, eine Nichtprimzahl in ihre Primfaktoren zu zerlegen. Falltür-Algorithmen sind ähnlich, haben aber eine **Falltür**. Das heißt: Wenn ein Stück Information bekannt ist, kann man leicht die Umkehrfunktion berechnen. Wenn Sie z.B. eine aus zwei Primfaktoren bestehende Zahl haben, so macht die Kenntnis eines der Faktoren es leicht, den zweiten zu berechnen.

Angenommen, ein Verfahren beruhe auf der Bildung einer Zahl aus Primfaktoren, dann enthält der öffentliche Schlüssel eine aus zwei großen Primfaktoren zusammengesetzte Zahl, und das Verschlüsselungsverfahren benutzt dann diese Nichtprimzahl zum Verschlüsseln der Nachricht. Das Verfahren zum Wiederherstellen dieser Nachricht erfordert dann die Kenntnis der Primfaktoren. So ist die Entschlüsselung möglich, wenn Sie den privaten Schlüssel haben, der einen der Faktoren enthält, ist aber praktisch unmöglich, wenn Sie ihn nicht haben.

Wie bei guten symmetrischen Verschlüsselungsverfahren beruht die Sicherheit auch bei Public-Key-Verfahren ausschließlich auf dem Schlüssel. Aus diesem Grund kann man die Schlüsselgröße als ein Maß für die Sicherheit des Systems nehmen. Allerdings kann man die Größe eines symmetrischen Schlüssels nicht mit der von Public-Key-Verfahren vergleichen, um Rückschlüsse auf deren relative Sicherheit ziehen zu können. Bei einem Brute-Force-Angriff auf eine symmetrische Verschlüsselung mit einer Schlüsselgröße von 80 Bit muß der Angreifer bis zu 280-1 Schlüssel ausprobieren, um den richtigen Schlüssel zu finden. Bei einem Brute-Force-Angriff auf eine Public-Key-Verschlüsselung muß der Angreifer bei einer Schlüsselgröße von 512 Bit eine in 512 Bit codierte (bis zu 155 Dezimalstellen umfassende) Nichtprimzahl in ihre Primfaktoren zerlegen. Der Rechenaufwand für den Angriff weist je nach der Verschlüsselung gewaltige Unterschiede auf. Während 128 Bit für symmetrische Schlüssel ausreichen, werden angesichts der heutigen Verfahren zur Faktorisierung grosser Zahlen für die meisten Zwecke öffentliche Schlüssel mit 1024 Bits empfohlen.

3 Hybride Verschlüsselungsverfahren

Public-Key-Verschlüsselung ist kein Allheilmittel. Viele symmetrische Verfahren sind vom Sicherheitsstandpunkt aus betrachtet wirksamer, und die Ver- und Entschlüsselung ist bei Public-Key-Verfahren aufwendiger als bei entsprechenden symmetrischen Systemen, sie sind aber nichtsdestoweniger ein wirksames Werkzeug für den sicheren Austausch von symmetrischen Schlüsseln. Das ist die Idee bei hybriden Verschlüsselungssystemen.

Eine hybride Verschlüsselung benutzt sowohl eine symmetrische Verschlüsselung als auch ein asymmetrisches Public-Key-Verfahren. Die eigentliche Nachricht wird mit einem symmetrischen Sitzungsschlüssel verschlüsselt, welcher von einem Zufallsgenerator erzeugt wird. Dieser Sitzungsschlüssel wird dann mit dem öffentlichen Schlüssel des Empfängers verschlüsselt.

Sowohl PGP als auch GnuPG benutzen hybride Verschlüsselungsverfahren. Der mit dem öffentlichen Schlüssel des Empfängers verschlüsselte Sitzungsschlüssel und die symmetrisch verschlüsselte Nachricht werden automatisch zusammengefaßt. Der geheime Schlüssel des Empfängers wird zum Entschlüsseln des Sitzungsschlüssels verwendet, und dieser wird dann zum Entschlüsseln der eigentlichen Nachricht verwendet.

Ein hybrides Verschlüsselungsverfahren ist immer nur so gut wie der unsicherste Teil, egal ob das die Public-Key-Verschlüsselung oder die symmetrische Verschlüsselung ist. Da die symmetrischen Sitzungsschlüssel bei jedem Vorgang neu erzeugt werden, könnte ein Angreifer - selbst wenn er einen Sitzungsschlüssel entschlüsseln könnte - nur die mit dem betreffenden Sitzungsschlüssel verschlüsselte Nachricht entschlüsseln. Er müßte also für jede weitere Nachricht, die er lesen möchte, erneut einen Sitzungsschlüssel entschlüsseln.

4 Digitale Unterschriften

Eine Hash-Funktion [*]ist eine kryptographische Prüfsumme. Durch eine eindeutige Funktion wird aus einer Datei eine wesentlich kürzere Datensequenz erzeugt, die ein eindeutiges Abbild der Ursprungsdatei ist.

Die digitale Unterschrift eines Dokumentes ist das Ergebnis der Anwendung einer Hash-Funktion auf das Dokument. Um für digitale Unterschriften brauchbar zu sein, muß die Hash-Funktion jedoch zwei wichtige Eigenschaften haben:

Erstens sollte es unmöglich sein, zwei Dokumente zu finden, die dasselbe Hash-Ergebnis haben. Zweitens sollte es bei einem gegebenen Hash-Ergebnis schwer sein, das ursprünglich Dokument wiederherzustellen, aus dem dieser Hash erzeugt wurde.

Einige Public-Key-Verfahren könnten auch zum Unterschreiben von Dokumenten benutzt werden. [**] Der Unterzeichner verschlüsselt das Dokument mit seinem **privaten** Schlüssel. Jeder, der die Unterschrift prüfen und das Dokument sehen will, benutzt einfach den öffentlichen Schlüssel des Unterzeichners, um das Dokument zu entschlüsseln. Dieses Verfahren besitzt in der Tat die beiden Eigenschaften, die eine gute Hash-Funktion braucht, doch ist es in der Praxis zu langsam, um effektiv nutzbar zu sein.

Besser ist es, spezielle Hash-Algorithmen zu benutzen, welche diese beiden wichtigen Eigenschaften aufweisen; wie beispielsweise SHA1 und RIPE-MD160. Bei einem solchen Verfahren wird der Hash-Wert eines Dokumentes als Unterschrift verwendet. Man kann die Unterschrift dadurch prüfen, dass man auf die Kopie des Dokumentes ebenfalls die Hash-Funktion anwendet und den Hash-Wert, den man erhält, mit dem Hash-Wert des Originaldokumentes vergleicht. Wenn beide Werte übereinstimmen, dann sind beide Dokumente identisch.

Das Problem ist jetzt natürlich, Hash-Funktionen für digitale Unterschriften zu benutzen, ohne einem Angreifer das Manipulieren der Unterschrift zu ermöglichen. Wenn das Dokument und die Unterschrift unverschlüsselt geschickt werden, könnte ein Angreifer das Dokument verändern und eine entsprechende neue xxUnterschrift erzeugen, ohne dass der Empfänger es merkt. Wenn nur das Dokument verschlüsselt wird, könnte ein Angreifer die Unterschrift verfälschen und so das Scheitern einer Unterschriftsprüfung verursachen.

Eine dritte Möglichkeit besteht darin, ein hybrides Verfahren zu benutzen, um sowohl die Unterschrift als auch das Dokument zu verschlüsseln. Der Unterzeichner benutzt seinen privaten Schlüssel, und jedermann kann dessen öffentlichen Schlüssel benutzen, um die Unterschrift und das Dokument zu prüfen. Dies klingt zwar gut, ist aber in Wirklichkeit Unsinn. Wenn dieses Verfahren das Dokument wirklich sichern könnte, würde es dieses auch gegen Verfälschung sichern, und dann wäre die Unterschrift gar nicht nötig. Das ernstlichere Problem ist jedoch, dass dies keinen Schutz gegen Verfälschung bietet, weder für die Unterschrift noch für das Dokument. Bei diesem Verfahren wird nur der Sitzungsschlüssel für die symmetrische Verschlüsselung unter Benutzung des privaten Schlüssels des Unterzeichners verschlüsselt. Jeder kann den öffentlichen Schlüssel benutzen, um den Sitzungsschlüssel wiederherzustellen. Deshalb ist es für einen Angreifer einfach, den Sitzungsschlüssel wiederherzustellen und ihn zum Verschlüsseln von Ersatzdokumenten und Ersatzunterschriften zu benutzen, die er dann im Namen des Absenders an andere schickt.

Ein wirklich funktionierendes Verfahren ist es, nur die Unterschrift mit einem Public-Key-Verfahren zu verschlüsseln. Das heißt, es wird der geheime Schlüssel des Unterzeichners benutzt, um die digitale Unterschrift zu erzeugen, die dann jeder mit dem dazugehörigen öffentlichen Schlüssel checken kann. Das unterzeichnete Dokument kann man unverschlüsselt verschicken, wenn es öffentlich ist oder verschlüsselt, wenn es vertraulich ist. **Wenn das Dokument nach dem Unterzeichnen verändert wurde, wird die Unterschriftsprüfung negativ ausfallen.** Der von GnuPG standardmäßig benutzte **Digital Signature Algorithm** (**DSA**) arbeitet nach dieser Methode.

5 Fußnoten

Eine einfache Hash-Funktion ist **f(x) = 0** für alle ganzen Zahlen **x**. Eine interessantere Hash-Funktion ist **f(x) = x mod 37**, welche **x** auf den Rest von **x** dividiert durch 37 abbildet.

Die Verschlüsselung muß die Eigenschaft haben, dass der aktuelle öffentliche oder private Schlüssel vom Verschlüsselungsverfahren als der öffentliche Schlüssel benutzt werden könnte. RSA ist ein Beispiel eines solchen Verfahrens, ElGamal dagegen nicht.