Dirk Fox

TLS, das Vertrauen und die NSA

Wie die NSA die Sicherheitsinfrastruktur des Internet untergräbt

Die Datenzugriffe der NSA betrafen nicht nur Daten aus ‚Sozialen Netzwerken', von Suchmaschinen-Anbietern und Cloud-Dienstleistern, sondern auch ein Kernstück der Sicherheitsinfrastruktur des Internet: das SSL/TLS-Protokoll. Der Beitrag fasst die inzwischen bekannt gewordenen Methoden der NSA zusammen und schlägt Abhilfemaßnahmen vor.

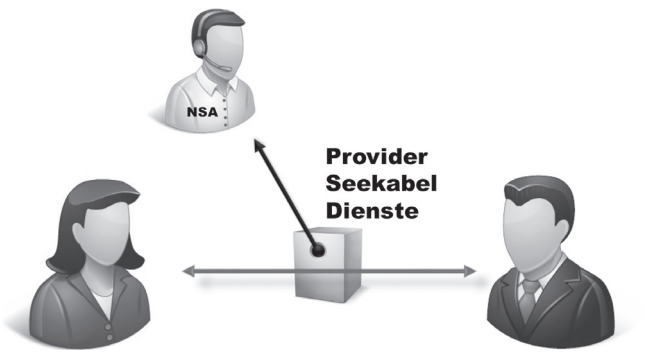
Einführung

Wie im Laufe des Jahres 2013 durch Dokumente bestätigt wurde, die der ehemalige Geheimdienstmitarbeiter Edward Snowden öffentlich machte, hat die US-amerikanische *National Security Agency* (NSA) seit Jahren nicht nur systematisch Kommunikationsverbindungen zwischen Europa und den USA abgehört, sondern auch die Metadaten und Inhalte von Internet-Verbindungen Anlass unabhängig gespeichert und ausgewertet.

Überraschend daran sind nicht so sehr die Tatsache an sich oder der gigantische Umfang der Abhörmaßnahmen, denn der *Foreign Intelligence Surveillance Act* (FISA) aus dem Jahr 1978 [1] ermächtigt die NSA seit über 35 Jahren zu derartigen Maßnahmen, und die Mittel dafür dürften ihr bei einem Jahres-Budget von über 10 Mrd. US-Dollar (2013) auch nicht fehlen. Dabei nutzt sie die Seekabel zwischen den USA und Europa als „Flaschenhals", den jede Kommunikation und Datenübertragung mit den USA oder über amerikanische Dienstanbieter passieren muss, und lässt sich dabei von (heimischen) Telekommunikations- und Telemedienanbietern und befreundeten Nachrichtendiensten unterstützen, insbesondere dem britischen GCHQ, aber auch dem deutschen BND (Abb. 1).

Überraschend sind eher die von der NSA inzwischen angewandten Methoden. Verstößt die Anlass unabhängige und unbegrenzte Vorratsdatenspeicherung der NSA ‚nur' gegen europäisches Recht, so schwächen einige der von der NSA eingesetzten Techniken die Sicherheitsmechanismen im Internet im Kern und schaffen damit Angriffspunkte für Dritte. Spionagemethoden, die die Sicherheit von (auch kritischen) Infrastrukturen insgesamt beeinträchtigen, sollten jedoch tabu sein bzw. international geächtet und in bi- oder multilateralen zwischenstaatlichen Vereinbarungen ausgeschlossen werden.

Abb. 1 | Abhörmaßnahmen der NSA



Um die Funktionsweise und Wirkungen dieser Methoden nachvollziehen zu können, ist eine genauere Betrachtung hilfreich. Am Beispiel der Angriffe der NSA auf das SSL- (bzw. TLS-) Protokoll wird dem im vorliegenden Beitrag nachgegangen.

1 Die Bedeutung des SSL-Protokolls

Das im Jahr 1994 als IETF-Standard publizierte *Secure Sockets Layer* (SSL) Protokoll [2] (Anfang des Jahrtausends in *Transport Layer Security* (TLS) umbenannt und inzwischen als TLS-Version 1.2 standardisiert [3]) bildet heute das Sicherheits-Rückgrat des Internet: Für praktisch jedes Anwendungsprotokoll gibt es eine „SSL-Variante", meist erkennbar am „s" im Protokoll-Akronym: HTTPS, FTPS, POP3S, SMTPS etc. Diese Protokollvarianten verwenden eigene TCP-Ports, die das Anwendungsprotokoll in eine SSL-Verbindung „einbetten".

Praktisch alle Internet-Dienste, die schützenswerte Daten übermitteln, wie Online-Shops, E-Banking-Lösungen, sicherer Filetransfer oder der verschlüsselte Zugriff auf E-Mail-Postfächer, vertrauen heute auf SSL. Das funktioniert nicht nur deshalb, weil SSL-basierte Protokollvarianten heute in vielen verbreiteten Standard-Programmen (Browser, E-Mail-Clients etc.) ent-

Dirk Fox

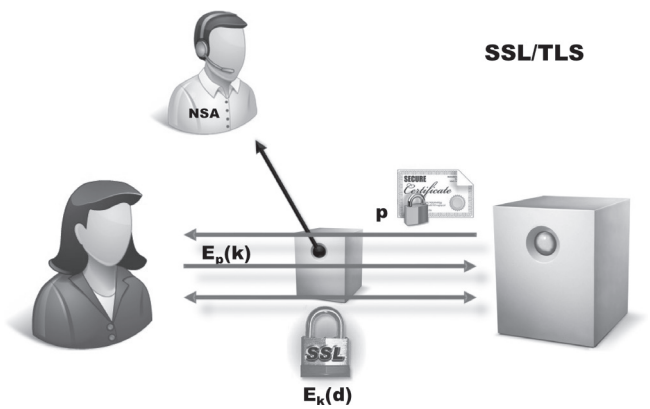
Geschäftsführer der Secorvo Security Consulting GmbH und Herausgeber der DuD.

E-Mail: dirk.fox@secorvo.de

halten sind und sich mit wenigen Mausklicks aktivieren lassen, sondern auch, weil es inzwischen anerkannt starke kryptographische Verfahren gibt, die unter Mitwirkung von internationalen Krypto-Experten in öffentlichen Verfahren entwickelt und als Standards verabschiedet wurden (wie AES, SHA, DSS oder ECDSA). Alle einschlägigen Verfahren werden vom SSL-Standard und den meisten SSL-Implementierungen unterstützt. Hilfreich und wichtig für die Verbreitung von SSL (TLS) war sicherlich auch das OpenSSL-Projekt: eine Open-Source-Implementierung von SSL, die 1995 von dem Australier Eric A. Young initiiert wurde und inzwischen in sehr viele, auch kommerzielle, Lösungen und Produkte integriert wurde.

Das SSL-Protokoll funktioniert im Wesentlichen wie folgt: Nach der Authentifizierung des Servers anhand eines (von einem vertrauenswürdigen Dritten ausgestellten und vom Browser überprüften) Schlüsselzertifikats wird zwischen Client (Browser) und Server ein nur für die Verbindung gültiger Verschlüsselungsschlüssel (Session Key) vereinbart. Anschließend kann die Datenübertragung der Internetverbindung mit anerkannt starken Verschlüsselungsverfahren verschlüsselt werden, so dass ein unberechtigter Zugriff auf die übertragenen Daten ohne Kenntnis der Schlüssel ausgeschlossen werden kann (Abb. 2).

Abb. 2 | Verschlüsselung der Daten mittels SSL



Dabei ist *k* der unter Verwendung des zertifizierten öffentlichen Schlüssels *p* verschlüsselt übermittelte Verschlüsselungsschlüssel (*Session Key*).

Beim Abhören einer solchen Kommunikation gewinnen Angreifer und Nachrichtendienste lediglich die Meta-Daten – wer hat wann an wen welche Datenmenge übermittelt – sowie verschlüsselte Inhaltsdaten $E_k(d)$, die sich auch mit der erheblichen Rechenkapazität eines Nachrichtendienstes ohne Kenntnis des Verschlüsselungsschlüssels *k* nach heutiger Kenntnis auch in Jahrhunderten nicht entschlüsseln lassen werden.

Rund um SSL und die standardisierten, starken Kryptoverfahren hat sich in den vergangenen Jahren eine „Vertrauensinfrastruktur" entwickelt: Anerkannte Zertifizierungsstellen bestätigen die Authentizität eines SSL/TLS-Server-Schlüssels durch ein digitales Zertifikat – und das Client-System kann dieses automatisch überprüfen.

Zwar funktioniert auch in dieser Vertrauensinfrastruktur nicht alles perfekt: Nicht jede Zertifizierungsinstanz prüft die Identität desjenigen, der einen Schlüssel zur Zertifizierung vorlegt, mit gleicher Sorgfalt, und nicht jedes Anwendungsprogramm durchläuft systematisch alle Prüfschritte bei der Vorlage eines Zertifikats. Jedoch sorgten bisher die „Selbstreinigungskräfte" des Marktes für eine Korrektur der einen oder anderen Nachlässigkeit.

Daher konnte man – zumindest bis Mitte 2013 – bei der Verwendung von SSL einigermaßen guten Gewissens auf zweierlei vertrauen: Darauf, dass man mit dem richtigen Server verbunden war, wenn dieser ein gültiges SSL-Zertifikat einer seriösen Zertifizierungsinstanz „vorlegte", und dass anschließend mittels SSL verschlüsselte Daten selbst einem nachrichtendienstlichen „Lauscher" auf Jahrhunderte verborgen bleiben würden, sofern in der SSL-Konfiguration hinreichend starke Verschlüsselungsverfahren ausgewählt waren.

2 Die Kompromittierung von SSL durch die NSA

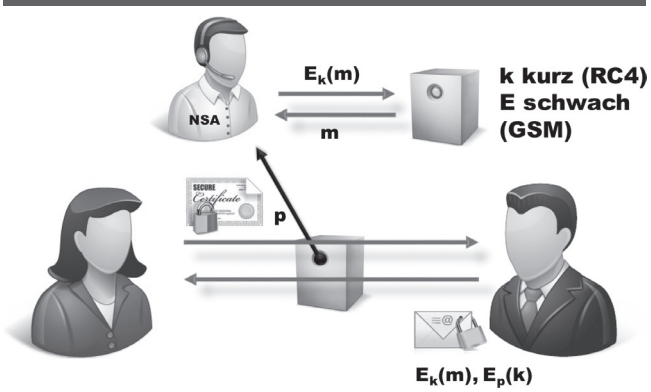
Dass eine solche Verschlüsselungsinfrastruktur nicht auf die Begeisterung von Nachrichtendiensten gestoßen sein dürfte, wird niemanden überraschen. Daher ist es auch nicht verwunderlich, dass Politik und Nachrichtendienste – allen voran die amerikanischen – über Jahrzehnte versuchten, die Entstehung einer solchen, sich ihrem Zugriff wirkungsvoll entziehenden sicheren Kommunikationsinfrastruktur zu verhindern oder wenigstens zu erschweren. Die Ursprünge solcher Maßnahmen finden sich bereits in den 70er Jahren des vergangenen Jahrhunderts.

2.1 Exportregulierung und schwache Verschlüsselung

Seit den 70er Jahren des 20. Jahrhunderts war die typische Antwort der NSA auf die Verbreitung von Verschlüsselungsverfahren zum Schutz digitaler Daten (auch vor nachrichtendienstlichem Zugriff) der Versuch, schwache Verschlüsselungsverfahren zu etablieren, oder genauer: Die Standardisierung (und damit indirekt auch die Verbreitung) von zu starken Verschlüsselungsverfahren zu verhindern. So geschehen bei der Standardisierung des *Data Encryption Standards* (DES) im Jahr 1977, der auf einer von IBM entwickelten Chiffre namens Lucifer beruhte. Diese von Horst Feistel entwickelte Chiffre besaß eine Schlüssellänge von 128 bit – zu viel für die NSA.

Im Verlauf des Standardisierungsverfahrens wurde die Schlüssellänge zunächst auf 64 bit verkürzt – damit schrumpfte der Schlüsselraum (und damit der Aufwand einer „Brute Force"-Ana-

Abb. 3 | Schwache Verschlüsselungsverfahren, die mit den Ressourcen der NSA entschlüsselt werden können



lyse, dem Ausprobieren aller möglichen Schlüssel) auf ein $2 \cdot 10^{19}$tel. Weitere acht Schlüsselbits wurden zu Paritätsbits gemacht, die sich aus den vorausgehenden 56 bit berechnen. Diese Maßnahme reduzierte die Komplexität eines Brute-Force-Angriffs noch einmal um den Faktor 2^{-8} – also auf ein 256stel.

Weil die NSA in den darauffolgenden Jahren feststellen musste, dass der verbleibende Rechenaufwand eines Brute-Force-Entschlüsselungsangriffs offenbar noch zu groß war, wurde flankierend der Export reguliert: Verschlüsselungsverfahren mit mehr als 40 bit Schlüssellänge fielen unter die Bestimmungen der Waffenexportgesetze und durften nicht mehr aus den USA (und Kanada) exportiert werden. Amerikanische Softwareanbieter, die ihre Produkte international verkaufen wollten und in ihren Produkten Verschlüsselungsmechanismen verwendeten, implementierten daher in den 90er Jahren 40-bit-Verschlüsselungsverfahren (wie z. B. den RC4, Abb. 3) – mit dem Resultat, dass aus Kompatibilitätsgründen auch die Käufer innerhalb der USA mit lediglich 40 bit langen Schlüsseln verschlüsselten. Der Aufwand der NSA für eine Brute-Force-Entschlüsselung mitgeschnittener verschlüsselter Kommunikation sank dabei auf ein 65.000stel gegenüber einer DES-Entschlüsselung mit 56 bit langen Schlüsseln.

Derweil gelang es den USA, auch bei einigen internationalen Standards, wie z. B. dem Mobilfunkstandard GSM, schwache Verschlüsselungsverfahren wie die Stromchiffre A5/2 zu etablieren, die von vielen Netzbetreibern – auch europäischen – noch bis 2007 zur Gesprächsverschlüsselung auf der „Luftschnittstelle" eingesetzt wurden. Selbst die 1987 entwickelte Stromchiffre A5/1 wurde bis vor kurzem auch von deutschen Mobilfunknetzbetreibern eingesetzt, obwohl sie bereits in den 90er Jahren gebrochen worden und seit 2000 eine effiziente Entschlüsselung möglich war [4]. Veröffentlichungen von Snowden belegen, dass die NSA daher GSM-Telefonate in Echtzeit entschlüsseln konnte.

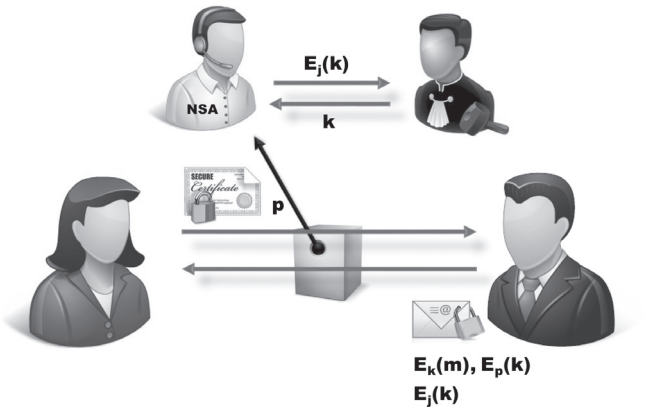
2.2 Key-Escrow-Regulierung

Anfang der 90er Jahre startete die US-Regierung unter Bill Clinton einen transparenteren Ansatz: Mit dem „Clipper"-Projekt sollten Telekommunikationsendgeräte mit einem Standard-Verschlüsselungschip mit starken Verfahren und ausreichend großen Schlüssellängen ausgestattet werden. Dieser Chip verschlüsselte zugleich den für die Verschlüsselung der Verbindung genutzten Verschlüsselungsschlüssel und schickte ihn in einem „LEAF" (*Law Enforcement Access Field*) mit, um bei Vorliegen einer richterlichen Abhörgenehmigung einen Zugriff auf die verschlüsselte Kommunikation zu ermöglichen (Abb. 4) [5].

Tatsächlich wäre mit Clipper die Verschlüsselung der Kommunikation zum Standard geworden – und der staatliche Zugriff auf die Kommunikationsdaten einer technischen Kontrolle unterworfen worden: ein erheblicher Fortschritt selbst gegenüber der heutigen Situation, denn telefonische und E-Mail-Kommunikation erfolgt nach wie vor überwiegend unverschlüsselt. Doch das Protokoll war unsicher und wurde 1994 von Matt Blaze gebrochen [6]; der Vorschlag endete daher mit einer Blamage.

Dennoch verfolgten die USA die Strategie der Schlüsselhinterlegung als Alternative für eine (ohnehin praktisch nicht durchsetzbare) Regulierung der Schlüssellängen weiter und ließen ab Ende 1996 den Export von Schlüssellängen bis 56 bit zu, sofern ein – euphemistisch *Key Recovery* genannter – Mechanismus im Bedarfsfall im Nachhinein einen Zugriff auf verschlüsselte Nachrichten ermöglichte. Sie initiierten die Gründung einer „*Key Recovery Alliance*", der zahlreiche Anbieter von Verschlüsselungslösungen beitraten, darunter auch die E-Mail-Verschlüsselungslösung PGP (*Pretty Good Privacy*). Die Hersteller integrierten daraufhin *Key Recovery*-Mechanismen in ihre Produkte. Bei PGP wurde in der Business-Version die Option eingebaut, den Verschlüsselungsschlüssel mit dem CMR-Key des Unternehmens (*Company Message Recovery*) zu verschlüsseln und im Header einer PGP-verschlüsselten Nachricht mitzuschicken – technisch vergleichbar dem Clipper-Konzept, wenn auch optional.

Abb. 4 | Key-Escrow-Mechanismen: Mit richterlicher Genehmigung kann der Verschlüsselungsschlüssel k zurückgewonnen werden



Key Escrow bzw. *Key Recovery* stießen jedoch auf erheblichen Widerstand von Experten [7] und konnten sich schließlich nicht durchsetzen. Schließlich gab das US-amerikanische *Department of Commerce* unter dem Druck der wachsenden IT-Industrie, die wegen der Sicherheitsschwächen ihrer Produkte in Europa Umsatzeinbußen hinnehmen musste, im Januar 2000 die bestehenden Export-Restriktionen für starke kryptographische Verfahren faktisch auf.

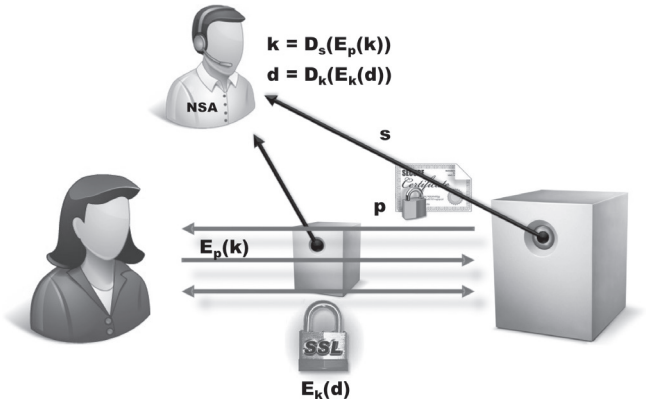
2.3 Direkter Zugriff auf den Secret Key

Was bleibt einem Nachrichtendienst, wenn starke kryptographische Verfahren verwendet werden und kein Schlüsselhinterlegungsmechanismus für den dabei verwendeten *Session Key* (den temporären Verschlüsselungsschlüssel für eine einzige Verbindung) existiert? Nur noch eines: Der Zugriff auf den geheimen Schlüssel eines der beiden Kommunikationspartner, mit dessen Hilfe der *Session Key* vereinbart oder verschlüsselt verschickt wurde. Bei Anwendungen, die das SSL-Protokoll verwenden, ist das aus zwei Gründen meist besonders einfach (Abb. 5):

- SSL-Verbindungen haben eine „*One-to-Many*"-Beziehung: Der (immer gleiche) Server kommuniziert mit vielen Nutzern. Der Zugriff auf den geheimen Server-Schlüssel erlaubt damit die (ggf. auch nachträgliche) Entschlüsselung aller Verbindungen.
- Der zum Zertifikat gehörige, geheime Schlüssel wird von Anbietern in der Regel über viele Jahre, oft auch bei einer Erneuerung des Schlüsselzertifikats nicht gewechselt. Daher können mit diesem Schlüssel meist auch lange zurückliegende Kommunikationsverbindungen nachträglich entschlüsselt werden.

Statt des im *Key Recovery*-Konzept vorgesehenen Zugriffs auf einen *Session Key* (und damit eine einzige Verbindung eines einzelnen Nutzers) erhält ein Nachrichtendienst mit dem *Secret Key* des Serverbetreibers den Zugriff auf alle Verbindungsschlüssel al-

Abb. 5 | Zugriff über den SSL-Secret-Key des Server-Betreibers

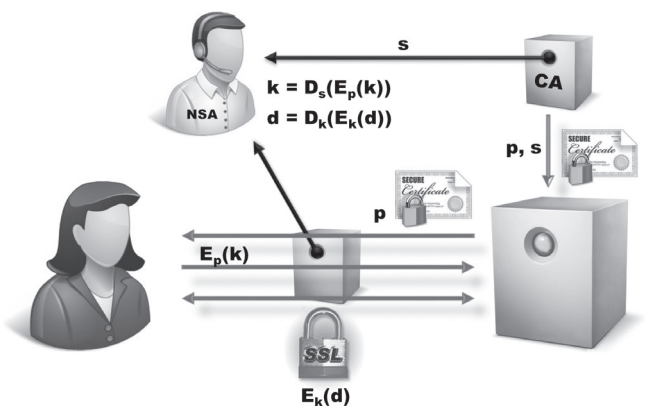


ler Nutzer über einen ggf. sehr weit zurückreichenden Zeitraum. In den USA gibt es seit dem verheerenden Terroranschlag auf die Twin Tower in New York für solche Zugriffe eine Rechtsgrundlage: den USA Patriot Act [8], verabschiedet am 25.10.2001, nur wenige Wochen nach dem Anschlag – ein Gesetz, dessen Entwurf zweifellos bereits in einer Schublade bereitgelegen hatte.

Zahlreiche Zertifizierungsdienstanbieter, die Zertifikate für die öffentlichen SSL-Schlüssel eines Anbieters ausstellen, erzeugen die Schlüsselpaare – d. h. den geheimen und den öffentlichen Schlüssel – gleich mit und bieten die Aufbewahrung des Schlüsselpaars als *Recovery*-Service für den Fall eines Schlüsselverlusts an. Zwar ist das beim SSL-Protokoll überflüssig, da man jederzeit ein neues Schlüsselpaar (mit Zertifikat) einsetzen kann und nie ältere Schlüssel zur Entschlüsselung benötigt, denn die Entschlüsselung erfolgt immer in Echtzeit mit einem zuvor ausgehandelten *Session Key*. Einem Nachrichtendienst bietet dieser (vom Kunden bezahlte) Service eines kooperierenden Zertifizierungsdienstanbieters den Zugriff auf viele Server – und deren zukünftige und ggf. zurückliegende Kommunikation), ohne Wissen des Serverbetreibers und der Service-Nutzer (Abb. 6). Zugleich erlangt die NSA auf diese Weise Zugriff auf geheime SSL-Schlüssel von ausländischen Servern, wenn diese einen amerikanischen Anbieter für die Schlüsselgenerierung und Zertifikatserstellung genutzt haben.

Welchen Anbieter die NSA zur Herausgabe der Schlüssel auffordern muss, ist dabei einfach herauszufinden: Der Aussteller des Zertifikats ist – praktischerweise – im Zertifikat eingetragen.

Abb. 6 | Zugriff über den bei einem Zertifizierungsdienstanbieter (Certification Authority, CA) gespeicherten Secret-Key

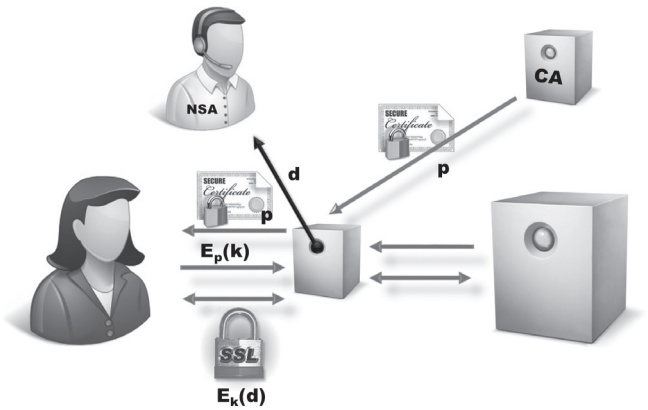


2.4 Man-in-the-Middle-Angriff mit falschem Zertifikat

Mit dem Zugriff auf einen (oder mehrere) Zertifizierungsdiensteanbieter – nicht nur aufgrund rechtlicher Sonderbefugnisse, sondern zum Teil offenbar auch durch verdeckte „Gründung" eines solchen Anbieters – ergab sich für die NSA noch eine weitere Möglichkeit, SSL-Verbindungen zu kompromittieren: durch die Verwendung falscher Zertifikate.

Wenn sich die NSA über einen Zertifizierungsdiensteanbieter ein Zertifikat für eine Domäne ihrer Wahl ausstellen lassen kann, kann sie sich als ein beliebiges Shop-System oder als Informationsanbieter „maskieren" und so einen eleganten *Man-in-the-Middle*-Angriff (MitM) realisieren: Bei einem SSL-Verbindungsaufbau eines Client-Systems mit einer überwachten Verbindung kann sich ein NSA-Server mit dem „falschen" Server-Zertifikat als eben dieser Anbieter ausgeben. Damit die Fälschung nicht auffällt, verbindet sich der NSA-Server dabei über eine reguläre SSL-Verbindung mit dem Server des echten Anbieters und leitet die Datenpakete weiter (Abb. 7).

Abb. 7 | MitM-Angriff mit falschem Zertifikat



Jedes Paket, das nun zwischen Client und Server ausgetauscht wird, wird auf dem NSA-Server entschlüsselt und erst dann, erneut verschlüsselt, an den eigentlichen Empfänger (Nutzer, echter Server) weitergeleitet. Alle Inhaltsdaten liegen damit in Echtzeit unverschlüsselt auf dem NSA-Server vor.

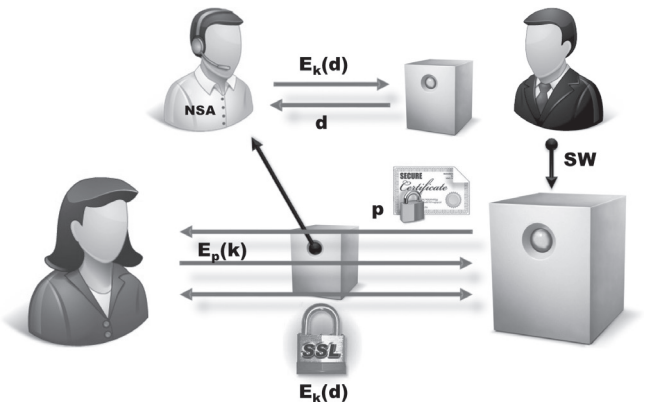
Entdecken lässt sich dieser Angriff vom von der Überwachungsmaßnahme betroffenen Nutzer nur durch (manuelles) Vorhalten des ursprünglichen Server-Zertifikats und einen Vergleich mit dem Zertifikat (und dem öffentlichen Schlüssel), das der Server beim Verbindungsaufbau übermittelt. Da diese Prüfung von keinem Webserver unterstützt wird, ist das praktisch unmöglich. Außerdem kann der Client nicht unterscheiden, ob es sich bei einem neuen Zertifikat um eine solche „Fälschung" handelt, oder ob der Anbieter lediglich ein neues Zertifikat verwendet. Und wenn er sich erstmalig mit dem Anbieter verbindet, ist ihm auch kein „Referenz-Zertifikat" bekannt, mit dem er das übermittelte vergleichen könnte.

Nachteilhaft aus der Perspektive der NSA ist allerdings, dass ein solcher Überwachungsangriff gezielt und auf wenige Domänen bezogen erfolgen muss, also nicht gut „skaliert" und daher für eine Massenüberwachung ungeeignet ist.

2.5 Hintertüren

Flankierend sind daher Hintertüren eine gute Methode. Eine Hintertür in einem Sicherheitsprodukt wie SSL kann unterschiedlich realisiert sein: als verdeckter Kanal, der geheime Schlüssel preisgibt (entweder versteckt innerhalb des Protokolls, z. B. als „Padding" eines zu kurzen Datenpakets, oder – auffälliger – durch die Übermittlung des *Session Keys* an einen voreingestellten Zielserver unter der Kontrolle der NSA). Besser, da schwieriger zu entdecken, sind Mechanismen, die den Schlüssel „erratbar" machen, indem sie z. B. den Schlüsselraum künstlich verkleinern (Abb. 8).

Abb. 8 | Hintertür in der Software



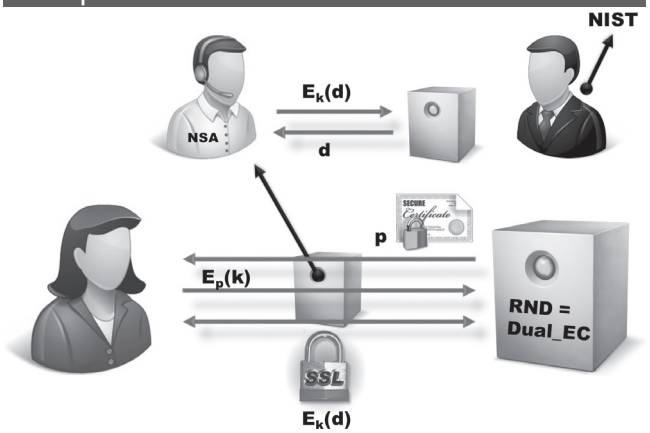
Denn ein verdeckter Kanal kann bei einer sorgfältigen Analyse des Protokolls oder der Datenverbindungen eines Servers auffallen. Die Verkleinerung des Schlüsselraums hingegen hat den Charme, dass sie nur durch eine aufwändige Analyse des Programmcodes entdeckt werden kann. Denn die mangelhafte Zufälligkeit einer Schlüsselwahl lässt sich von außen praktisch nicht feststellen, wenn zumindest Teile des Schlüssels zufällig gewählt werden (z. B. 100 Bits fest, 28 Bits zufällig wählen, dann mit SHA-1 hashen – und der verbleibende Suchraum für einen Brute-Force-Angriff reduziert sich auf 2^{28} mögliche Schlüssel).

Aber das Einspielen von Hintertüren setzt Kooperation voraus – entweder muss die Hintertür-Software von einem kooperierenden Unternehmen (wie der Firma RSA, Tochter von EMC² Corp. [9]) verbreitet oder sie muss durch „Ersetzung" der eigentlichen Software installiert werden. Letzteres kann aufwändig sein, wie die von der NSA nachweislich vorgenommene manuelle Installation von Hintertür-Software in Cisco-Geräten im Verlauf des Postversands [10].

Möglicherweise hat die NSA auch versucht, Hintertüren in Open-Source-Software wie OpenSSL einzubringen. Tatsächlich sind mehrere Fälle bekannt, in denen Fehler in OpenSSL-Implementierungen als Hintertür genutzt werden konnten, wie die Abschaltung des Zufallsgenerators in einer Debian-Distribution von 2006, die man erst im Jahr 2008 entdeckte [11], oder die *Heartbleed*-Schwachstelle, die Anfang April 2014 bekannt wurde [12]. Ob tatsächlich die NSA für diese Schwachstellen verantwortlich war, ist allerdings bislang nicht belegt.

Noch eleganter ist es, die Hintertür quasi vorzugeben – indem sie in einem Standard verdeckt festgelegt wird. Das funktioniert natürlich nicht auf so offensichtlichem Wege, wie durch die Spezifikation eines schwachen Verschlüsselungsalgorithmus – wohl aber, wenn es gelingt, die Parameter eines Zufallszahlengenerators zu manipulieren. So schaffte es die NSA, wie von Shumow und Ferguson bereits 2007 vermutet [13] und inzwischen bestätigt, in den vom NIST erstmalig 2007 publizierten Standard *Recommendation for Random Number Generation Using Deterministic Random Bit Generators* (SP 800-90A [14]) den auf Elliptischen Kurven basierenden Zufallszahlengenerator Dual_EC_DRBG einzuschleusen. Im Anhang des Standards werden feste Kurvenpunkte P und Q spezifiziert – kennt man deren mathematische Beziehung zueinander, kann man die damit erzeugten Zufallszahlen voraussagen. Die mathematische Beziehung von P und Q kann man aber nur kennen, wenn man die Punkte geeignet gewählt hat – nachträglich lässt sie sich nicht berechnen. Auffällig ist, dass im Standard empfohlen wird, fast alle erzeugten Bits als Zufallswert zu verwenden (und nicht, wie sonst üblich, nur maximal die Hälfte). Zwar kann man den Dual_EC_DRBG auch mit selbst erzeugten Punkten P und Q verwenden – allerdings tauchen eben jene im Anhang des Standards spezifizierten Punkte P und Q in den Testvektoren für die Zertifizierung nach FIPS-140-2 [15] auf – ein Schelm, wer Arges dabei denkt.

Abb. 9 | Hintertür in Standards



3 Gegenmaßnahmen

Vor einigen dieser von der NSA systematisch betriebenen Bedrohungen der Sicherheitsinfrastruktur kann man sich (bzw. die Nutzer eines SSL/TLS-geschützten Web-Dienstes) durch die Beachtung einiger Grundregeln schützen.

3.1 Selbst erzeugte Schlüssel

Zahlreiche Zertifizierungsstellen bieten nicht nur Zertifikate, sondern auch gleich die Erzeugung des zugehörigen (asymmetrischen) Schlüsselpaars an. Wer diesen Dienst in Anspruch nimmt, erspart sich zwar den Vorgang der Schlüsselgenerierung – erkauft sich diese Erleichterung aber mit der Ungewissheit, ob nicht doch eine Kopie des geheimen Schlüssels beim Anbieter verbleibt (oder an Dritte weitergegeben wird).

Die Werbung mit einem „Schlüssel-Backup" ist im Zusammenhang mit SSL/TLS-Schlüsseln zudem irreführend – denn ein Backup der Schlüssel ist schlechthin überflüssig, da die Schlüssel nur dem Schutz einer Kommunikationsverbindung dienen. Sie können jederzeit durch ein neues Schlüsselpaar ersetzt werden; dafür ist prinzipiell – zumindest, wenn keine Kompromittierung vorliegt – nicht einmal eine Schlüsselsperrung erforderlich.

3.2 Sichere Zufallszahlen und Schlüssel

Die Sicherheit einer verschlüsselten und authentischen Verbindung steht und fällt mit der Sicherheit der verwendeten Schlüssel: Die Schlüssel müssen hinreichend lang sein, dürfen keinem Dritten (außer dem Server) bekannt oder zugänglich und dürfen nicht vorhersagbar sein.

Daher muss der Schlüsselerzeugung besondere Aufmerksamkeit gewidmet werden: Der Zufallszahlengenerator muss mit einem guten *Seed* (einer Quelle möglichst echt zufälliger Daten – oft erzeugt aus Mausbewegungen, verknüpft mit anderen, praktisch nicht vorhersagbaren Prozessordaten) arbeiten und daraus mit einem kryptografischen Pseudozufallszahlengenerator eine nicht vorhersagbare (Pseudo-) Zufallsfolge erzeugen. Der auf diese Weise erzeugte geheime Schlüssel darf wiederum nur an besonders geschützter Stelle gespeichert werden.

Idealerweise verwendet man für Erzeugung und Speicherung der Schlüssel ein *Hardware Security Module* (HSM), das nachweislich gute Algorithmen zur Schlüsselerzeugung verwendet, zertifiziert ist (FIPS-140-2 [15]) und die erzeugten Schlüssel nie preisgibt, sondern die erforderlichen kryptografischen Operationen im Modul selbst durchführt.

3.3 Zertifizierte Software

Software kann Hintertüren enthalten. Wer den Einsatz solcher Software verhindern möchte, sollte auf Software ausweichen, deren Sicherheit nach den *Common Criteria* (CC) zertifiziert ist.[1] Leider gibt es nur sehr wenige Anbieter von CC-zertifizierten Sicherheitsprodukten. Das hängt auch mit dem Aufwand zusammen, der mit einer solchen Zertifizierung verbunden ist. Dennoch wäre es wünschenswert, für sicherheitstechnische Kernkomponenten (wie z. B. einen SSL/TLS-Stack) auf zertifizierte Software zurückgreifen zu können.

Der Einsatz von *Open Source*-Lösungen wird oft als Alternative empfohlen. Tatsächlich bieten Produkte mit offengelegtem Quellcode die Chance, Fehler oder Hintertüren in der Implementierung zu finden. Allerdings hat gerade die Geschichte von OpenSSL gezeigt, dass immer wieder kritische Fehler über Jahre nicht entdeckt wurden – darunter auch der bereits erwähnte elementare Bug eines festen *Seed* für die Zufallszahlenerzeugung [11]. Auch werden häufig nicht die Original-Quellen neu übersetzt, sondern bereits fertig übersetzte Bibliotheken verwendet – damit gibt es keine Garantie, dass die verwendete Bibliothek mit dem ursprünglichen, veröffentlichten Quellprogramm übereinstimmt.

3.4 Sichere Kryptoverfahren

Bei der Auswahl der SSL/TLS-Protokollvarianten sollten ausschließlich sichere Kryptoverfahren verwendet werden. Immer wieder werden aus vermeintlich erforderlicher Interoperabilität mit veralteten Browser-Versionen noch Protokollvarianten mit schwachen oder bereits gebrochenen Verfahren unterstützt – und damit die Sicherheit aller Verbindungen mit dem Server der Gefahr einer Kompromittierung ausgesetzt.[2]

3.5 Neue Schlüssel nach einer Kompromittierung

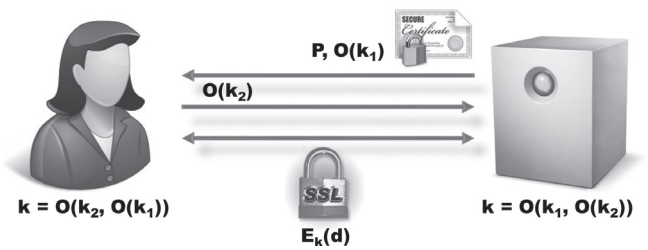
Immer wieder kommt es zu Kompromittierungen von Servern – nicht allein durch Fehler in SSL/TLS-Implementierungen, sondern auch nach Exploit-Angriffen, die Fehler in einer auf dem Server betriebenen Anwendungen ausnutzen. Meist lässt sich nach einem solchen Angriff nicht ausschließen, dass auch der geheime SSL/TLS-Schlüssel kompromittiert wurde, denn nur selten werden HSMs zur sicheren Speicherung der geheimen Schlüssel verwendet.

Die Betreiber lassen sich anschließend zwar oft neue SSL/TLS-Zertifikate ausstellen – vergessen dabei aber, meist aus unzureichendem Verständnis der kryptografischen Mechanismen, auch die zugehörigen Schlüssel neu zu erzeugen.

3.6 Perfect Forward Secrecy

Ein besonders wirksamer Mechanismus, um eine nachträgliche Entschlüsselung von mitgeschnittenen SSL/TLS-Verbindungen zu verhindern, ist die Verwendung einer Protokollvariante, die *Perfect Forward Secrecy* ermöglicht [16]. Ein solches Protokoll ist der Diffie-Hellman-Schlüsselaustausch, bei dem bei der Erzeugung des *Session Key* jeweils ein Schlüsselteil verwendet wird, das nur dem Client bzw. dem Server bekannt ist und nach der Schlüsselerzeugung gelöscht wird [17]. Die Schlüsselteile k_1 und k_2 werden auch nicht verschlüsselt ausgetauscht, sondern als Ergebnis einer Einweg-Funktion *O*, die von einem Lauscher nicht „zurückgerechnet" werden kann (Abb. 10). Der *Session Key* wird nach Ende der Verbindung von Client und Server gelöscht – und auch bei nachträglichem Zugriff der NSA auf den geheimen Schlüssel des Servers kann der bei einer belauschten Verbindung verwendete *Session Key* nicht zurückgewonnen und der Mitschnitt nicht entschlüsselt werden.

Abb. 10 | Perfect Forward Secrecy (PFS)



4 Fazit

Die Maßnahmen, mit denen die NSA versucht, Zugriff auf Kommunikationsdaten zu erhalten, überschreiten – auch wenn man die Legitimität einer solchen Überwachung befürworten sollte – eine inakzeptable Grenze, wenn sie dazu führen, dass die Sicherheitsinfrastruktur des Internet insgesamt geschwächt wird.

Gegen die Zugriffe auf Server-Schlüssel helfen Protokolle wie PFS, die ohne „*Master Key*" auskommen – ein Aspekt, der bei der Erforschung und Entwicklung von Protokollen in der Vergangenheit nicht immer adäquat berücksichtigt wurde. Die Fälschung

1 Siehe die Materialsammlung des Bundesamtes für Sicherheit in der Informationstechnik (BSI): http://www.bsi.bund.de/DE/Themen/ZertifizierungundAnerkennung/ZertifizierungnachCCundITSEC/ITSicherheitskriterien/CommonCriteria/commoncriteria_node.html

2 Zur sicheren Konfiguration von TLS siehe Kai Jendrian, TLS Dos and Don'ts, in diesem Heft.

von Zertifikaten, die Installation von Hintertüren in Produkten und vor allem die Manipulation von Standards pervertieren jedoch die Bestrebungen freier Gesellschaften, das Persönlichkeitsrecht der vertraulichen Kommunikation auch im „digitalen Zeitalter" sicherzustellen.

Die Analyse und Auswertung von Metadaten durch Nachrichtendienste lässt sich (zumindest derzeit) technisch nicht verhindern, sondern nur (ggf. missbilligend) ertragen. Aber Maßnahmen von Nachrichtendiensten freiheitlicher Staaten, die undifferenziert und systematisch das technisch erreichte Schutzniveau der IT-Infrastrukturen unterspülen, gehören in zwischenstaatlichen oder völkerrechtlichen Vereinbarungen untersagt.

Literatur

[1] 95th United States Congress: Foreign Surveillance Act. Public Law 95-511, 1978. (http://en.wikisource.org/wiki/Foreign_Intelligence_Surveillance_Act_of_1978)

[2] Esslinger, Bernhard; Müller, Maik: *Secure Sockets Layer (SSL) Protocol*. Datenschutz und Datensicherheit (DuD) 11/1997, S. 691-697.

[3] IETF: *The Transport Layer Security (TLS) Protocol Version 1.2*. RFC 5246, August 2008. (http://tools.ietf.org/html/rfc5246)

[4] Alex Biryukov, Adi Shamir, David Wagner: *Real Time Cryptanalysis of A5/1 on a PC*. 27.04.2000. (http://cryptome.info/0001/a51-bsw/a51-bsw.htm)

[5] Rueppel, Rainer: *Clipper – Der Krypto-Konflikt am Beispiel der Amerikanischen ESCROW Technologie*. Datenschutz und Datensicherung (DuD), 8/1994, S. 443-451.

[6] Matt Blaze: *Protocol Failure in the Escrowed Encryption Standard*. Proceedings of Second ACM Conference on Computer and Communications Security, Fairfax, VA, November 1994. (http://www.crypto.com/papers/eesproto.pdf)

[7] Abelson, Hal; Anderson, Ross; Bellovin, Steven M.; Benaloh, Josh; Blaze, Matt; Gilmore, John; Neumann, Peter G.; Rivest, Ronald L.; Schiller, Jeffrey I.; Schneier, Bruce: *Risiken von Key Recovery, Key Escrow und Trusted Third Party-Verschlüsselung*. Bericht vom 27.05.1997. Übertragung ins Deutsche: Holger Heimann, Dirk Fox. Datenschutz und Datensicherheit (DuD), 1/1998, S. 14-23.

[8] 107th United States Congress: Uniting and Strengthening America by Providing Appropriate Tools Required to Intercept and Obstruct Terrorism (USA PATRIOT ACT) Act of 2001. H.R.3162.ENR. (http://thomas.loc.gov/cgi-bin/query/z?c107:H.R.3162.ENR:)

[9] *Coviello verteidigt Zusammenarbeit mit der NSA*. golem.de, 26.02.2014. (http://www.golem.de/news/rsa-security-chef-coviello-verteidigt-zusammenarbeit-mit-der-nsa-1402-104817.html)

[10] arstechnica: *Photos of an NSA "upgrade" factory show Cisco router getting implant*. 14.05.2014. (http://arstechnica.com/tech-policy/2014/05/photos-of-an-nsa-upgrade-factory-show-cisco-router-getting-implant/)

[11] Debian.org: *DSA-1571-1 openssl – Voraussagbarer Zufallszahlengenerator*. Debian-Sicherheitsankündigung, 13.05.2008. (https://www.debian.org/security/2008/dsa-1571)

[12] Codenomicon Ltd.: *The Heartbleed Bug*. (http://heartbleed.com/)

[13] Shumow, Dan; Ferguson, Niels: On the Possibility of a Back Door in the NIST SP800-90 Dual EC Prng. Vortrag auf der Rump Session der Crypto 2007, Santa Barbara. (http://rump2007.cr.yp.to/15-shumow.pdf)

[14] Barker, Elaine; Kelsey, John: *Recommendation for Random Number Generation Using Deterministic Random Bit Generators*. NIST Special Publication 800-90A, January 2012. (http://csrc.nist.gov/publications/nistpubs/800-90A/SP800-90A.pdf)

[15] NIST: *Security Requirements for Cryptographic Modules. FIPS 140-2*, 25.05.2001. (http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf)

[16] Fox, Dirk: *Perfect Forward Secrecy*. Gateway, DuD 11/2013, S. 792.

[17] Diffie, Whitfield; Hellman, Martin E.: *New Directions in Cryptography*. IEEE Transactions on Information Theory, Vol. IT-22, No. 6, 1976, S. 644-654.