DuD
Datenschutz und Datensicherheit

Schwäche im OpenPGP-Standard

Attacke auf die geheimen Schlüssel von PGP-Benutzern

Hans-Joachim Knobloch

Zwei tschechische Kryptologen entdeckten Ende März 2001 „offensichtliche" Sicherheitsschwächen im offenen Sicherheitsstandard OpenPGP. Ein Schulbeispiel für die Schwierigkeit der Spezifikation starker Sicherheitsstandards und die Unmöglichkeit von Fehlerfreiheit – auch in der Welt des OpenSource.

Hans-Joachim Knobloch

Secorvo Security Consulting GmbH. Arbeitsschwerpunkt: Internet-Sicherheit, IP-VPN, Kryptologie.
E-Mail: knobloch@secorvo.de

Einleitung

Ende März 2001, just während der Tage, in denen sich die IT-Welt zur CeBIT in Hannover traf, machte die Nachricht über eine Attacke auf das verbreitete Verschlüsselungsprogramm PGP Schlagzeilen. Die Computerzeitschrift *c't* stellte sogar vorübergehend die alljährlich angebotene Zertifizierung von PGP-Schlüsseln auf ihrem Messestand ein.[1]

Glücklicherweise erwiesen sich nach ersten Analysen der kurz zuvor publizierten Attacke die schlimmsten Befürchtungen als unbegründet. Was bleibt, mahnt dennoch zur Vorsicht beim Umgang mit geheimen Schlüsseln und ergibt ein interessantes Lehrstück über die Problematik der Entwicklung starker Sicherheitsstandards.

1 Die Attacke

Die von den beiden tschechischen Kryptologen Vlastimil Klíma und Tomáš Rosa entwickelte Attacke [KR01] richtet sich gegen das in RFC 2440 [CDFT98] genormte Datenformat in OpenPGP zur Speicherung des geheimen (privaten) PGP-Nutzerschlüssels. Dieses Format wird auch von anderen zum OpenPGP-Standard konformen Verschlüsselungsprogrammen (z. B. GnuPG[2]) verwendet.

Diese Datei, in PGP in der Regel „secring.skr" genannt, ist mit einem starken symmetrischen Verschlüsselungsalgorithmus und einem Schlüssel geschützt, der aus der vom Benutzer gewählten Passphrase abgeleitet wird.

Die Attacke von Klíma und Rosa ermöglicht es einem Angreifer, durch eine Manipulation dieser Datei den geheimen Schlüssel eines Benutzers zu rekonstruieren, ohne dessen Passphrase zu kennen. Inzwischen wurde von Orlin Grabbe eine Java-Implementierung des Angriffs veröffentlicht [G01].

1.1 Angriffsszenario

Voraussetzung für die Attacke ist, dass der Angreifer die Datei mit dem geheimen Schlüssel des Benutzers unbemerkt beschreiben, also verändern kann.

Der Angriff besteht aus den drei folgenden Schritten:

- Der Angreifer modifiziert die Passphrase-geschützte Datei mit dem geheimen Schlüssel des Benutzers.
- Der Benutzer aktiviert irgendwann danach seinen geheimen Schlüssel mit der Passphrase und signiert eine Datei oder Nachricht.
- Der Angreifer fängt diese mit dem geheimen – aus der von ihm modifizierten Schlüsseldatei gewonnenen – Schlüssel signierte Nachricht ab und kann daraus den geheimen Schlüssel ermitteln.

1.2 Schwachpunkt

Der Schwachpunkt, den der Angriff der beiden tschechischen Kryptologen ausnutzt, ist die Tatsache, dass die Verschlüsselung mit der Passphrase nicht in ausreichendem Maße die Integrität des gesamten privaten Schlüssels schützt, sondern hauptsächlich die Vertraulichkeit der geheimen Parameter. Bestimmte Veränderungen dieser nur teilweise verschlüsselten Datei durch einen Angreifer fallen daher nicht auf.

[1] Siehe Heise-News-Ticker unter http://www.heise.de/newsticker/data/hag-22.03.01-001/default.shtml

[2] Gnu Privacy Guard; vom Bundesinnenministerium geförderte Open-Soure-Initiative zur Entwicklung einer PGP-kompatiblen, freien Softwarelösung. Siehe http://www.gnupg.de.

Dies ermöglicht es einem Angreifer, einen DSA-Schlüssel[3] so zu modifizieren, dass er anstelle des Problems des diskreten Logarithmus (DLP), das eigentlich der Sicherheit von DSA zugrunde liegt[4], nur eine wesentlich einfachere Variante des DLP lösen muss.

Im Falle von RSA-Schlüsseln kann der Angreifer Modifikationen vornehmen, die es ihm erlauben, grundlegende Methoden der Fehler-Kryptanalyse [BDL96, BS96] anzuwenden, um den RSA-Modulus zu faktorisieren und damit den Schlüssel zu brechen.

1.3 DSA-Schlüssel

Die Basisparameter des DSA-Verfahrens sind zwei Primzahlen p und q sowie ein Generator g der multiplikativen Untergruppe der Ordnung q modulo p. Aus dem geheimen Exponenten x leitet sich der öffentliche Schlüssel $y = g^x \bmod p$ ab.

Gemeinhin wird x als der geheime Schlüssel bezeichnet, und tatsächlich wird auch nur dieser Parameter vom OpenPGP-Dateiformat mit der Passphrase geschützt. Strenggenommen sind aber auch p, q und g Teile des privaten Schlüssels, deren Integrität sichergestellt werden muss.

Genau dies aber leistet das standardisierte Format nicht (vgl. Bild 1). Die Attacke auf DSA-Schlüssel besteht nun darin, die Primzahl p, die in der Regel 1024 Bit groß gewählt wird, durch eine wesentlich kleinere, 159 Bit große Primzahl p' zu ersetzen.

Octets	Data	C/E
1	Version	-/-
4	Creation Time	-/-
1	Public Key AlgID	-/-
2+128	Prime p	-/-
2+20	Prime q	-/-
2+128	Generator g	-/-
2+128	Public Value y	-/-
1	String-to-Key Usage	-/-
(1)	Key Encryption AlgID	-/-
(11)	String-to-Key Specifier	-/-
(8)	Initialization Vector	-/-
2+20	Secret Exponent x	x/x
2	Checksum	-/x

Bild 1: Datenfelder eines OpenPGP v4 DSA Secret Key Packets (C/E steht für Schutz durch Checksum/Encryption).

[3] DSA: Digital Signature Algorithm; in den USA standardisiertes digitales Signaturverfahren [NIST00].

[4] Zur Sicherheit des DSA siehe [F97].

Generell ist es mit der heute verfügbaren Rechenleistung recht einfach, diskrete Logarithmen modulo eines p' dieser Größe zu berechnen. Der Angreifer kann sich seine Arbeit zusätzlich erleichtern, indem er p' so wählt, dass p'-1 in eine Zweierpotenz und einen kleinen Primfaktor zerfällt. Dadurch wird es möglich, einen speziell für diese Form von p' optimierten Algorithmus einzusetzen, der um Größenordnungen effizienter ist als die Standardalgorithmen, die üblicherweise verwendet werden, um diskrete Logarithmen zu berechnen.

1.4 RSA-Schlüssel

Der Basisparameter des RSA-Verfahrens ist der Modulus n, der sich als Produkt zweier (geheimzuhaltender) Primzahlen p und q ergibt. Ein geheimer Exponent d korrespondiert mit dem öffentlichen Exponenten e so, dass $a = a^{ed} \bmod n$.

Zur Effizienzsteigerung werden häufig, so auch von PGP, die Berechnungen mit dem geheimen Exponenten nach dem Chinesischen Restesatz durchgeführt. Das heisst, die Software rechnet zunächst getrennt modulo p und modulo q und erst abschließend werden die beiden Teilergebnisse kombiniert. In diesem Fall sind neben dem geheimen Exponenten auch p und q sowie der für die Verknüpfung der beiden Teilergebnisse benötigte Faktor $p^{-1} \bmod q$ Teil des geheimen Schlüssels.

Zwar sind nach dem aktuellen OpenPGP-Dateiformat alle diese Parameter mit der Passphrase geschützt. Eine weitere Schwäche des Datenformats macht dennoch eine unbemerkte Änderung möglich:

Zur Verschlüsselung der Parameter wird eine Blockchiffre im Cipher-Feedback Modus (CFB) eingesetzt. Die Charakteristik der Fehlerfortpflanzung im CFB macht diesen Modus höchst geeignet für Anwendungen, bei denen es auf die Resynchronisation bei der Entschlüsselung eines Datenstroms ankommt. Zum Integritätsschutz eignet sich der CFB jedoch gerade aufgrund dieser Charakteristik nicht. Speziell im letzten Block des Chiffretexts ist es möglich, gezielt einzelne Bits zu „kippen“, ohne dass sich dies auf andere Teile des anschließend entschlüsselten Klartexts auswirkt.

Diese Eigenschaft machen sich Klíma und Rosa zunutze, um den Parameter $p^{-1} \bmod q$ abzuändern. Dabei ist es nicht wichtig, den Wert gezielt zu ändern. Alleine die Tatsache, dass der Parameter anschließend nicht mehr „stimmt“, erlaubt es, eine grundlegende Methode der Fehler-Kryptanalyse zum Brechen von Berechnungen nach dem Chinesischen Restesatz [BDL96] anzuwenden.

Durch die Störung von $p^{-1} \bmod q$ wird ein Fehler bei der Zusammensetzung der beiden zuvor modulo p und modulo q errechneten Teilergebnisse provoziert. Die so erhaltene Signatur s' ist zwar korrekt – d. h. kongruent zu dem aufbereiteten Hash-Wert m der Nachricht – modulo p, nicht aber modulo q. Aus

- $s' = m \bmod p$ und
- $s' \neq m \bmod q$

folgt, dass s'-m zwar von p, nicht aber von q geteilt wird. Daher kann der öffentliche RSA-Modulus n durch eine simple GGT-Berechnung mit s'-m faktorisiert werden. Aus p und q wiederum ergeben sich direkt auch alle anderen geheimen Parameter.

2 Relevanz

Es sind keine Fälle bekannt, in denen die Attacke außerhalb von Laborumgebungen durchgeführt worden wäre.

Das Angriffsszenario setzt voraus, dass der Angreifer über weitgehende Zugriffsmöglichkeiten auf das Rechnersystem und die Kommunikation des Angegriffenen verfügt. Zudem dürfte es ihm schwerfallen, andere Effekte des modifizierten Schlüssels (wie das Fehlschlagen von Entschlüsselungen im Fall von RSA) zu vertuschen.

Benutzer, die PGP ausschließlich zum Verschlüsseln und nicht zum Signieren einsetzen, sind ohnehin nicht bedroht.

Aus diesen Gründen ist die Bedeutung der Attacke für den täglichen Einsatz von PGP eher gering einzustufen.

2.1 Plausibilität

Die wichtigste Voraussetzung für den Angreifer ist, dass er schreibend auf die „secring.skr“-Datei des Opfers zugreifen kann. Dies erscheint plausibel, wenn der geheime Schlüssel von seinem Eigentümer z. B. per E-Mail auf ein anderes System transferiert wird.

Einem Angreifer hingegen, der Schreibrechte auf dem Arbeitsplatzsystem seines Opfers hat, eröffnen sich auch andere, z. T. sicherlich einfachere Möglichkeiten zum Angriff, z. B. die Installation eines Paßwort-Sniffers oder einer passend „gepatchten“ Version von PGP.

2.2 Erkennung

Nach der Modifikation des geheimen Schlüssels ist dieser nicht mehr zum korrekten Entschlüsseln von Nachrichten zu gebrauchen. Auch die damit erzeugten Signaturen sind ungültig. Dies ermöglicht es dem Opfer, den Angriff zu erkennen.

2.3 „Original"-PGP

Das mittlerweile von Network Associates vertriebene „Original"-PGP unterstützt sowohl das DSA- als auch das RSA-Verfahren.

Während es für die DSA-Attacke anfällig ist, verhindern zusätzliche, über die Anforderungen des OpenPGP-Standards hinausgehende interne Checks des PGP-Programms die beschriebene Attacke auf RSA-Schlüssel.

3 Abwehr

Letzendlich muss die Reaktion auf die durch Klíma und Rosa aufgedeckten Schwächen des Datenformats in einer Überarbeitung des OpenPGP-Standards bestehen.

Bis dahin können sowohl die Benutzer als auch die Hersteller und Autoren von OpenPGP-konformer Software weitere effektive Gegenmaßnahmen ergreifen.

3.1 Benutzer

PGP-Benutzer sollten aus Sicherheitsgründen auch unabhängig von der aktuellen Attacke dafür sorgen, dass ihr geheimer Schlüssel nicht unbefugt verändert werden kann. Eine gute Praxis ist es, die Datei mit dem Schlüssel auf einer schreibgeschützten Diskette aufzubewahren und diese nur für die Dauer der PGP-Operationen einzulegen. Möglichkeiten des Betriebssystems, die Zugriffsrechte auf Dateien einzuschränken, wie sie z. B. Unix-Derivate oder Windows NT anbieten, sollten selbstverständlich wahrgenommen werden.

Zusätzlich sollten Benutzer regelmäßig prüfen, ob die Signaturen, die sie erzeugen, auch gültig sind. Fehlgeschlagene Signaturprüfungen oder Entschlüsselungsoperationen sind ein Alarmsignal.

Dementsprechend kann man es als einen Akt der Höflichkeit ansehen, dem Absender einer PGP-gesicherten Nachricht mitzuteilen, falls die Prüfung seiner Signatur fehlschlägt.

3.2 Hersteller

Die Hersteller bzw. Autoren von Verschlüsselungssoftware werden aus nachvollziehbaren Gründen davor zurückschrecken, proprietäre Ad-hoc-Änderungen am Dateiformat für den geheimen Schlüssel vorzunehmen.

Ihnen bleibt als kurzfristige Lösung, ihre Software so zu ergänzen, dass nach der Erzeugung einer Signatur diese erst intern auf Gültigkeit geprüft wird. Schlägt diese interne Prüfung fehl, so wird anstelle der Signatur eine Warnung ausgegeben, dass möglicherweise der geheime Schlüssel manipuliert wurde.

Ein entsprechender Patch ist z. B. für GnuPG bereits verfügbar.

Fazit

Obwohl die Attacke von zwei Kryptologen entwickelt wurde und hervorragenden Gebrauch von kryptanlaytischen Methoden macht, ist sie im Kern kein Angriff auf die von OpenPGP verwendeten kryptographischen Verfahren, sondern auf ein unzureichend geschütztes Datenformat.

Die praktische Bedeutung der Attacke ist verglichen mit den ersten Meldungen eher gering. Deutlich schwerer wiegt demgegenüber die Frage, wie die nun aufgedeckten Schwächen des Dateiformats während des Standardisierungsprozesses und mehr als zwei Jahre danach unentdeckt bleiben konnten.

Es ist zu erwarten, dass Gegner wie Befürworter von offenen Standards und Open-Source-Software im Sicherheitsbereich das Ergebnis jeweils in ihrem Sinne interpretieren werden.

Während die einen argumentieren können, dass trotz OpenSource viele Augen die Sicherheitsschwächen jahrelang übersehen haben, bleibt den anderen die Feststellung, dass die Lücken dank OpenSource letztlich doch entdeckt, publiziert und somit auch geschlossen werden konnten.

> Es bleibt schwierig, gute Sicherheitsstandards zu entwerfen. Auch viele Augen sehen Lücken manchmal erst (zu?) spät.

Literatur

[BDL96] D. Boneh, R. A. DeMillo, R. J. Lipton, „On the Importance of Checking Computations", Extended Abstract, 1996. http://jya.com/smart.pdf

[BS96] E. Biham, A. Shamir, „Identifying the Structure of Unknown Ciphers Sealed in Tamper-Proof Devices" (Draft), Research Announcement, 10.11.96.

[CDFT98] J. Callas, L. Donnerhacke, H. Finney, R. Thayer, „RFC 2440: OpenPGP Message Format", November 1998. ftp://ftp.isi.edu/in-notes/rfc2440.txt

[F97] D. Fox, „Fälschungssicherheit digitaler Signaturen", Datenschutz und Datensicherheit (DuD), 2/1997, S. 69-74.

[G01] J. O. Grabbe, „The DSA Flaw in OpenPGP", 2001. http://orlingrabbe.com/DASflaw_OpenPGP.htm.

[KR01] V. Klíma, T. Rosa, „Attack on Private Signature Keys of the OpenPGP format, PGP™ programs and other applications compatible with OpenPGP", März 2001. http://www.i.cz/en/pdf/openPGP_attack_ENGvktr.pdf

[NIST00] National Institute of Standards and Technology (NIST), „Digital Signature Standard (DSS)", Federal Information Processing Standards Publication 186-2 (FIPS-PUB), 27.01.2000. http://csrc.nist.gov/publications/fips/fips186-2/fips186-2.pdf