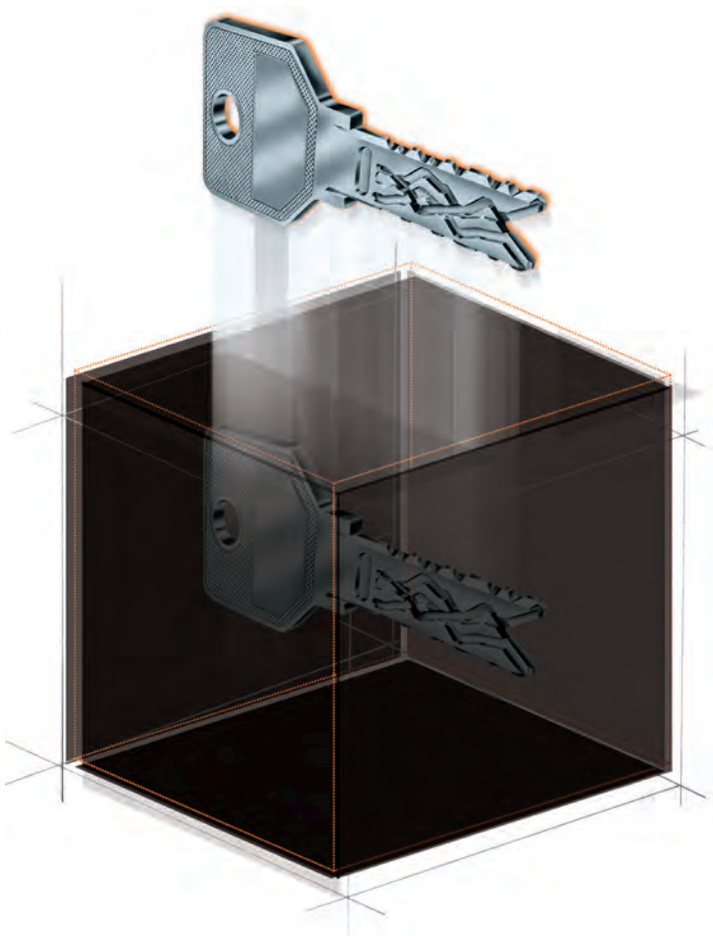

Die dunkle Seite der Kryptografie

Kleptografie bei Black-Box-Implementierungen

Hardware-Security-Module (HSM) und andere Black-Box-Implementierungen sollen wichtige Krypto-Schlüssel sicher kapseln. Doch bei entsprechender Implementierung kann die Kryptografie selbst als verdeckter Kanal missbraucht werden, um vermeintlich Unkopierbares aus dem Schlüsselspeicher zu schmuggeln.

Von Bernhard Esslinger, Siegen

Obwohl bereits seit Mitte der Neunzigerjahre bekannt [1], wurde der Kleptografie bisher keine große Aufmerksamkeit geschenkt. Dabei handelt es sich keineswegs nur um eine theoretische Gefahr, sondern inzwischen wurden mehrere kleptografische Angriffe beispielhaft implementiert. Ein grundlegender Aspekt von Black-Box-Kryptosystemen wird hierbei zum Problem: Hardware-Security-Module (HSM), Smartcards oder auch Trusted-Platform-Modules (TPMs) sollen kryptografische Inhalte sicher versiegeln und ihren Inhalt vor jeglichem Zugriff von außen schützen. Somit bleibt aber auch der Besitzer ein Stück weit im Ungewissen und kann sich nicht sicher sein, ob die Black Box genau das macht, was sie soll – oder doch etwas mehr?!

Wenn die Manipulation eines solchen Kryptosystems von außen nicht erkennbar ist und auch die generierten Ausgaben keinen Verdacht auslösen, kann man letztlich nur auf den Hersteller vertrauen. Dabei können sich (nicht nur eingefleischte) Verschwörungstheoretiker auch die Einflussnahme von Regierungsorganisationen auf manche Hersteller vorstellen.

Kleptografie

Kleptografie ist das Studium vom sicheren und unterschwelligen Stehlen von Informationen („study of stealing information securely and subliminally“ [2]). Unter einem kleptografischen Angriff versteht man einen Angriff, bei dem ein böswilliger Entwickler eine kryptografische Hintertür implementiert, bei der er asymmetrische Verschlüsselung nutzt. Hier wird Kryptografie gegen Kryptografie eingesetzt – es geht also nicht um eine Hintertür, die außerhalb des Kryptosystems einen zusätzlichen Kanal oder zusätzliche Datenübertragungen eröffnet, sondern um „eingebaute“ Missbrauchsmöglichkeiten innerhalb der vorgesehenen Kommunikation. Kleptografie gehört zum Themengebiet der Kryptovirolgie, die sich allgemein mit der Anwendung von Kryptografie in Schadsoftware beschäftigt [3,6,11] – bei der Kleptografie ist dann keine allgemeine Software, sondern ein Kryptosystem Ziel des Angriffes.

Das folgende Szenario beschreibt einen möglichen kleptografischen Angriff: Eine Black-Box generiert asymmetrische Schlüsselpaare, jeweils bestehend aus einem geheimen (Private Key) und einem öffentlichen Schlüssel (Public Key). Die geheimen Schlüssel, die zum Entschlüsseln und zur Signaturerstellung dienen, sollen ausschließlich in der Black Box verbleiben, um sie gegen missbräuchliche Nutzung und Kopie zu schützen – nur die öffentlichen Schlüssel werden exportiert. Wie allgemein bekannt, kann niemand aus den öffentlichen Schlüsseln die zugehörigen geheimen Schlüssel berechnen – oder doch?

Tatsächlich wird dies möglich, sofern bei der Schlüsselgenerierung entsprechende manipulative Schritte erfolgt sind. Bei der Implementierung könnte eine kryptografische Hintertür eingebaut worden sein,

mit deren Hilfe der Angreifer im Nachhinein die erzeugten geheimen Schlüssel ermitteln kann – und niemand merkt etwas: weder sind die erzeugten öffentlichen Schlüssel irgendwie auffällig, noch gibt es irgendwelche unerwartete Kommunikation oder Fehler bei der Nutzung der kryptografischen Funktionen. Die Auswirkungen sind jedoch fatal: Mit dem „geklonten" geheimen Schlüssel kann der Angreifer Daten entschlüsseln oder Signaturen fälschen, obwohl der Kryptoschlüssel in der abgeschotteten Black Box generiert wurde und darin weiterhin vor unberechtigtem Zugriff geschützt ist.

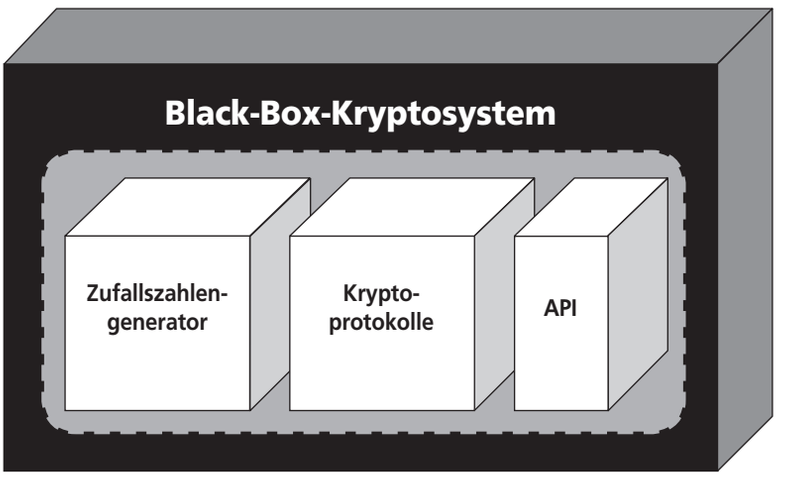

Abbildung 1: Kleptografische Angriffe sind in allen dargestellten Komponenten eines Kryptosystems möglich – die Kapselung einer Black-Box-Implementierung schützt dabei gleichermaßen die gewünschte wie die missbräuchliche Implementierung gegen den Zugriff von außen.

In einem vereinfachten Angriff könnte eine Manipulation im Zufallszahlengenerator erfolgen (s. a. Abb. 1), sodass bei der Generierung von Schlüsseln statt echter Zufallszahlen eine Pseudozufallsfunktion mit einem – dem Angreifer bekannten – Startwert (Seed) benutzt wird: Durch das Wissen um die erzeugten Primzahlen könnte dieser dann „parallel" ebenfalls die geheimen Schlüssel berechnen. Eine derartige Manipulation kann im Prinzip durch Reverse-Engineering entdeckt werden, sofern keine Sicherheitsmechanismen (wie bei der Implementierung in dedizierter Kryptohardware) den Zugriff auf den Code unmöglich machen. Da der Startwert der Pseudozufallsfunktion im Code verankert ist, könnte dann auch der Entdecker die geheimen Schlüssel nachberechnen – bei „bes-

seren“ Angriffen (s. u.), ist dies jedoch nicht mehr der Fall.

SETUP-Angriff

Erstmals diskutiert wurde Kleptografie bereits 1996 auf der CRYPTO-Konferenz, als Adam Young und Moti Yung auf zahlreiche Angriffsmöglichkeiten gegen Kryptografie in Black-Box-Systemen aufmerksam machten [1]. Bereits damals wurde der Begriff „Secretly Embedded Trapdoor with Universal Protection“ als SETUP-Angriff eingeführt und im Hinblick auf die RSA-Schlüsselerzeugung beschrieben. Charakteristisch ist, dass ein SETUP-Angriff (wenn überhaupt) nur durch Reverse-Engineering zu entdecken ist, aber der Entdecker den Angriff dennoch nicht selbst nutzen kann: Der Reverse-Engineer wird im Allgemeinen nur den öffentlichen Schlüssel, aber nicht den geheimen Schlüssel des Angreifers finden, sodass dieser auf asymmetrischer Kryptografie basierende Angriff als „sicher“ bezeichnet werden darf – aus Sicht des Angreifers. Das mathematische Prinzip einer Version des SETUP-Angriffes auf die RSA-Schlüsselgenerierung ist im nebenstehenden Kasten skizziert. Zum besseren „Kennenlernen“ sind zudem mehrere kleptografische Angriffe im Lernprogramm JCrypTool implementiert (siehe Kasten auf S. 10, vgl. Abb. 2).

Im Laufe der Jahre wurde der SETUP-Angriff weiterentwickelt: Die ersten Attacken zielten auf Verfahren, die auf dem Problem der Primfaktorzerlegung (z. B. RSA) beruhen. Bald wurden jedoch auch Angriffe auf Verfahren veröffentlicht, die auf dem Diskreten-Logarithmus-Problem basieren: So wurde 2002 ein starker SETUP-Angriff auf den Diffie-Hellman-Schlüsselaustausch beschrieben [2]. Im Dezember 2009 wies Moti Yung auf dem 26. Chaos Communication Congress (26C3) mit seinem Vortrag „Yes We Can‘t!“ [5] erneut auf die Problematik hin und betonte, dass Kleptografie das Vertrauen in den Hersteller einschränkt: „I will discuss some of the results and their influence on the limitation of the notion of ‚trust‘ in systems...“, so Yung in Berlin.

Gegenmaßnahmen

Doch warum wurde der Kleptografie bisher kaum Aufmerksamkeit geschenkt, obwohl der Einsatz von Hardware-Kryptosystemen doch gerade dort erfolgt, wo sehr hohe Sicherheitsanforderungen gestellt werden? Schließlich werden etwa HSMs benutzt, um besonders sensitive Infrastrukturschlüssel in Unternehmen zu schützen – also dort, wo der mögliche finanzielle und reputative Schaden bei einem erfolgreichen Angriff auf diese Schlüssel sehr hoch ist. Möglicherweise ist die grundlegende Problematik noch nicht hinreichend bekannt? Eine andere Erklärung könnte die Annahme der Nutzer sein, dass der Hersteller ein Interesse daran haben müsste, keine manipulierte Hardware zu verkaufen – denn er würde wohl vom Markt verschwinden, wenn so etwas „auffliegt“.

Bei sicherheitsrelevanter Industriehardware in der EU wird zudem auf unabhängig voneinander durchgeführte Evaluationen in zwei verschiedenen EU-Staaten gesetzt,

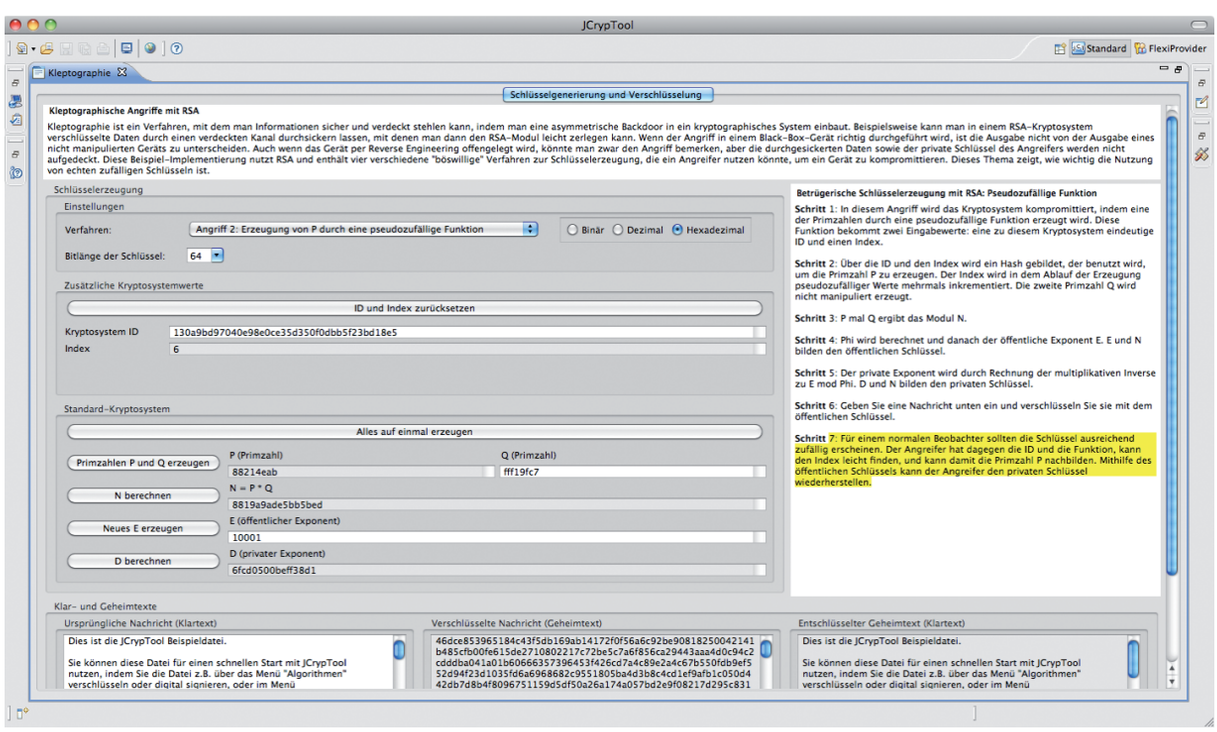

Abbildung 2: Für das E-Learning-Werkzeug „JCrypTool“ existiert ein Plugin, das die Möglichkeiten der Kleptografie verdeutlicht.

um eine möglichst hohe Transparenz über die gesamte Produktion zu erlangen. Zudem gibt es den Ansatz, Hardware verschiedener Hersteller kombiniert einzusetzen: So könnte man beispielsweise zwei Smartcards verschiedener (unabhängiger!) Hersteller benutzen und Dateien zweimal nacheinander verschlüsseln (Kaskadenverschlüsselung [6]). Somit könnte ein Hersteller allein, selbst wenn er die Schlüsselgenerierung manipuliert hätte, keine Daten entschlüsseln, da er den zweiten geheimen Schlüssel der anderen Smartcard nicht kennt. Doch mit diesen beiden Ansätzen kann man das Risiko lediglich minimieren

Einfacher SETUP-Angriff auf RSA-Schlüsselgenerierung

Zum Vergleich beziehungsweise zur Erinnerung hier zunächst der Ablauf einer **normalen RSA-Schlüsselgenerierung** (z. B. für einen 2048-Bit-Schlüssel):

- wähle zufällig zwei Primzahlen $p \neq q$ (hier z. B. jeweils 1024 Bit lang) – in der Praxis werden dazu Zufallszahlen erzeugt und anschließend Primzahltests durchgeführt.
- bestimme n (= $p \cdot q$) und e (meist wird $2^{16}+1$ gewählt)
- bestimme d: über die Kongruenz $e \cdot d \equiv 1 \bmod \varphi(n)$ – dazu wird der erweiterte euklidische Algorithmus verwendet: Da p und q prim sind, ist $\varphi(n) = (p\text{-}1) \cdot (q\text{-}1)$.

Im Ergebnis erhält man den öffentlichen Schlüssel (n, e) und den geheimen Schlüssel (d). Die **RSA-Ver- und Entschlüsselung einer Nachricht m** ($m \in Z_n^*$) erfolgt beim RSA-Verfahren durch Potenzierung:

- Verschlüsselung: $c = m^e \bmod n$
- Entschlüsselung: $m = c^d \bmod n$

Die **kleptografische RSA-Schlüsselgenerierung** (hier zur Darstellung in einer vereinfachten Version aus [4]) erfolgt modifiziert und nutzt den öffentlichen Teil eines RSA-Schlüssels des Angreifers (N, E). Zu beachten ist, dass dieser Angreifer-Schlüssel halb so lang ist (z. B. 1024 Bit) wie der anzugreifende Schlüssel (z. B. 2048 Bit):

- wähle eine Zufallszahl s (z. B. 1024 Bit lang) und berechne $p = H(s)$, wobei H eine kryptografische Einwegfunktion ist – das wird so lange wiederholt, bis p prim ist.
- verschlüssele s mit dem öffentlichen Schlüssel (N, E) des Angreifers: $c = s^E \bmod N$
- wähle eine Zufallszahl z
- bestimme q so, dass die Gleichung $c \| z = p \cdot q + r$ mit beliebigem r erfüllt wird – ist q nicht prim, beginne erneut mit einer anderen Zufallszahl s
- bestimme n (= $p \cdot q$), e (meist wird $2^{16}+1$ gewählt) und d (siehe oben)

Das Ergebnis ist erneut ein öffentlicher Schlüssel (n, e) und ein geheimer Schlüssel (d).

Das **Aufdecken des geheimen Schlüssels** ist nun mit Kenntnis des per kleptografischer RSA-Schlüsselgenerierung erzeugten öffentlichen Schlüssels (n, e) sowie des geheimen Angreifer-Schlüssels (D) möglich:

- nehme die obersten $n/2$ Bits von n als u (hier z. B. die ersten 1024 Bit)
- setze $c_1 = u$ und $c_2 = u + 1$ (c_2 wird benötigt, da bei der Berechnung von $n = p \cdot q = (c \| z) - r$ ein Bit „verloren gehen" kann)
- entschlüssele c_1 und c_2 mit dem geheimen Schlüssel (D) des Angreifers, um s_1 und s_2 zu erhalten: $s_1 = c_1^D \bmod N$ und $s_2 = c_2^D \bmod N$
- berechne $p_1 = H(s_1)$ und $p_2 = H(s_2)$
- berechne $q_1 = n/p_1$ und $q_2 = n/p_2$ – der Quotient, der n ohne Rest teilt, ergibt q und das zugehörige p_i ergibt p
- bestimme d (siehe oben)

Im Ergebnis erhält der Angreifer den geheimen Schlüssel (d) des Opfers.

JCrypTool: E-Learning für Kryptografie

Unter dem Namen JCrypTool (JCT) wird seit 2007 im Rahmen eines Open-Source-Projekts an einem der beiden offiziellen Nachfolger von „CrypTool“ gearbeitet, dem weltweit verbreitetesten Lernprogramm zu Kryptografie und Kryptoanalyse (vgl. www.cryptool.org sowie <kes> 2008#3, S. 62).

JCrypTool basiert auf der Eclipse „Rich Client Platform“ (RCP) und auf Java, ist somit betriebssystemunabhängig und kann sehr einfach von jedem durch Plugins erweitert werden. In JCrypTool ist auch ein Plugin integriert, mit dem man mehrere einfache kleptografische Angriffe inklusive einem SETUP-Angriff durchspielen kann (vgl. Abb. 2).

JCrypTool ist schnell einsatzbereit: einfach auf http://sourceforge.net/projects/jcryptool dem Download-Link zur gewünschten Plattform folgen und dann das heruntergeladene Zip-Archiv entpacken. Das Programm lässt sich dann sofort über die jeweilige ausführbare Datei (unter Windows: JCrypTool.exe) starten. JCrypTool ermittelt automatisch die Sprache des Betriebssystems und wird entsprechend mit deutscher oder englischer Oberfläche gestartet.

Beim ersten Lauf präsentiert die Welcome-Page eine Kurztour durch das Tool: Hier finden sich vor allem für Einsteiger gedachte Informationen und Links auf tiefer gehende Tutorials.

Durch Schließen der Einführung gelangt man in die Standard-Perspektive (siehe Abb. 2, hier mit maximiert geöffneter Kleptografie-Sicht). Diese Standard-Perspektive ist dateiorientiert, das heißt der Anwender wählt zunächst die gewünschte Datei aus (z. B. die JCrypTool-Beispieldatei oder eine beliebige eigene Datei) und anschließend den darauf auszuführenden Algorithmus (z. B. den symmetrischen AES-Algorithmus). Eine kryptografische Operation in JCrypTool verändert dabei niemals die Originaldatei, sondern generiert immer eine neue Datei.

Um die Dateiauswahl zu erleichtern, befindet sich auf der linken Fensterseite der „Datei-Explorer“. Auf der rechten Seite zeigt die Algorithmen-Sicht alle in JCrypTool vorhandenen Algorithmen, Analysen, Spiele und Visualisierungen.

Nutzungsmöglichkeiten

Kryptografische Operationen werden in JCrypTool entweder durch Assistenten unterstützt oder mithilfe von Sichten visualisiert: So starten in der Algorithmen-Sicht (auf dem ersten Tab „Algorithmen“) derzeit alle Einträge der Assistenten, die Schritt für Schritt beispielsweise zur Verschlüsselung des aktiven Editorinhalts führen. Viele Dialoge verfügen über eine kontextsensitive Hilfe, die über die F1-Taste oder durch Klick auf das „?“ links unten eingeblendet werden kann.

Die Einträge auf den weiteren Tabs (Analysen, Visualisierungen und Spiele) öffnen jeweils Sichten, in denen weiter gehende Operationen ausgeführt werden können.

In der Standard-Perspektive ist „hinter“ dem Datei-Explorer die Actions-Sicht angeordnet: Diese Sicht arbeitet nach der Aktivierung des Aufzeichnen-Buttons als eine Art Rekorder und speichert durchgeführte kryptografische Operationen. So kann man Krypto-Kaskaden aufzeichnen (beispielsweise die mehrfache Durchführung von Substitutions- und Transpositions-Verschlüsselungen) und anschließend wiederholt ausführen oder mit anderen Anwendern austauschen.

Mit einer via Icon erreichbaren Konsole lassen sich kryptografische Operationen zudem auch über eine Art Kommandozeile eingeben: So können Dateien schnell ohne die Unterstützung von Assistenten ver- oder entschlüsselt werden. Die Eingabe „help“ zeigt, welche Operationen in der Konsole derzeit bereits möglich sind.

Neben der Online-Hilfe für Benutzer und Entwickler findet man aktuelle Informationen auf der SourceForge-Projektseite unter http://sourceforge.net/projects/jcryptool und auf der Homepage unter http://jcryptool.sourceforge.net – dort steht neben einem Diskussionsforum auch eine Mailingliste für Benutzer zur Verfügung.

Mitmachmöglichkeiten

JCrypTool kann von jedem interessierten Entwickler mit eigenen Plugins erweitert werden: Eclipse-Plugins sind spezialisierte Programme, die an die von JCrypTool vorgegebenen Punkte „andocken“ können und so die Plattform erweitern. Eigene Plugins können dabei völlig unabhängig vom JCrypTool-Team entwickelt werden – bei Interesse ist jedoch jederzeit eine Integration in das Projekt „JCrypTool Plugins“ möglich.

Zusätzliche Unterstützung finden interessierte Entwickler zum JCrypTool-Plugins-Projekt unter http://sourceforge.net/projects/jctplugins, im Wiki unter http://jcryptool.wiki.sourceforge.net und in der Entwickler-Mailingliste.

– eine Garantie gegen Manipulationen bieten sie jedoch nicht und der erhöhte Aufwand macht sie eher unpraktikabel.

Da bei der Kleptografie ein verdeckter Kanal genutzt wird, um Informationen unentdeckt aus der Black Box zu schmuggeln, könnte man überdies versuchen, alle möglichen verdeckten Kanäle zu eliminieren: Erste Ideen dazu wurden 1984 von Gus Simmons [7] sowie später auch in weiteren Publikationen [8,9] vorgestellt, indem man etwa bei Authentisierungsprotokollen Zufallszahlen einbaut. Konkretere Ansätze wurden beispielsweise 2002 mit einem Verfahren vorgestellt, bei dem ein Dritter die RSA-Schlüsselgenerierung verifizieren kann [10]: Dabei kommt eine Art verteilte Schlüsselgenerierung zum Einsatz, wobei der geheime Schlüssel aber nur der Black Box bekannt ist – somit wäre sichergestellt, dass die Schlüsselgenerierung nicht modifiziert wurde und der Schlüssel nicht durch einen kleptografischen Angriff aufgedeckt werden kann. Zumindest die Forschung sucht also durchaus aktiv nach Möglichkeiten, um die Bedrohung durch Kleptografie zu bekämpfen.

Fazit

Kleptografie ist kryptografisch sehr interessant, aber in der Praxis nur eine von mehreren Bedrohungen im gesamten System: So ist es zumeist einfacher, nicht die Krypto-Implementierung selbst, sondern andere beteiligte Komponenten anzugreifen – etwa am Endpunkt. So könnte man zum Beispiel vertrauliche Daten auf einem PC mit einem Trojaner abfangen, bevor sie überhaupt mit einer Smartcard verschlüsselt werden. Ein System ist immer nur so sicher wie das schwächste Glied in der Kette – Kleptografie und ihre Abwehr werden daher vermutlich erst dann deutlich an Bedeutung gewinnen, wenn derartige Angriffe auf andere Komponenten zu schwer werden.

Man sollte aber nicht vergessen, dass sich bei besonders hohen Sicherheitsanforderungen möglicherweise schon heute der Aufwand lohnen könnte. Und auch wenn sich dieses Beispiel nicht 1:1 auf abgeschottete Black-Box-Systeme übertragen lässt, hat das Auftreten einer manipulationsanfälligen Firmwareupdatefunktion in einem sicherheitsevaluierten Chipkartenterminal (http://colibri.net63.net/Smartcard-Reader%20von%20Kobil%geknackt.pdf) im Frühsommer 2010 doch eindringlich gezeigt, dass es auch im stark regulierten Bereich nicht unbedingt der Hersteller sein muss, der Hintertüren einbaut und damit seine Existenz aufs Spiel setzt. ■

Prof. Bernhard Esslinger lehrt IT-Security und Kryptografie an der Uni Siegen und leitet das Open-Source-Projekt CrypTool.

Literatur

[1] A. Young, M. Yung, The Dark Side of Black-Box Cryptography, or: Should we trust Capstone?, in: N. Koblitz (Ed.), Advances in Cryptology – Crypto '96, LNCS 1109, Springer, 1996, ISBN 978-3-540-61512-5

[2] A. Young, M. Yung, Kleptography: Using Cryptography Against Cryptography, in: W. Fumy (Ed.), Advances in Cryptology – Eurocrypt '97, LNCS 1233, Springer, 1997, ISBN 978-3-540-62975-7

[3] A. Young, M. Yung, Cryptovirology FAQ, Version 1.31, http://www.cryptovirology.com/cryptovfiles/cryptovirologyfaqver1.html

[4] M. Yung, Kleptography: The Outsider Inside Your Crypto Devices, and its Trust Implications, DIMACS Workshop on Theft in E-Commerce: Content, Identity, and Service, 2005, Powerpoint-Präsentation: http://dimacs.rutgers.edu/Workshops/Intellectual/slides/yung.ppt

[5] M. Yung, Yes We Can't, 26th Chaos Communication Congress, 2009, MPEG-4-Video-Aufzeichnung: http://events.ccc.de/congress/2009/Fahrplan/events/3702.en.html

[6] A. Young, M. Yung, Malicious Cryptography: Exposing Cryptovirology, John Wiley & Sons, 2004, ISBN 978-0-7645-4975-5

[7] G. J. Simmons, The Prisoners' Problem and the Subliminal Channel, in: D. Chaum (Ed.), Proceedings of Crypto '83, Plenum Press, 1984, ISBN 978-0-306-41637-8

[8] G. J. Simmons, The Subliminal Channel and Digital Signatures, in: T. Beth, N. Cot, I. Ingemarsson (Eds.), Proceedings of Eurocrypt '84, LNCS 209, Springer-Verlag, 1985, ISBN 978-3-540-16076-2

[9] Y. Desmedt, C. Goutier, S. Bengio, Special Uses and Abuses of the Fiat-Shamir Passport Protocol, in: C. Pomerance (Ed.), Proceedings of Crypto '87, LNCS 293, Springer-Verlag, 1988, ISBN 978-3-540-18796-7

[10] A. Juels, J. Guajardo, RSA Key Generation with Verifiable Randomness, in: D. Naccache, P. Pallier (Eds.), Public Key Cryptography: 4th International Workshop on Practice and Theory in Public Key Cryptosystems, Springer-Verlag, 2002, ISBN 978-3-540-43168-8

[11] A. Yung, M. Young, Towards a Book on Advances in Cryptovirology, Selected Chapters as PDF, www.cryptovirology.com/cryptovfiles/newbook.html

Sind Sie verantwortlich für die IT-Sicherheit?

<kes> liefert alle relevanten Informationen zum Thema IT-Sicherheit – sorgfältig recherchiert von Fachredakteuren und Autoren aus der Praxis.

In jeder Ausgabe finden Sie wichtiges Know-how, Hinweise zu Risiken und Strategien, Lösungsvorschläge und Anwenderberichte zu den Themen:

Internet/Intranet-Sicherheit
Zutrittskontrolle
Virenabwehr
Verschlüsselung
Risikomanagement
Abhör- und Manipulationsschutz
Sicherheitsplanung
Elektronische Signatur und PKI

<kes> ist seit 20 Jahren *die* Fachzeitschrift zum Thema Informations-Sicherheit - eine Garantie für Zuverlässigkeit.

Jetzt Probeheft anfordern!

<kes>-online

<kes>-Leser können neben der Print-Ausgabe auch <kes>-online unter www.kes.info nutzen. Hier finden Sie ohne Zugangsbeschränkung, das Thema der Woche, viele interessante Links, Stichwort-Lexikon IT-Security-Begriffe, Verzeichnis relevanter Veranstaltungen und außerdem aktuelle Artikel zum Probelesen.

Abonnenten erhalten zusätzlich ein Passwort mit dem sie Zugriff auf alle aktuellen Artikel und auch auf das Online-Archiv erhalten.

PROBEHEFT-ANFORDERUNG

☐ **ja,** bitte schicken Sie mir gratis und unverbindlich ein Exemplar der <kes> - Die Zeitschrift für Informations-Sicherheit zum Probelesen zu.

Es kommt nur dann ein Abonnement zustande, wenn ich es ausdrücklich wünsche.

Das Abonnement beinhaltet ein Passwort zur Nutzung des Abo-Bereichs auf www.kes.info

Datum | Zeichen | Unterschrift

Lieferung bitte an

Telefon Durchwahl

FAX an +49 6725 5994

SecuMedia Verlags-GmbH
Abonnenten-Service
Postfach 12 34
55205 Ingelheim