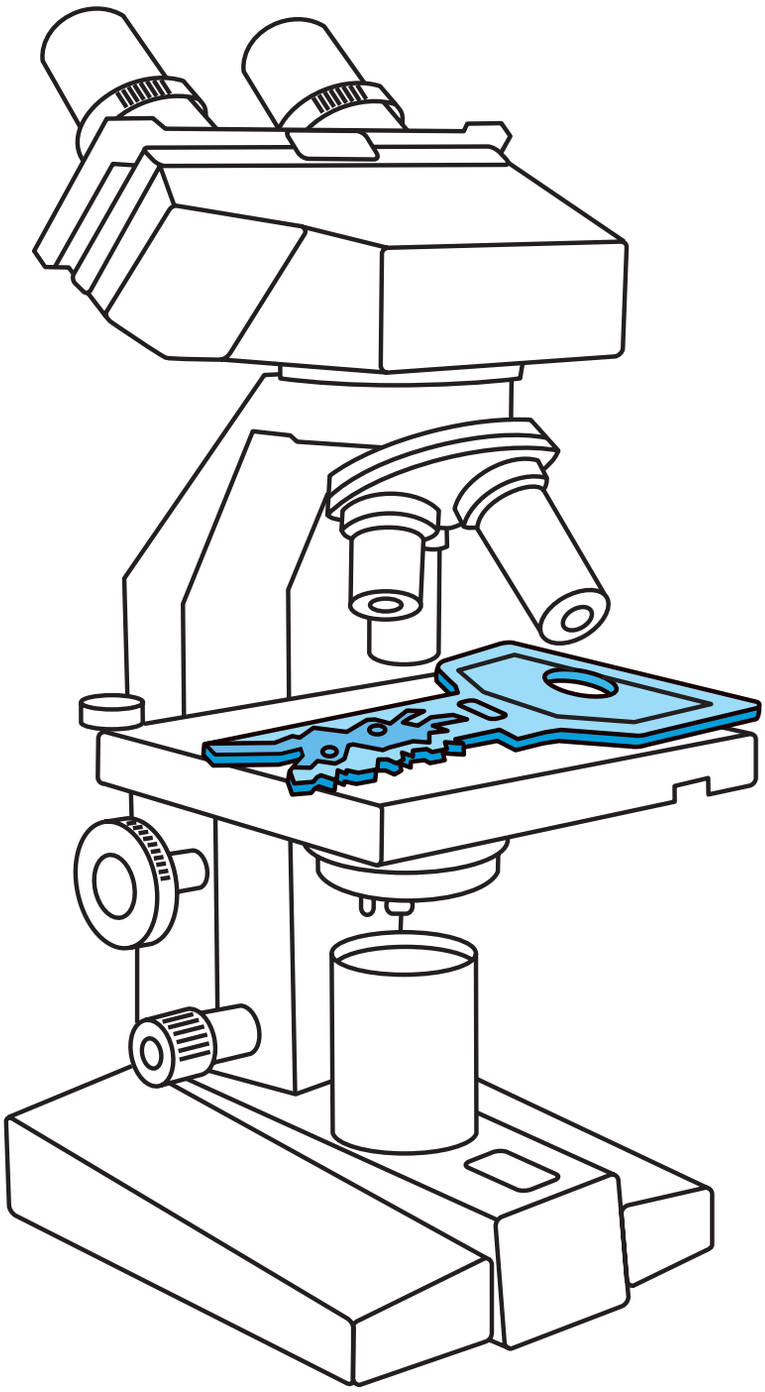

RSA-Sicherheit in der Praxis

Fehler in der Anwendung des RSA-Algorithmus

Nachdem Schweizer Kryptologen die Ergebnisse einer Feldstudie zu RSA-Schlüsseln veröffentlicht haben, ging ein Raunen durch die Security-Welt: Gefährden einige tausend identische Primfaktoren in Public-Key-Zertifikaten die Vertrauenswürdigkeit von SSL, Onlinebanking und Trust-Centern? Wie steht es um die Sicherheit von RSA-Keys bei Unternehmen? Der vorliegende Beitrag hilft, die Ergebnisse aus Lausanne einzuordnen, und liefert neue Erkenntnisse aus dem Firmenumfeld.

Von Bernhard Esslinger, Siegen, Volker Simon und Jörg Schneider, Frankfurt/Main

Mitte Februar haben Forscher der École Polytechnique Fédérale de Lausanne (EPFL) in Lausanne (http://lacal.epfl.ch/) eine Feldstudie veröffentlicht: Ihre Arbeit [1] bezeichneten sie selbst als Plausibilitätsprüfung (Sanity-Check), ob bei der Schlüsselgenerierung genug Zufälligkeit benutzt wird. Sie hatten über 11 Millionen RSA-Moduli untersucht, die alle öffentlich zugänglich und unter anderem vom SSL-Observatory (www.eff.org/observatory) gesammelt worden waren. Die Ergebnisse waren in mehrfacher Hinsicht brisant – allem voran, da die Voraussetzungen für eine korrekte Schlüsselgenerierung und eine sorgsame Zertifikatserstellung den Fachleuten seit Jahren bekannt sind.

Dieser Artikel konzentriert sich auf die in der Studie untersuchten RSA-Schlüssel (also nicht DSA-Keys) und ergänzt konkrete Gründe für die aufgetretenen Fehler sowie anonymisierte Ergebnisse einer eigenen, nichtöffentlichen Untersuchung von rund 100 000 RSA-Schlüsseln, die nicht für die öffentliche Verwendung, sondern zum Einsatz innerhalb von Unternehmen erzeugt wurden.

Erläuterungen zum EPFL-Paper

„Unsicher“ kann im Kontext der betrachteten RSA-Schlüssel unterschiedliche Ausprägungen haben

(aus eher mathematischer Sicht) – vier verschiedene Fälle werden im Folgenden näher erörtert.

Mehrfache Verwendung desselben RSA-Modulus

Über 70 000-mal haben die Schweizer festgestellt, dass derselbe RSA-Modulus mehrfach verwendet wird. Von 6.6 Millionen untersuchten, unterschiedlichen X.509-Zertifikaten und PGP-Schlüsseln mit RSA-Moduli hätten somit rund 4 % keinen „exklusiven" RSA-Modulus besessen – ein „Teilen" des Modulus mit Dritten sei häufig zu beobachten gewesen.

Ein möglicher Grund: Falls bei der Generierung der Primzahlen zu wenig Zufall mit einfließt, kann es passieren, dass derselbe Modulus bei unterschiedlichen Schlüsselgenerierungen erzeugt wird. Verwendet aber eine andere Person denselben Modulus, so kann diese den geheimen Schlüssel (Private Key) des anderen Nutzers einfach berechnen oder kennt ihn sogar schon, falls sie auch denselben öffentlichen Schlüssel hat.

Werden für beide Moduli Zertifikate vom gleichen Trust-Center erstellt, haben dessen Kontrollen versagt. In einem extremen Fall entdeckte die Studie sogar ein RSA-Schlüsselpaar, für das über 16 000 verschiedene X.509-Zertifikate existieren (für wie viele unterschiedliche Inhaber wurde allerdings nicht genannt).

In der Realität werden Moduli in Zertifikaten zudem auch wiederverwendet, wenn etwa Schlüssel von der User-Acceptance-Testing-(UAT)-Umgebung (Probebetrieb, Entwicklung, ...) in die Produktion überführt werden, anstatt neu generiert zu werden – oder wenn man beim Renewal (Verlängerung) in der Produktionsumgebung keine neuen Schlüssel generiert.

Moduli mit gemeinsamem, großem Primfaktor

Einen gemeinsamen, großen Primfaktor bei verschiedenen Moduli hat die Studie über 12 000-mal gefunden. Einige der untersuchten semi-primen RSA-Moduli haben genau einen ihrer beiden Primfaktoren mit anderen Moduli gemeinsam (semi-prime Moduli sind Zahlen, die das Produkt von genau zwei Primzahlen sind). In diesem Fall kann man den gemeinsamen Primfaktor einfach per Berechnung des größten gemeinsamen Teilers (ggT) bestimmen.

Von den untersuchten öffentlich genutzten 1024-Bit-RSA-Moduli, die weiterhin den weitaus größten Teil der gefundenen Moduli ausmachen, sind somit laut EPFL 0,2 % schlicht unsicher – das betreffe 21 419 X.509-Zertifikate und PGP-Schlüssel, deren „geheime Schlüssel für jeden zugänglich sind, der es auf sich nimmt, unsere Arbeit nachzuvollziehen" [1]. Die meisten dieser Schlüssel seien zwar abgelaufen, 5250 der Zertifikate seien jedoch noch gültig, benutzen eine zeitgemäße Hashfunktion und könnten daher durchaus noch in Gebrauch sein – 727 davon sind den Schweizern zufolge selbst-signiert und wurden auch benutzt, um andere RSA-Moduli zu signieren.

Neben dem inhaltlichen Ergebnis der Studie steckt eine beeindruckende Leistung der Forscher darin, dass sie den Vergleich aller Moduli miteinander für eine so große Anzahl von Moduli effizient durchführen konnten. Wenn man naiv jeden Modul eines neuen zu prüfenden Schlüssel mit allen existierenden Schlüsseln vergleicht, wächst die Komplexität quadratisch mit der Anzahl der Schlüssel. Die Forscher haben ihre angewandte Methode nicht veröffentlicht, jedoch gibt es eine schon im Jahre 2005 publizierte Methode von Dan Bernstein [2], welche effizient die Berechnung der ggT-Paare durchführt. Unter anderem nutzt Bernstein dabei eine Vorberechnung, bei

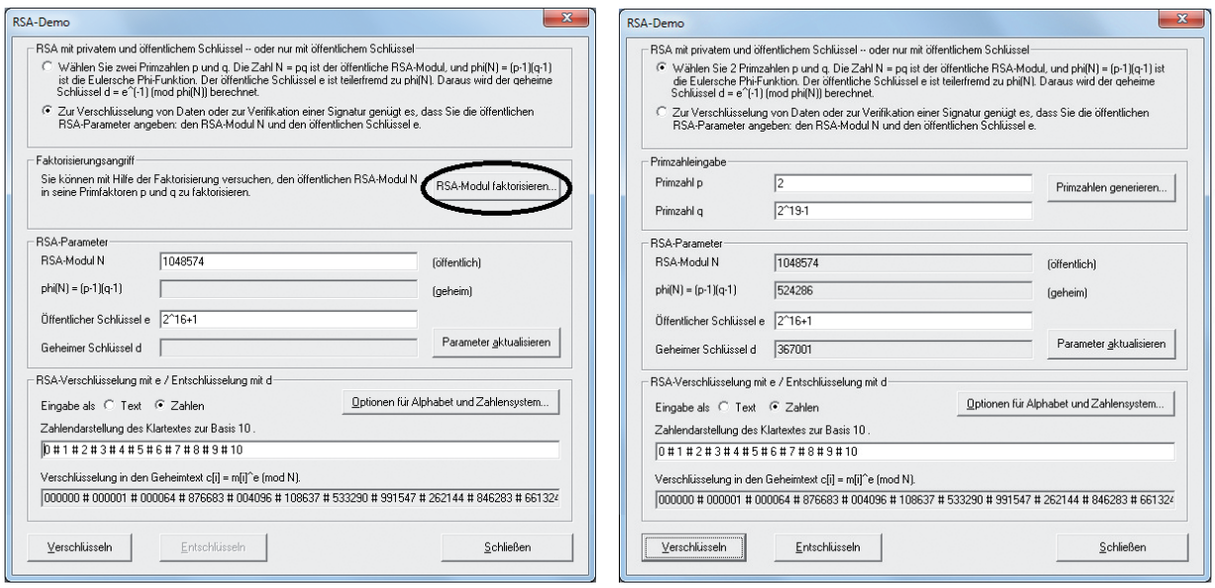

Abbildung 1: Was passiert, wenn man RSA-Schlüssel mit einem viel zu kleinen Primfaktor generiert, zeigen diese beiden Dialoge aus CrypTool 1.4.30 (www.cryptool.org/de/cryptool1) – links kann jeder mit dem öffentlichen Schlüssel verschlüsseln, rechts kann jeder (nicht nur der Eigentümer des geheimen Schlüssels) auch entschlüsseln.

der das Produkt aller Moduli gebildet wird – was wieder einmal eindrücklich zeigt, wie sinnvoll eine Vorberechnung beim „Knacken" kryptografischer Systeme sein kann (vgl. auch vorberechnete Rainbow-Tables [3] zum Finden eines Urbilds bei Hash-Funktionen).

Vorgehensweise des CrypTool-Projekts

Bei der Überprüfung von Public-RSA-Keys in Unternehmen ging das CrypTool-Projekt wie folgt vor:

1. Sammeln von Zertifikaten und Anonymisieren der Schlüssel

2. Extrahieren der Moduli aus den Zertifikaten über OpenSSL (www.openssl.org) mit dem Befehl:
`$ openssl x509 -in certificate.pem -noout –modulus`

Anmerkung: Hierbei wurde ein Bug in OpenSSL 1.0.1 festgestellt – beinhaltete das Zertifikat einen DSA-Schlüssel, wurde $g^x \bmod p$ ausgegeben, nicht etwa ein Fehler oder p, wie man bei einem DSA-Schlüsselpaar erwarten würde.

3. Paarweiser Vergleich aller Moduli miteinander mit einem Python-Skript, welches die gcd()-Funktion in dem frei verfügbaren Mathematik-Computer-Algebra-System Sage (www.sagemath.org) aufruft. Vor dieser Prüfung werden kleine Primfaktoren und gleiche Moduli identifiziert.

Anmerkung: Zukünftig wird Dan Bernsteins effizientere Methode [2] implementiert werden.

Moduli mit zu kleinen Primfaktoren

Haben semi-prime Moduli kleine Primfaktoren (z. B. 2, 17), sind auch Schlüssellängen von 2048-Bit nutzlos. Bei dieser vereinzelt beobachteten Schwachstelle lag wohl schon bei der Primzahlgenerierung ein Fehler vor: Es fehlte dabei nicht an Zufall, sondern die Generierungsfunktion ist dann völlig falsch implementiert. Primzahlen sind schließlich so zu wählen, dass sie eine Mindestgröße haben, bestimmte Bedingungen erfüllen und Primzahltests bestehen.

Eine andere Ausprägung von kleinen Primfaktoren trat auf, wo der Modul multi-prim ist (vgl. [4]) , also das Produkt von mehr als zwei Primzahlen – auch dann kann RSA funktionieren (wenn das bei der Schlüsselgenerierung berücksichtigt wird). Die Autoren glauben jedoch, dass es nicht beabsichtigt war, den Modul als Produkt von mehr als zwei Primzahlen zu bilden – wahrscheinlicher erscheint, dass hier ein Primzahltest versagt hat.

Öffentlicher Exponent ist 1

Ein einziges Mal in der untersuchten Stichprobe wurde als öffentlicher Schlüssel e=1 gesetzt – RSA wird damit zur Identitätsfunktion (eine Zahl wird immer auf sich selbst abgebildet), völlig unabhängig davon, wie gut die Primzahlen gewählt wurden.

Um die Betroffenen zu schützen, haben die Lausanner Krypto-Forscher nicht veröffentlicht, welche Schlüssel in PGP-Keys, in selbst-signierten Zertifikaten von Billig-Routern oder in durch öffentliche Trust-Center erstellten Zertifikaten gefunden wurden. Würden auch öffentliche Trust-Center solch schwache öffentliche Schlüssel signieren, wäre dies jedenfalls ein weiteres

Schlaglicht auf die unterentwickelte Qualitäts- und Risikomanagement-Kultur dieser Branche (so wie der Anfang Februar 2012 publik gewordene Fall, dass bewusst ein CA-Zertifikat zum Ausspionieren von SSL-Verbindungen verkauft wurde [5,6]).

Untersuchung von Firmen-Schlüsseln

Das CrypTool-Projekt (www.cryptool.org) wurde in der Folge der Schweizer Veröffentlichung gebeten, Firmen mit seinem Know-how zu unterstützen, um rund 100 000 Schlüssel zu prüfen, damit die Firmen feststellen konnten, ob ihre selbst erzeugten Schlüssel ähnliche Schwachstellen aufweisen.

Bei dieser Untersuchung (vgl. Kasten „Vorgehensweise...") von hauptsächlich auf Servern generierten RSA-Schlüsseln konnte das Ergebnis der Schweizer Feldstudie nur teilweise für das Firmenumfeld bestätigt werden: Es wurden 531 Schlüssel mit demselben Modulus gefunden – hauptsächliche Gründe dafür waren wiederverwendete Schlüssel bei Zertifikatserneuerungen (381) sowie eine Verwendung gleicher Schlüsselpaare in der UAT-Umgebung und der Produktion (113). Die restlichen wiederverwendeten Schlüssel wurden firmenintern näher analysiert; dabei fand man jedoch keine Indizien für schwache Zufallszahlengeneratoren, sondern auch hier lag eine Wiederverwendung beispielsweise für weitere Services, die auf dem gleichen Server laufen, vor.

RSA-Algorithmus und -Schlüsselerzeugung

Der RSA-Algorithmus funktioniert wie im Folgenden skizziert – die Schritte 1–3 dienen dabei der Schlüsselerzeugung, die Schritte 4–6 der Ver- und Entschlüsselung:

1. Wähle zufällig zwei verschiedene Primzahlen p und q und berechne deren Produkt $n = p \cdot q$. Der Wert n wird als RSA-Modul bezeichnet.

2. Wähle ein beliebiges $e \in \{2,...,n-1\}$ so, dass gilt:
e ist teilerfremd zu $\varphi(n) = (p-1) \cdot (q-1)$.

Danach benötigt man p und q nicht mehr und kann diese im Prinzip löschen – aus Effizienzgründen werden diese jedoch bei der späteren Nutzung des geheimen Schlüssels genutzt, um mit dem Chinesischen Restsatz effizienter zu dechiffrieren.

3. Wähle $d \in \{2,...,n-1\}$ mit $e \cdot d \equiv 1 \bmod \varphi(n)$
(d. h. d ist die multiplikative Inverse zu $e \bmod \varphi(n)$. Danach kann man $\varphi(n)$ löschen, da es nicht mehr benötigt wird.

(n, e) ist der öffentliche Schlüssel.

(n, d) ist der geheime Schlüssel (es ist nur d geheim zu halten).

4. Zum Verschlüsseln wird die als (binäre) Zahl dargestellte Nachricht in Teile aufgebrochen, sodass jede Teilzahl m kleiner als n ist.

5. Verschlüsselung des Klartexts m (bzw. in den meist angewandten Hybridverfahren ein gepaddeter symmetrischer Schlüssel):
$c = E((n; e);m) := m^e \bmod n$

6. Entschlüsselung eines Geheimtexts c:
$m = D((n; d);c) := c^d \bmod n$

Hinweise zur Schlüsselerzeugung

Bei der Schlüsselerzeugung sind die folgenden Punkte zu beachten:

_____ Bei der Wahl von p und q ist genügend Entropie zu nutzen: Echte Zufallszahlgeneratoren, wie sie in einem Hardware-Security-Modul (HSM) verbaut werden, sind ein gutes Beispiel.

_____ Probabilistische Primzahltests (z. B. Solovay-Strassen, Miller-Rabin) sind mehrfach durchzuführen: Da die 100%-ige Prüfung, ob eine große Zahl prim ist, nicht praktikabel ist, müssen die jeweiligen „Primzahl-Kandidaten" den probabilistischen Primzahltests genügen. Nur wenn die gewählten Kandidaten p und q diese gängigen Primzahltests bestehen, sind sie gute Kandidaten als Primfaktoren für den RSA-Modulus.

_____ Es sollte ein RSA-Modulus von 2048 Bit Länge gewählt werden, wie es das Bundesamt für Sicherheit in der Informationstechnik (BSI) in seinem Algorithmenkatalog [10] empfiehlt, um eine ausreichende Sicherheitsmarge zu haben.

_____ Kleine öffentliche Schlüssel (z. B. $e = 3$) sollten vermieden werden.

_____ Der geheime Schlüssel d sollte größer als $\sqrt{n}$ gewählt werden.

Bei diesen wiederverwendeten Schlüsseln besteht zunächst kein akutes Sicherheitsproblem, jedoch sollte die Wiederverwendung von Schlüsseln in Test- und Produktionsumgebung sowie bei Zertifikatserneuerung tunlichst vermieden werden: Denn organisatorische Kontrollen, wie sie in Produktivumgebungen normalerweise implementiert werden, sind bei UAT-Instanzen nicht zwangsläufig vorhanden, sodass möglicherweise mehr Personen als gewollt Zugriff auf einen produktiv genutzten geheimen Schlüssel haben.

Letztendlich wurde die bald erscheinende Untersuchung von Nadia Henninger et al. [7] bestätigt, dass das Problem schwacher Zufallsgeneratoren in Standard-Serverumgebungen vermutlich nicht so akut ist wie in eingebetteten Systemen.

Auch das CrypTool-Projekt wird die untersuchten Daten (Public-Keys) nicht veröffentlichen, wohl aber demnächst sein Programm allen zur Verfügung stellen, die ihre Schlüssel ebenfalls auf solche Schwachstellen prüfen möchten: Initial werden die Schlüssel/Zertifikate so anonymisiert, dass man statistische Auswertungen machen kann (z. B. dass auf einem bestimmten Server die Zufallsquelle schwach war oder dass eine Applikation eine fehlerhafte Primzahlgenerierung implementiert hat). Damit kann man dann auch Schlüssel prüfen, die von unterschiedlichen Trust-Centern signiert wurden – also Szenarien analysieren, die außerhalb der Möglichkeiten eines Trust-Centers stehen.

Empfohlene Kontrollen für Schlüsselsignierer

Als Signierer eines Schlüsselpaars (z. B. als CA oder Trust-Center) sind die folgenden Punkte zu beachten:

_____ Wenn ein Modulus schon einmal verwendet wurde, sollte ein weiterer Schlüssel mit dem gleichen Modulus nicht akzeptiert werden. Dies gilt insbesondere bei der Erneuerung von digitalen Zertifikaten mit dem gleichen Schlüsselpaar.

_____ Kleine öffentliche Schlüssel sollten nicht akzeptiert werden.

_____ Schlüssellängen kleiner 2048-Bit sollten (wie vom BSI empfohlen **[10]**) nicht akzeptiert werden.

_____ Es sollte ein CA-übergreifender ggT-Test genutzt werden, mit dem man für neue Schlüssel prüft, ob ein Primfaktor schon einmal verwendet wurde.

Zusätzlich wäre zu überlegen, ob nicht das Angebot eines Shared-Service Sinn ergäbe, der Hashwerte von Moduli Trust-Center-übergreifend speichert und mit dem sich leicht überprüfen ließe, ob gewisse Moduli schon benutzt werden. So einen Service Trust-Center-übergreifend anzubieten (z. B. von der European Bridge-CA), wäre möglicherweise ein Ansatz, um das Risiko gleicher verwendeter Moduli einzudämmen.

Fazit

Die Tatsache, dass Tausende von Schlüsseln mit dem längst bekannten Euklidischen Algorithmus (vgl. etwa [8]) geknackt werden konnten, ist bemerkenswert. Das heißt aber nicht, dass der RSA-Algorithmus (siehe Kasten „RSA-Algorithmus") nun als unsicher einzustufen ist, sondern es wurden nur Schwächen beim Aufbau der Public-Key-Infrastruktur (PKI) und besonders bei der RSA-Implementierung und den dabei verwendeten Zufallsgeneratoren zur Schlüsselgenerierung deutlich.

Wie die Untersuchung der Autoren gezeigt hat, sind die professionellen PK-Infrastrukturen innerhalb von Firmen oder von öffentlichen Trust-Centern nicht zwangsläufig auch durch die Ergebnisse der Studie betroffen, sofern sie ordentliche PKI-Prozesse etabliert haben (siehe Kasten „Empfohlene Kontrollen"). Dazu gehört unter anderem, gute Zufallsquellen (z. B. HSMs oder /dev/random) einzusetzen, nach der Schlüssel-Erzeugung auch zu testen, ob derselbe Modul schon einmal vorkam, und über eine ggT-Berechnung alle bisher erzeugten Schlüsselpaare auf gleiche Primfaktoren zu überprüfen (sowohl in der eigenen CA als auch Trust-Center-übergreifend). Aus organisatorischer Sicht muss zudem vermieden werden, dass ein Schlüssel bei einem Zertifikats-Renewal oder bei einem Deployment von einer UAT-Instanz in die Produktion wiederverwendet wird.

Die Schlüssellänge allein garantiert keinen unknackbaren Schlüssel, ebensowenig ein Zertifikat: Die Implementierung, insbesondere die korrekte RSA-Schlüsselgenerierung (siehe Kasten „RSA-Algorithmus"), das Schlüsselmanagement und das Management einer PKI bedürfen entsprechend ausgebildeter Fachleute. Unwissen und Kosten-Sparen (statt Risikomanagement) können ansonsten das ganze Sicherheitsfundament „zerbröseln".

PK-Infrastrukturen kann man sicher betreiben, wenn man in Software Funktionen aus etablierten Bibliotheken wie OpenSSL nutzt, konsequent bei kritischen Funktionen Schlüssel in Hardwaresicherheitsmodulen (HSMs) generiert und speichert, Experten im Schlüssel-Management einsetzt und Betreiber von Applikationen darin schult, wie mit sensitivem Schlüsselmaterial umzugehen ist. Dan Kaminsky hat das in seinem Blog auf den Punkt gebracht: „On the basic level, risk in cryptography is

utterly dominated, not by cipher selection, but by key management." [9]

Aus mathematischer Sicht ist es im Übrigen immer wieder faszinierend, dass man zwar mit hoher Sicherheit die Aussage treffen kann, ob eine Zahl mit Tausenden von Dezimalstellen eine Primzahl ist, aber dennoch eine 600-stellige Semi-Primzahl (aus korrekt generierten großen Primzahlen) nicht faktorisieren (also ihre Primteiler bestimmen) kann. Liegen aber zwei semi-prime Moduli vor, die nur einen solch großen Primteiler gemeinsam haben, dann lässt dieser sich bestimmen, obwohl man jeden einzelnen Modulus nicht faktorisieren könnte. Es ist erstaunlich, wie unterschiedlich gut die mathematischen Verfahren für große Zahlen sind, wenn sich die Fragestellung scheinbar nur geringfügig verändert. ■

Prof. Bernhard Esslinger lehrt IT-Security und Kryptografie an der Uni Siegen und leitet das Open-Source-Projekt CrypTool (www.cryptool.org). Volker Simon und Jörg Schneider sind Sicherheitsexperten und Mitglieder des CrypTool-Projekts.

Literatur

[1] Arjen K. Lenstra, James P. Hughes, Maxime Augier, Joppe W. Bos, Thorsten Kleinjung und Christophe Wachter, Ron was wrong, Whit is right, A Sanity Check of Public Keys Collected on the Web, http://eprint.iacr.org/2012/064.pdf

[2] Daniel J. Bernstein, Factoring into coprimes in essentially linear time, Journal of Algorithms 54 (2005), online via http://cr.yp.to/lineartime/dcba-20040404.pdf

[3] Philippe Oechslin, Making a Faster Cryptanalytic Time-Memory Trade-Off, http://lasecwww.epfl.ch/pub/lasec/doc/Oech03.pdf

[4] Compaq, Cryptography Using Compaq MultiPrime Technology in a Parallel Processing Environment, Technical Brief, ftp://ftp.compaq.com/pub/solutions/CompaqMultiPrimeWP.pdf

[5] MozillaWiki, CA:Communications, Info über Nachrichtenaustausch, https://wiki.mozilla.org/CA%3ACommunications#February_17.2C_2012

[6] Christian Kirsch, Mozilla geht gegen Aussteller von Man-in-the-Middle-Zertifikaten vor, www.heise.de/security/meldung/Mozilla-geht-gegen-Aussteller-von-Man-in-the-Middle-Zertifikaten-vor-1437241.html

[7] Nadia Heninger, New research: There's no need to panic over factorable keys – just mind your Ps and Qs, https://freedom-to-tinker.com/blog/nadiah/new-research-theres-no-need-panic-over-factorable-keys-just-mind-your-ps-and-qs

[8] Wikipedia, Euklidischer Algorithmus, http://de.wiki-pedia.org/wiki/Euklidischer_Algorithmus

[9] Dan Kaminsky, Survey is good. Thesis is strange., Blogeintrag, http://dankaminsky.com/2012/02/14/ron-whit/

[10] Bundesnetzagentur, Bekanntmachungen zur elektronischen Signatur nach dem Signaturgesetz und der Signaturverordnung – Übersicht über geeignete Algorithmen, www.bundesnetzagentur.de/cln_1912/DE/Sachgebiete/QES/Veroeffentlichungen/Algorithmen/algorithmen_node.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