Dieser Artikel erschien ursprünglich in c't 15/08 von **Karsten Nohl**

Quelle: http://www.heise.de/security/Von-Woerterbuechern-und-Regenboegen--/artikel/113681/0 bis 4

Kunterbuntes Schlüsselraten

Von Wörterbüchern und Regenbögen

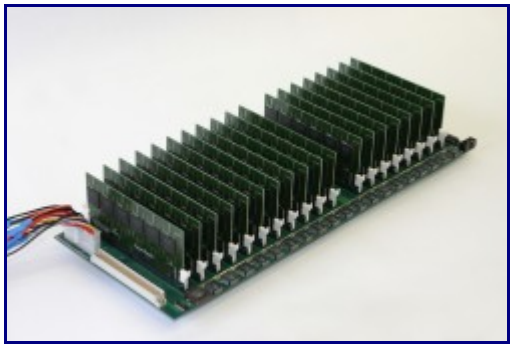
Moderne Kryptoangriffe knacken Handygespräche und Bezahlkartensysteme in Sekunden. Es gilt, mit vorberechneten Tabellen einen praktikablen Kompromiss zwischen Rechenzeit und Speicherplatz zu finden. Kaum ein Algorithmus ist dagegen gefeit, doch spezielle Techniken können die Angriffe vereiteln.

Die Sicherheit von Verschlüsselung beruht darauf, dass niemand außer den legitimen Kommunikationspartnern den geheimen Schlüssel kennt. Doch kryptografische Algorithmen kommen durch Fortschritte in der Kryptanalyse in Bedrängnis und die verwendeten Angriffe erlauben es, diesen Schlüssel zu errechnen. Selbst Algorithmen, für die keine speziellen Angriffe existieren, sind unter Umständen mit guter Aussicht auf Erfolg angreifbar.

Der einfachste erfolgversprechende Angriff, der sich auf jeden Verschlüsselungsalgorithmus anwenden lässt, ist das stumpfe Durchprobieren aller möglichen Schlüssel – einer muss ja schließlich passen und Klartext liefern. Doch um diese sogenannten Brute-Force-Angriffe möglichst zeitintensiv zu machen, setzen die meisten Kryptosysteme sehr lange Schlüssel ein.

Alle vier Milliarden 32 Bit langen Schlüssel auf einem PC durchzuprobieren dauert lediglich Minuten. Arbeitet das System jedoch mit 48-Bit-Schlüsseln, bräuchte ein Angreifer schon etwa

einen Monat; bei 64 Bit sind es bereits tausende Jahre. Selbst die teure Spezialhardware COPACOBANA der Ruhr-Universität Bochum benötigt zum Knacken eines 64-Bit-Schlüssels rund ein Jahr, was Brute-Force-Angriffe gegen Schlüssel mit 64 Bit oder mehr so gut wie aussichtslos macht.



Die Spezialhardware COPACOBANA der Ruhr-Uni Bochum macht mit ihren 120 FPGAs kurzen Prozess mit kurzen Kryptoschlüsseln unter 64 Bit Länge.

Für asymmetrische Verschlüsselungsverfahren wie RSA berechnet sich das Sicherheitsniveau etwas anders: Ein RSA-Schlüssel mit 1024 Bit ist etwa so sicher wie ein 80-Bit-Schlüssel eines symmetrischen Verfahrens.

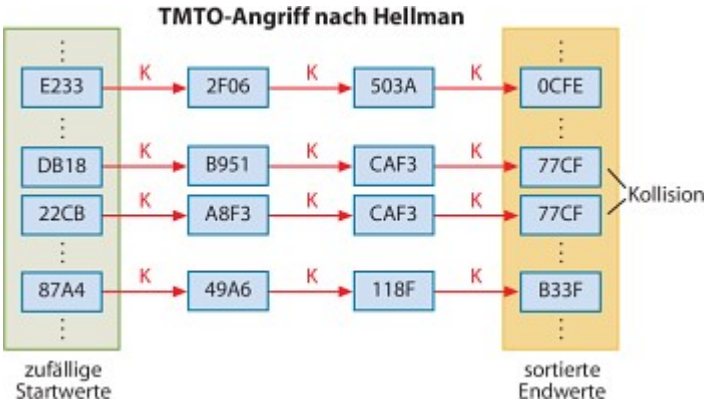
Je nachdem wie ein Verschlüsselungsverfahren eingesetzt wird, lässt sich der geheime Schlüssel mit weiteren fundamentalen Angriffsmethoden weit schneller als mit Brute-Force finden. Diese Angriffe benötigen weniger Rechenzeit, bedürfen aber einer rechen- oder speicherplatzintensiven Vorbereitung. Eine solche Angriffsvorbereitung ist das Anlegen eines Wörterbuchs zu einer bestimmten Nachricht, welches jedem Wert der verschlüsselten Nachricht den verwendeten Schlüssel zuordnet. Auf der Festplatte abgelegt wird das Wörterbuch als Liste aus Schlüsseln, deren Index den verschlüsselten Wert repräsentiert. Um mit ihr den geheimen Schlüssel zu finden, muss ein Angreifer das avisierte System nur noch dazu bringen, die zur Liste passende Nachricht zu verschlüsseln. Dann kann er das Ergebnis als Listenindex verwenden.

Das einmalige Generieren des Wörterbuchs benötigt allerdings etwa so viel Rechenzeit wie ein Brute-Force-Angriff und ist daher nur sinnvoll, wenn sich der Aufwand durch viele damit durchgeführte Angriffe amortisieren kann. Zudem verschlingt ein solches Wörterbuch enorme Mengen Speicherplatz, zum Beispiel 1536 Terabyte für 48-Bit-Schlüssel. Für 64-Bit-Schlüssel wäre bereits mehr Speicher nötig, als der Menschheit heute zur Verfügung steht. Wegen des hohen Platzverbrauchs sind Wörterbuchangriffe deshalb bei langen Schlüsseln sogar weniger praktikabel als Brute-Force.

Weites Feld

Der vom Diffie-Hellman-Verfahren bekannte Kryptograph Martin Hellman war es, der erstmals 1980 einen praktikablen Kompromiss zwischen den beiden Extremen vorschlug, der sich Zeiteffizienz beim Angriff mit Platzverbrauch durch vorberechnete Daten erkauft. Als Begriff für solche Techniken hat sich "Time Memory Trade-Off" oder kurz TMTO eingebürgert. Ihnen liegt die Idee zu Grunde, das Wörterbuch nur teilweise oder komprimiert zu speichern und während der Schlüsselsuche die fehlenden Teile zu berechnen oder die Suche so oft leicht modifiziert zu wiederholen, bis ein Treffer im Wörterbuch vorkommt. Seitdem hat die TMTO-Technik viele Anwendungen gefunden, etwa zum Online-Knacken von Windows-Passwörtern und zum Entschlüsseln von Handygesprächen [1].

Hellman hatte die Idee, das Wörterbuch in Form von langen Ketten zu speichern. Am Beginn einer solchen Kette steht ein zufällig gewählter Schlüssel, mit dem eine festgelegte Nachricht verschlüsselt wird. Die Ausgabe dieser ersten Runde dient in der Folgerunde als Schlüssel zur Verschlüsselung der Ursprungsnachricht. Die Ausgabe dieser Runde liefert den nächsten Rundenschlüssel und so weiter, bis die Kette eine vorgegebene Länge erreicht hat. Um Platz zu sparen, speicherte Hellman von diesen Ketten jeweils nur den ersten und letzten Wert. Das derart komprimierte Wörterbuch ist eine nach den Endwerten sortierte zweispaltige Tabelle.

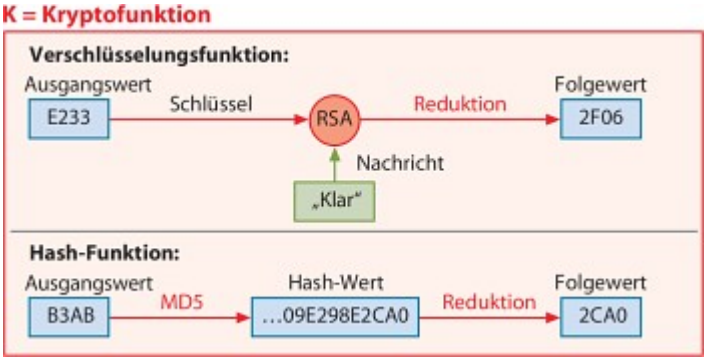


Um Angriffe zu beschleunigen, ohne ein riesiges Wörterbuch speichern zu müssen, speichert man nur Start- und Endwert von Wörterketten. Nur die fehlenden Zwischenschritte der passenden Kette sind beim Angriff noch einmal zu berechnen.

Um einen Schlüssel zu knacken, muss der Angreifer das Zielsystem zunächst dazu bringen, die zu den Ketten passende Nachricht zu verschlüsseln. Die anschließende Suche in dem Wörterbuch ist ein zweistufiges Verfahren. Im ersten Schritt vollzieht der Angreifer mit der verschlüsselten Nachricht dieselben Kettenschritte wie bei der Wörterbucherstellung – und zwar so lange, bis er seinen Zwischenstand als Kettenendwert im Wörterbuch findet.

Im zweiten Schritt muss er ausgehend vom zugehörigen Startwert die Kette nur noch einmal bis zu der Stelle nachrechnen, an der er die verschlüsselte Nachricht findet. Der vorangehende Wert in der Kette ist der gesuchte Schlüssel – so zumindest die vereinfachte Theorie.

Je nach Verschlüsselungsfunktion können aber Kollisionen auftreten, wenn unterschiedliche Zwischenwerte zu demselben Folgewert führen. Da nach einer Kollision stets dieselbe Reihe von Kettengliedern folgt, kann es zu einem Endwert mehrere Kettenstartwerte geben. Falls kein Startwert erwischt wurde, der unterwegs am gesuchten Schlüssel vorbeiführt, bleibt die Suche erfolglos.



Die Kryptofunktion K, die von einem Kettenglied zum nächsten führt, ist je nach angegriffenem Algorithmus unterschiedlich strukturiert. Ist die Ausgabe des Algorithmus zu lang, muss sie an geeigneter Stelle zurechtgestutzt (reduziert) werden.

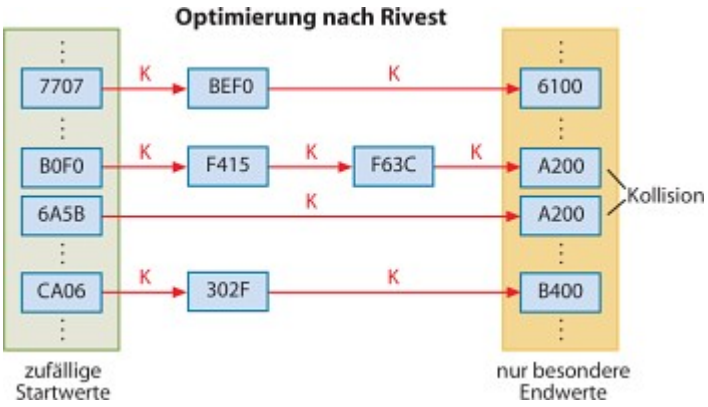
Durch die zufällige Wahl des Startwerts entstehen außerdem unweigerlich Lücken in der Abdeckung – nicht alle möglichen Schlüssel sind in den Ketten enthalten. Statistisch gesehen ist die Menge der berechneten Ketten zu verdoppeln, um die Menge der nicht von einem Kettenglied abgedeckten Schlüssel zu halbieren, was schnell zu großen redundanten Datenmengen führt, wenn man es auf eine gute Abdeckung des Schlüsselraums abgesehen hat.

Die maximale Schrittzahl für das Knacken ist durch die Länge der Ketten bestimmt, und der nötige Speicherplatz ist durch die Anzahl der Ketten gegeben. Je weniger Ketten berechnet wurden, desto länger müssen diese sein und desto länger dauert es, einen Schlüssel zu finden. Mit perfekten Tabellen wäre es möglich, den Schlüssel eines Systems mit N möglichen Schlüsseln in $N^{1/2}$ (Wurzel aus N) Schritten in einem Wörterbuch der Größe $2 \cdot N^{1/2}$ zu finden. Solche perfekten Tabellen existieren aber für die gängigen Verschlüsselungsverfahren und deren große Schlüssellängen nicht. Selbst wenn es sie gäbe, ließen sie sie nicht in realistischer Zeit durchsuchen. Hellman präsentiert in seiner Arbeit einen Trade-off mit $N^{2/3}$ Suchzeit, $2 \cdot N^{2/3}$ Speicherplatz und $N^{2/3}$ Vorberechnungszeit der Tabellen. Für viele Jahre war dies der beste bekannte Trade-Off.

Hellmans Trade-Off ist in vielen Fällen jedoch nicht optimal. Der Flaschenhals bei der Wörterbuchsuche ist heute typischerweise nicht der Prozessor, sondern die Festplatte. Der Algorithmus muss nach jedem Schritt prüfen, ob das erzeugte Element in der Tabelle vorkommt, und dafür in der Regel mehrere Festplattenzugriffe tätigen. Der Prozessor kann daher die Daten meist weit schneller generieren, als sie die Festplatte auftreiben kann.

Um den Flaschenhals "Festplatte" möglichst wenig in Anspruch nehmen zu müssen, schlug der Kryptologe Ronald "Das R aus RSA" Rivest, der auch maßgeblich an Algorithmen wie MD5 und RC4 beteiligt war, 1982 eine Optimierung von Hellmans Methode vor: Es landen nur Endwerte in der Wörterbuchtabelle, die einem bestimmten Kriterium wie "die letzten zwölf Bits sind Nullen" entsprechen. Beim Generieren der Ketten werden diese – wiederum von Zufallswerten ausgehend – nur jeweils soweit durchlaufen, bis ein Wert das Kriterium erfüllt und als Endpunkt dienen kann. Rivests Ansatz ist daher auch als "Distinguished Points" (etwa "besondere Punkte") bekannt. Entspricht bei der Wörterbuchsuche ein Zwischenwert nicht dem Distinguished-Point-Kriterium, braucht er erst gar nicht auf der Festplatte nachgeschlagen werden.

Da der Endpunkt zu einem Startwert nicht vorhersehbar ist, ohne die Kette durchzurechnen, sind die Ketten unterschiedlich lang. Das ändert nichts am durchschnittlichen Bedarf an Suchzeit und Speicherplatz, sehr wohl aber an der Häufigkeit von Kollisionen. Die längeren Ketten haben ein größeres Risiko für Kollisionen, was im Schnitt die geringere Kollisionswahrscheinlichkeit in den kürzeren Ketten nicht wettmachen kann. Zudem ist das Ablaufen einer langen Kette zeitintensiver, was die Nachteile durch die Kollisionen noch einmal verschlimmert.

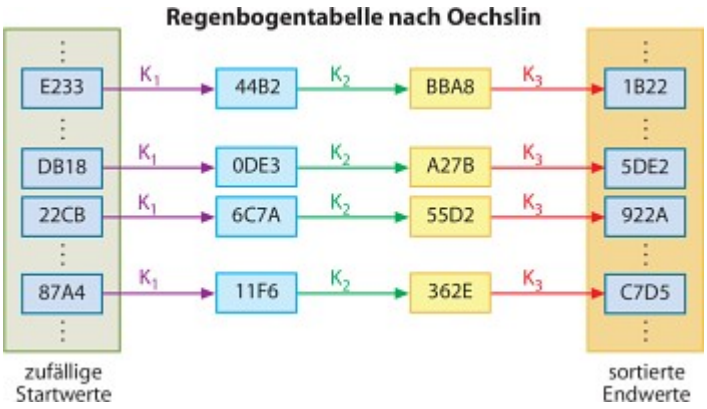


Entsprechen alle Endwerte einem bestimmten Kriterium, braucht man bei der Wörterbuchsuche viele Zwischenwerte erst gar nicht auf der langsamen Festplatte nachzuschlagen. Dies steigert aber das Risiko für Kollisionen.

Alles so bunt hier!

Eine weitere Verbesserung gegenüber Hellmans Technik, die nicht zu vermehrten Kollisionen führt, schlug 2003 Philippe Oechslin vor. Sein Angriff verhindert Kollisionen fast vollständig. Wie bei den anderen TMTO-Techniken wird beginnend mit einem Zufallsschlüssel eine zuvor gewählte Nachricht verschlüsselt. Das Ergebnis wird aber mit einer rundenabhängigen Funktion für die nächste Runde vorbereitet. Bei den anderen Techniken kommt es zu einer Kollision, wenn nach der Rundenverschlüsselung ein bereits zuvor berechneter Wert herauskommt. Die zusätzliche Rundenfunktion sorgt jedoch dafür, dass der abgeleitete Schlüssel für die nächste Runde unterschiedlich ist. Zu Kollisionen kommt es nur dann, wenn die gleiche Ausgabe in zwei Ketten in der gleichen Runde auftritt.

Die Rundenfunktion kann recht einfach sein, etwa: Addiere 1 nach der ersten Runde, addiere 2 nach der zweiten Runde und so weiter. Je nach den kryptografischen Eigenschaften der Verschlüsselungsfunktion führen aber kompliziertere Rundenfunktionen zu besseren Ergebnissen, also zu weniger Kollisionen und besserer Abdeckung aller Schlüssel. Metaphorisch gesehen wandern die Daten durch einen Regenbogen, wo erst die rote Funktion angewendet wird, dann die orange, die gelbe, grüne und so fort. Dies hat der Technik den Namen Regenbogentabellen (Rainbow Tables) eingebracht.



Der Trick, mit den Regenbogentabellen Kollisionen zu vermeiden, ist, die Rundenfunktion in Abhängigkeit der Schrittzahl zu modifizieren.

Zum Finden eines Schlüssels mit Hilfe einer Regenbogentabelle muss ein Angreifer mehrere Abschnitte einer Kette wiederherstellen, denn aufgrund der Rundenfunktion ist es nicht mehr egal, an welcher Stelle die Berechnung beginnt. Als Erstes durchläuft die verschlüsselte Nachricht des anvisierten Systems die letzte Runde, also deren Rundenfunktionen sowie die Verschlüsselung. Liefert die Regenbogentabelle keinen Treffer für den Endwert, durchläuft die Nachricht die vorletzte und die letzte Runde, dann von der drittletzten Runde aus und so weiter, bis ein Endwert in der Tabelle auftaucht. Statistisch wird durchschnittlich die Hälfte der Schlüssel gefunden, sobald die halbe Kettenlänge erreicht ist.

Da die ersten zu berechnenden Unterketten vergleichsweise kurz sind, werden bei der Suche im Durchschnitt nur halb so viele Kettenglieder nachgerechnet wie bei den anderen TMTO-Techniken, was die Schlüsselsuche um bis zu Faktor zwei beschleunigt. Dieser Geschwindigkeitsgewinn tritt allerdings nur dann auf, wenn der gesuchte Schlüssel tatsächlich in der Tabelle vorhanden ist. Andernfalls müssen alle Unterketten generiert werden, um festzustellen, dass das gesuchte Element fehlt, was erheblich aufwendiger ist als bei den anderen Techniken.

Zwischen Oechslins Rainbow Tables und Rivests Distinguished Points gibt es folglich keinen klaren Gewinner und je nach den Eigenschaften der Verschlüsselungsfunktion ist das eine oder das andere Verfahren schneller. Rainbow Tables benötigen wesentlich mehr Festplattenzugriffe, Distinguished Points hingegen verursachen mehr Kollisionen, weil die Menge möglicher Endpunkte kleiner ist.

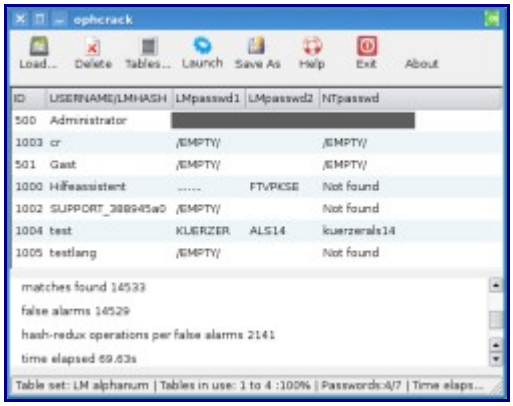
Erfolgsgeschichten

Seit ihrer Einführung sind TMTOs erfolgreich gegen eine Vielzahl von kryptografischen Algorithmen eingesetzt worden. Ein erstes Ziel waren die LM-Hashes von Windows-Passwörtern. Unabhängig von deren Länge speichert Windows ein lokales Nutzerkennwort in Blöcken zu sieben Zeichen, was zu der paradoxen Situation führt, dass ein Passwort mit zehn Zeichen unter Umständen schwächer ist als ein Passwort mit sieben Zeichen. Der Grund ist, dass der Hash über die letzten drei Zeichen sehr leicht zu knacken ist und der errechnete Klartext Aufschluss über die ersten sieben Zeichen Passwortes geben kann.

Die Anzahl der in einem Angriff zu berücksichtigenden Passwörter hängt stark davon ab, welche Arten von Zeichen ein Benutzer wählt beziehungsweise wählen kann. Als Maßeinheit für die Güte eines Passwortes hat sich die Entropie etabliert. Sie gibt an, wie viele mögliche Zustände es annehmen kann und somit, wie viele Passwörter bei einem Angriff durchzuprobieren wären. Sieben Zeichen lange Windows-Passwörter, die nur aus Buchstaben bestehen, entsprechen der Entropie von 33 zufällig gewählten Bits. Kommen Sonderzeichen und Ziffern zum Zeichenvorrat hinzu, sind es immerhin schon 36 Bit. Beliebte Kombinationen wie Namen oder Wörter mit einer Zahl oder einem Ausrufezeichen am Ende haben aber eine weit geringere Entropie.

Typische Windows-Passwörter sind daher ein leichtes Ziel für TMTOs. Tabellen mit 99-prozentiger Trefferrate brauchen etwa zwei Sekunden, um ein 14-stelliges alphanumerisches LM-Passwort zu finden und belegen nur etwa ein GByte Speicher [2]. Zu diesen zwei Sekunden kommt natürlich noch die Zeit, um die Tabellen von der Festplatte in den Arbeitsspeicher zu laden, die sich aber amortisiert, wenn große Mengen an Passwörtern geknackt werden. Aufgrund der hohen Kollisionsrate in der LM-Hash-Funktion benötigt die Vorberechnung der Tabellen etwa die zehnfache Zeit eines Brute-Force-Angriffs, also einige Wochen auf einem PC. Fertige Tabellen gibt es aber auch beispielsweise unter www.shmoo.com zum Download. Die hohe Kollisionsrate macht außerdem Rainbow Tables etwa zehnmal schneller als Distinguished Points.

Genau andersherum verhält es sich bei der Handy-Gesprächs- und SMS-Verschlüsselung A5/1, einer Stromchiffre mit 64 Bit langen Schlüsseln. Sie verwendet einen zufälligen Anfangszustand, um TMTO-Angriffe zu vereiteln. Doch aufgrund einer kryptografischen Schwäche lässt sich dieser Schutz umgehen. In einem verschlüsselten GSM-Paket, etwa einer SMS, finden sich mehr als 200 überlappende Segmente zu je 64 Bit. Um die gesamte Nachricht zu entschlüsseln, muss ein Angreifer nur eines dieser Segmente knacken, da sich die Chiffre von dort sowohl vorwärts als auch rückwärts berechnen lässt, um die fehlenden Bits aufzufüllen. Selbst mit lückenhaften Tabellen ist die Wahrscheinlichkeit vergleichsweise hoch, dass mindestens einer der 200 Werte enthalten ist.



Das Knacken von Passwörtern mit Rainbow Tables geht rasend schnell. Programme wie OphCrack und passende Tabellen gibt es schon fertig zum Download.

Diese Optimierung ermöglicht es, GSM-Schlüssel mit Tabellen zu knacken, die nur etwa eineinhalb Prozent des Platzes belegen, der rechnerisch für eine 64-Bit-Chiffre benötigt würde. Die Optimierung disqualifiziert allerdings Rainbow Tables, da vor einem Fund im Schnitt einige dutzend Male ins Leere gelaufen wird und dieser Fall bei ihnen besonders ungünstig ist.

Aufgrund der niedrigeren Kollisionsrate ist der Aufwand zur Vorberechnung von GSM-Tabellen weit geringer als bei Windows-Passwörtern und braucht gerade mal die doppelte Brute-Force-Zeit. Die 259 Berechnungen dauern aber immerhin noch mehrere Monate auf teurer Spezialhardware mit FPGAs.

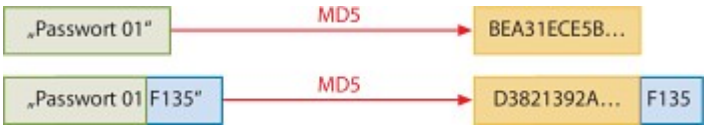
Aber wie bei den Windows-Hashes haben andere diesen Aufwand bereits getrieben. Das "GSM Cracking Project" hat kürzlich Tabellen für A5/1 fertiggestellt, die sie mit Forschern teilen, die drei TByte freien Speicherplatz haben. Mit den Tabellen lässt sich der geheime Schlüssel mit über 95-prozentiger Wahrscheinlichkeit innerhalb von 30 Sekunden finden. Ist ein Schlüssel erst einmal gefunden, lässt sich die komplette Gesprächs- oder SMS-Verbindung mithören beziehungsweise -lesen.

Ebenfalls anfällig für einen TMTO-Angriff ist die Verschlüsselung Crypto1 des Funkbezahlsystems "Mifare Classic" [3], das nicht nur in Fahr- und Mensakarten, sondern auch in Zugangskontrollsystemen weit verbreitet ist. Der in Crypto1 vorgesehene Schutz gegen TMTO verlässt sich darauf, dass die individuelle Chip-ID in die Verschlüsselung einfließt und ein Angreifer somit für jeden Chip eine eigene Tabelle vorberechnen müsste. Doch Schlüssel und ID sind mit einer umkehrbaren Funktion miteinander verknüpft, weshalb sich ein Satz Tabellen für alle Chip-IDs einsetzen lässt. Außerdem benutzt Crypto1 recht kurze 48-Bit-Schlüssel, was den Angriff zusätzlich erleichtert. Ein Satz Distinguished-Points-Tables konnte in gerade einmal 50 Minuten auf dem FPGA-Cluster generiert werden, den auch das GSM-Cracking-Projekt verwendete. Mit Hilfe dieser Tabelle lassen sich die Schlüssel in Sekunden finden.

Schutz gegen TMTO

TMTOs sind oft erfolgreich gegen ausgerechnet jene kryptografischen Systeme, die das Fundament der Computersicherheit bilden. Durch Vorberechnung lassen sich sinnvolle Kompromisse zwischen einem zeitintensiven Brute-Force-Angriff und einem platzintensiven Wörterbuchangriff finden. Ihr Erfolg lässt vermuten, dass diese Art von Angriff schwer zu verhindern ist. Das Gegenteil ist aber der Fall, denn ob sich TMTOs überhaupt sinnvoll einsetzen lassen, hängt nicht von der Krypto-Chiffre selbst, sondern vom Protokoll ab, in dem sie eingesetzt wird.

Der Trick, um die Methode ins Leere laufen zu lassen, ist, die kryptografische Funktion für jeden Einsatz zu variieren. Verschlüsseln zwei Benutzer die gleiche Nachricht mit demselben Schlüssel, sollten zwei unterschiedliche Ergebnisse herauskommen. Die Variation lässt sich beispielsweise dadurch erreichen, dass neben dem eingegebenen Schlüssel auch eine Benutzerkennung oder eine Geräte-ID in die Verschlüsselung einfließt. Ein Angreifer müsste folglich für jeden Benutzer oder gar für jedes Gerät eine eigene Regenbogentabelle berechnen, was in der Regel aufwendiger ist als ein Brute-Force-Angriff.



Ein zufälliger mitgespeicherter Anhang, das „Salz“, vereitelt TMTO-Angriffe. Ein Angreifer müsste nicht mehr nur jedes mögliche Passwort, sondern jedes mögliche Passwort zu jedem möglichen Salzwert berücksichtigen.

Die Crypto1-Chiffre wäre sicher gewesen, wenn sie ID und Schlüssel über eine etablierte Hash-Funktion wie SHA1 miteinander kombiniert hätte. Ein weiterer Fallstrick ist, dass die Eingaben in die Einwegfunktion für jedes Paket unterschiedlich sein müssen, was normalerweise über das Einbinden von Initalisierungsvektoren (etwa in WLAN-Verschlüsselung) oder Zählern realisiert wird.

Wenn Geräte-ID und Benutzername nicht zur Verfügung stehen oder sie häufig identisch sind (etwa "Administrator" oder "root"), kann man auch Zufallswerte erzeugen und mit der verschlüsselten Nachricht abspeichern. Fast alle Unix-Derivate inklusive Linux setzen seit drei Jahrzehnten diese Salt (Salz) genannte Technik für Passwort-Hashes ein. TMTO-Angriffe sind gegen sie aussichtslos. (cr)

Literatur

[1] Christiane Rütten, Lauschgelegenheit, Handy-Gespräche bald abhörbar, c't 24/07, S. 90
[2] Infos zum Crack-Programm OphCrack und passende Tabellen
[3] Karsten Nohl, Jan Krissler, Henryk Plötz, Chiptease, Verschlüsselung eines führenden Bezahlkartensystems geknackt, c't 8/08, S.80 (kostenpflichtiger Download)
[4] Website des Project RainbowCrack mit weiteren Infos