Polymorphe Verschlüsselung

Autor: C. B. Roellgen

20.04.2002

Die Geschichte der Kryptographie hat gezeigt, dass "sichere" kryptographische Methoden in vielen Fällen kurz nachdem sie nennenswerte Verbreitung gefunden hatten, geknackt werden konnten. Ein berühmtes Beispiel ist die im zweiten Weltkrieg von den Nazis verwendete "Enigma" Chiffriermaschine: Britische Spezialisten der Bletchley Park-Gruppe waren in der Lage, den cleveren Permutationscode zu brechen. Als Resultat waren den Briten viele Details deutscher Militäroperationen bekannt. Der Totalverlust der deutschen U-Boote war eine der Folgen.

Fig. 1

Der weitverbreitete DES Algorithmus galt lange Zeit als "sicher". Ein im Januar 1999 von RSA Data Security, Inc. (San Mateo, Calif., USA) durchgeführter Versuch bewies, dass es weniger als 22.25 Stunden dauert, um den 56 Bit Algorithmus mit der Brute-Force-Attacke (Ausprobieren aller 2^{56} Schlüsselkombinationen) zu bezwingen. Ein Jahr zuvor benötigte ein ähnlicher Test noch 41 Tage! RSA behauptet einen viel besseren Algorithmus zu haben, was mit Sicherheit zutrifft.
Heute wird eine Schlüssellänge von 128 Bit (entspricht etwa 1000 bit für RSA) von Experten als sicher angesehen. Es sind allerdings nur 25 Jahre vergangen, seitdem Experten die halbe Schlüssellänge von 64 Bit als absolut sicher bezeichneten. Die Steigerung der Schlüssellänge entspricht dabei lediglich dem fast unbedeutenden Faktor 10000000000000000000 an zusätzlicher Sicherheit!

Zurecht wird jedoch eine Schlüssellänge von 256 Bit heute als absolut sicher angesehen. Es ist keine Maschine denkbar, die einen derart langen Code in akzeptabler Zeit brechen kann.

Durch die Bemühungen der National Security Agency NSA, sichere kryptographische Verfahren in den USA und nach Möglichkeit auch im Rest der Welt zu verbieten, ist es durchaus wahrscheinlich, dass alle von der NSA unterstützte Verschlüsselungsverfahren eine Art Abkürzung beinhalten. Folglich kann man nicht besonders sicher sein, dass starke Verschlüsselungsverfahren wirklich so stark sind wie von einer (begrenzten) Anzahl Experten behauptet wird. Die 40 Bit-Kryptographie im Secure Socket Layer-Protokoll beweist mit ihrer ultraleichten "Knackbarkeit" (innerhalb weniger Sekunden auf einem mittleren Großrechner), dass bereits der Name nicht das hält was er verspricht.

Die Technologie macht jeden Tag Fortschritte, sodass es immer wahrscheinlicher wird, dass irgendjemand in der Lage ist, Dateien mit geheimen Daten zu entschlüsseln. Warum nicht einfach den Verschlüsselungsalgorithmus überdimensionieren, um dieses lästige und unausweichliche Sicherheitsproblem zu beseitigen?
Der Grund für die Unterdimensionierung heutiger Algorithmen ist neben der Politik vor allem die Geschwindigkeit! Es wird generell behauptet, dass lange Schlüssel den Algorithmus übermässig verlangsamen. **Das trifft in der Tat für herkömmliche Algorithmen zu, weil die Ausführungszeit mindestens quadratisch mit der Schlüssellänge ansteigt!**

Ein neuer Ansatz, der selbstcompilierenden Maschinencode impliziert, löst dieses Problem. **Bei diesem Verfahren steigt die Ausführungszeit lediglich linear mit der Schlüssellänge.** Die Idee dahinter ist, den Verschlüsselungsalgorithmus zufällig zu gestalten. Aus diesem Grund habe ich ihn „Polymorphe Methode“

getauft.

Ich stellte mir die Frage, was die Folgen wären, wenn sowohl der Datenschlüssel, als auch der Verschlüsselungsalgorithmus anfänglich undefiniert wären? Ein Gegner, welcher einen Schlüssel knacken wollte, wäre jeglicher Konstante beraubt! Die Arbeit mit lauter Variablen und keinerlei Konstanten wird schnell sehr komplex. Herkömmliche Verschlüsselungsverfahren verwenden nur einen Schlüssel - somit nur eine Variable. Eine mathematische Gleichung mit mehreren Variablen hingegen kann nicht gelöst werden! In der Kryptographie gibt es natürlich immer eine Lösung - das Probieren jeder möglichen Schlüsselkombination. Dieses Problem ist bei herkömmlichen Verfahren eindimensional und bei der Polymorphen Methode zweidimensional (dieser Abschnitt ist eher veranschaulichend gedacht).

Die Polymorphe Methode ist heute eines der stärksten Verschlüsselungsverfahren und sie ist wahrscheinlich sogar das stärkste. Die Methode nutzt zufällig zusammengestellten Maschinencode, um eine aussergewöhliche Sicherheit gegen jede Art von Attacke bereitzustellen. Sie ist sogar intrinsisch sicher gegen das Disassemblieren und Analysieren der Instruktionssequenz - die Abfolge der Befehle ist selbst eine Variable und ist nur für die kurze Zeit der Ausführung klar definiert!

Es ist entscheidend, dass bereits die grundlegende Voraussetzung für die erfolgreiche Kryptoanalyse, die Kenntnis des Verschlüsselungsverfahrens, nur rudimentär erfüllbar ist. Der Kryptocompiler ist analysierbar, nicht jedoch das tatsächliche Verfahren, mit dem Nachrichten verschlüsselt werden. Der tatsächliche Verschlüsselungsalgorithmus der Polymorphen Methode is inhärent variabel.

Arbeitsweise der polymorphen Verschlüsselungsmethode

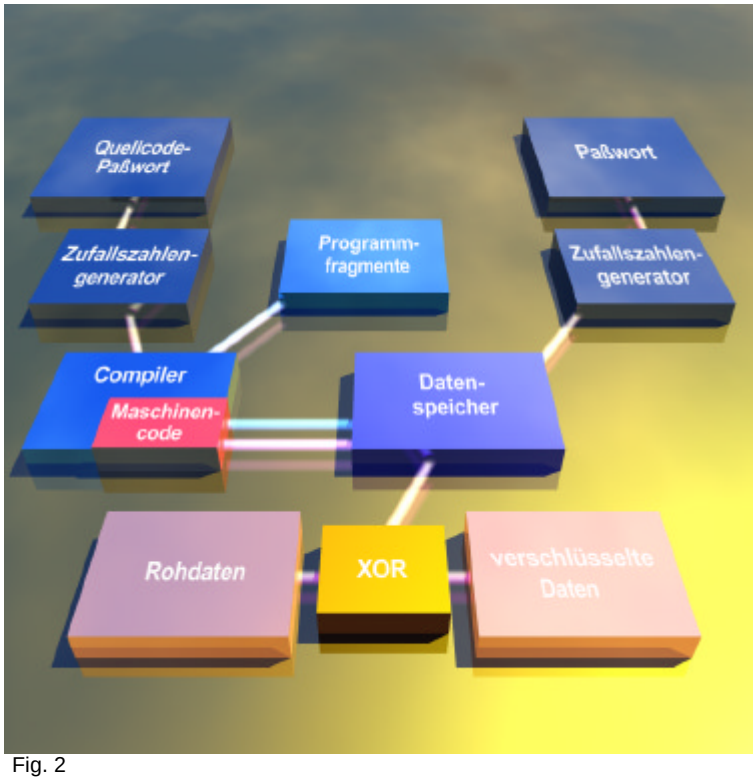


Fig. 2

Zwei unterschiedliche Passwörter (oder zwei Teile eines einzigen Passworts) werden als Saat zwei Zufallszahlengeneratoren zugeführt. Der Zufallszahlengenerator auf der linken Seite erzeugt einen Datenstrom, der zu einer Folge von kryptographischen Maschinencodeblöcken compiliert wird. Der für diesen Zweck bereitstehende Compiler assembliert im wesentlichen standardisierte Befehlsblöcke, richtet Zieladressen ein und erzeugt Eingangs- und Ausgangscode für den Maschinensprachteil, der bei Programmausführung den Inhalt des Datenspeichers während der Laufzeit intensiv manipuliert. Der Datenspeicher wird mittels des Zufallszahlengenerators auf der rechten Seite initialisiert. Das Passwort auf der rechten Seite dient dabei als Saat für den Zufallszahlengenerator auf der rechten Seite.
Nach Beendigung der Ausführung des Maschinencodes wird der Inhalt des Datenspeichers zur Verschlüsselung der Rohdaten unter Zuhilfenahme der XOR-Funktion verwendet. Der Inhalt des Datenspeichers sollte jedoch vorzugsweise als Saat für einen weiteren schnellen kryptographischen Algorithmus dienen. Dadurch erhöht sich die Komplexität des Verschlüsselungsverfahrens und es wird Angreifern die Analyse des internen Zustands erschwert. Es sei angemerkt, dass der Inhalt des Datenspeichers zu keiner Zeit Rückschlüsse auf einen der beiden oder beide Schlüssel zulässt.

Angreifern wird ihr Vorhaben zusätzlich erschwert, wenn der Maschinencode von Zeit zu Zeit neu compiliert wird. Ein derartiges Verschlüsselungsverfahren ist dynamisch polymorph.

Eine Implementierung der polymorphen Verschlüsselungsmethode ist als Windows-Programm unter dem

Namen "Best Possible Privacy" frei verfügbar. Die Kryptomaschine, basierend auf 32 Bit 80386-Code, verwendet das CPU-Register ebx als Eingangs-und Ausgangsregister, eax als allgemeinen Puffer und ebp als Adresszeiger für Zugriffe auf den Datenspeicher.

Beispiel für einen einfachen Codeblock

Beispielhaft soll ein kurzer und kryptographisch wenig anspruchsvoller Maschinencodeblock Ziel der Betrachtung werden. Bestandteile eines derartigen Codeblocks sind Zeigervorbereitung, kryptographische Operationen, Speichermanipulation und abschliessend Aufräumarbeiten.
Im folgenden Beispiel ändert eine xor-Operation ebx und vier Bytes im Datenspeicher:

```
  push ebp;           // save the start address of the key data array for later
  mov eax,123;        // load offset: constant data which was calculated by the compiler
  add ebp,eax;
  mov eax,[ebp+0];    // load key[ebp+0] in AL and key[ebp+1] in the next upper byte of eax and
                      // so on up to key[ebp+3]
  xor ebx,eax;        // this instruction can be replaced by another or a set of instructions
  xor [ebp+x],ebx;    // change the key data array frequently; x is defined by the compiler and
                      // chooses one element of the key
  pop ebp;            // restore start address of the key data array
```

Anstelle der xor-Verknüpfung ist es natürlich möglich, Summen zu berechnen, Schiebebefehle, Multiplikationen und Modulodivisionen durchzuführen, sowie Pseudozufallszahlen zu erzeugen und weitere komplexe Befehlssequenzen auszuführen. Eine gute Implementierung der vorgestellten Methode sollte sich auf einen Satz Codeblöcke verlassen, die grundsätzlich viele Bytes im Datenspeicher verändern. Einfache xor-Instruktionen, sowie Additionen und Subtraktionen sind linear und aus diesem Grund kryptographisch schwach - jedoch kann anhand eines einfachen Beispiels das Konzept der Blockverkettung leicht demonstriert werden.

Die Befehlsblöcke sollten den Schlüssel, der im Datenspeicher gepuffert wird, oft verändern. Dadurch wird Angreifern, welche eine Codetabelle führen, die Möglichkeit der Kryptoanalyse versperrt.
Wenn die Methode als Pseudozufallszahlengenerator verwendet wird, bietet sich der Inhalt des ebx-Registers zur weiteren Verarbeitung an. Der interne Zustand des Datenspeichers ist groß genug, um nicht direkt oder indirekt freigelegt zu werden.

Ein Beispiel für einen, mit obigem Beispiel verglichen kryptographisch sicheren Codeblock, ist folgende CRC32-Implementierung:

Die Funktion berechnet einen 32 bit CRC in Anlehnung an IEEE 802. Das verwendete Polynom lautet: x32 + x26 + x23 + x22 + x16 + x12 + x11 + x10 + x8 + x7 + x5 + x4 + x2 +x + 1. X32 existiert natürlich nicht physikalisch und die "1" invertiert lediglich das Bit am Eingang des Schieberegisters. Als 32 Bit Zahl kurz und knapp geschrieben, hat das Polynom den Wert $04C11DB7.

```
  push ebp;           // ebp MUST never be really destroyed
  and  eax,127;       // perform an operation with four key bytes at a time using eax from the
                         previous instruction block
  add  ebp,eax;
  mov  eax,[ebp+0];   // load key[ebp+0] in AL and key[ebp+1] in the next upper byte of eax
                         and so on up to key[ebp+3]

  mov  esi,ebp;       // save ebp for later to alter the key
  pop  ebp;           // get original base of the key data array
  push ebp;           // restore stack frame
  push ebx;           // save ebx for later

  mov  ecx,32;        // counter for the loop
  xor  edx,edx;       // edx is used to clear the zero flag before the loop @rep1 command below
@rep1:rcl ebx,1;      // shift data in from ebx
  rcl  eax,1;         // use eax as CRC buffer
  jnc  @cnt1;         // CRC decision
  xor  eax,$04C11DB7; // xor with IEEE 802 generator polynomial
@cnt1:add dl,1;       // clear Zero-Flag (will be rarely necessary)
  loop @rep1;
  pop  ebx;           // restore old ebx value. ebx keeps a running 32 bit result
  mov  ebp,esi;       // get the address of the previously selected key data bytes
```

```
mov  [ebp+0],eax;   // alter the key
xor  ebx,eax;       // alter ebx
// here is the end of the CRC routine

pop   ebp;         // exit the routine by restoring the original ebp
```

Der vorgestellte Codeblock beeinflußt nur vier Datenbytes im Datenspeicher. Abhändig von der Größe des Datenspeichers sollten weit mehr Speicherplätze verändert werden, damit eine optimale Angriffsicherheit gewährleistet werden kann. Es ist leicht möglich, die Routine zu erweitern, um dieser Forderung nachzukommen.

Es ist ferner möglich, Schleifen über einen oder über mehrere Codeblöcke zu legen. Dies wird in der Regel mittels loopne-Befehl beim 80386 und dessen Nachfolgern realisiert. Die Verschlüsselungsmethode benötigt dann mehr Zeit um Instruktionen abzuarbeiten, wodurch die Kryptoanalyse ein zeitraubender Zeitvertreib wird. Wenn ebp durch quasizufällige Initialisierung von eax bei jedem Schleifenzyklus geändert wird, kann der im Datenspeicher gepufferte Schlüssel den Algorithmus überdies öfter beeinflussen.

Einige Gedanken zur Angriffssicherheit

Jeder Codeblock beeinflußt mindestens 32 Bit und machmal ändert es sogar den Schlüssel im Datenspeicher.
Angenommen der Kryptocompiler verfüge lediglich über vier Codeblöcke und 16 dieser Blöcke könnten quasizufällig aneinandergereiht werden, so wären 4^{16} = 4294967296 verschiedene Möglichkeiten für den resultierenden Kryptoalgorithmus gleichermassen wahrscheinlich. Bei 128 aneinandergereihten Codeblöcken stünden 4^{128} = $1{,}158*10^{77}$ Möglichkeiten zur Verfügung (Anmerkung: Ein konventioneller 128 Bit Verschlüsselungsalgorithmus verfügt nur über $3{,}403*10^{38}$ mögliche Schlüsselvarianten!).
Es ist sehr wichtig darauf hinzuweisen, daß OHNE die Ausführungszeit des Verschlüsselungsalgorithmus zu verlängern, die Angriffsicherheit eines polymorphen Verschlüsselungsverfahrens um Größenordnungen über der herkömmlicher Verfahren liegt! Als Basis für diese These dient die Tatsache, daß anders als bei konventionellen Methoden bereits vor der Ausführung eines polymorphen Verschlüsselungsalgorithmus eine Auswahl aus einer riesigen Anzahl möglicher Kombinationen getroffen wird.

Um die Angriffssicherheit der polymorphen Methode zu berechnen, muß die Anzahl möglicher Variationen des Algorithmus mit der Anzahl Datenschlüsselkombinationen multipliziert werden.

Die Schlüssellänge im Datenspeicher betrage beispielsweise 16 Bytes = 128 Bits - folglich existieren 2^{128} = $3{,}403*10^{38}$ Kombinationen für den Schlüssel, der im Datenspeicher verarbeitet wird. Zusammen mit der Anzahl möglicher Algorithmen (16 Blöcke, 4 unterschiedliche Codeblöcke) ergibt sich das Produkt $1{,}158*10^{77}$ * $3{,}403*10^{38}$ = $3{,}913*10^{115}$ gesamter Schlüsselkombinationen bei der polymorphen Methode.

Um konventionelle kryptographische Methoden mit der polymorphen Methode vergleichen zu können, muss der gesamte Schlüsselbereich verglichen werden. Aufgrund der Annahme, dass alle verglichenen Methoden mit 128 Bits Datenschlüssel arbeiten, ist diese Art des Vergleichs zulässig. Das polymorphe Verschlüsselungsverfahren schlägt jeden konventionellen Verschlüsselungsalgorithmus um den Faktor $3{,}913*10^{115}$ / $3{,}403*10^{38}$ = $1{,}150*10^{77}$ (!). Dieser Faktor ist so groß, dass er die Anzahl Atome auf unserem Planeten übersteigt!

Die Implementierung des vorgestellten Verfahrens in Computerprogramm "Best Possible Privacy (BPP)" verwendet 32 Codeblöcke und eine drei-Bit Konstante pro Codeblock. Folglich gibt es 32 Wege den Algorithmus zu bestimmen multipliziert mit 8 Möglichkeiten der Initialisierung durch eine Konstante je Codeblock => 256 = 2^8 Varationen je Codeblock. Der Algorithmus ist auf 1024 Codeblöcke limitiert, wodurch $2^{(1024 * 8)}$ = 2^{8192} unterschiedliche Algorithmen möglich und gleich wahrscheinlich sind. Der 256 Byte lange Datenschlüssel im Datenspeicher erhöht die Sicherheit weiter auf 2^{8192} * 2^{2048} = 2^{10240} Schlüsselkombinationen. Es ist dabei interessant zu sehen, daß nahezu 100% der Sicherheit durch den Compiler herrührt. Das neue Verfahren, Schlüsselströme zu generieren, verwendet bekannte Technologieen, die dabei stark verbessert werden.

8192 Bit lange Schlüssel sind natürlich vollkommen überzogen! Die damit erzielbare Sicherheit ist nicht wesentlich größer als mit 256 Bit. Unknackbar ist eben unknackbar. Dennoch wurde das ursprünglich als Demonstrationssoftware konzipierte Programm BPP mit einer derartigen Crypto-Engine realisiert. Warum nicht?

Angriffe und ihre Wahrscheinlichkeit auf Erfolg beim polymorphen Verschlüsselungsverfahren

Angriffe sind keine Algorithmen, sondern allgemeine Ansätze, die bei jedem neuen Verschlüsselungsverfahren neu erfunden werden müssen.
Es wird generell vorausgesetzt, dass der Gegner die Einzelheiten des Verschlüsselungsverfahrens genau kennt und dass ihm nahezu unbegrenzt Klartext und dessen verschlüsselte Form bekannt sind ("known plaintext"). Es wird des weiteren vorausgesetzt, daß der Gegner über Echtzeitfähigkeit verfügt, um definierten Klartext durch dessen Verschlüsselung nach belieben zu sammeln sowie die Ergebnisse zu speichern.

Ausprobieren aller Schlüssel (Brute Force on the keys)

Versuche jede mögliche Schlüsselkombination bis die Nachricht entschlüsselt ist. Probiere die wahrscheinlichen Schlüssel zuerst.

128 Bit Schlüssellänge sollte ausreichend sein, um den Erfolg mit Brute Force auf absehbare Zeit zu verhindern. Das Zufallszahlensystem der polymorphen Methode läßt sich jedoch erst ab einer Länge von 256 Bit sinnvoll realisieren - 2048 Bit sind eher gewöhnlich, ohne den Algorithmenschlüssel zuzuzählen, der zu knapp 100% der Sicherheit beiträgt.

Ausgesuchte Schlüssel

Probiere unterschiedliche Schlüssel anhand bekanntem Klartext und vergleiche die resultierenden verschlüsselten Daten mit den tatsächlichen verschlüsselten Daten - versuche den korrekten Schlüssel zu berechnen.

Aufgrund der Tatsache, daß der Schlüssel mehr oder weniger der Algorithmus selbst ist, gerät die Aufgabe des Gegners zu einem hoffnungslosen Unterfangen. Die polymorphe Einwegfunktion verändert sich permanent mit jedem neuen Schlüssel, der so groß ist, daß es unmöglich ist, eine Tabelle zu isolieren und damit separat zu arbeiten. Ein Computer kann schließlich nur so groß sein wie es Atome auf diesem Planeten gibt!

Codebuch mit verschlüsseltem Text

Sammle so viele verschlüsselte Daten wie möglich, um ihre Inhalte durch Beziehungen untereinander zu extrahieren; wenn ein verschlüsselter Text auftaucht, lies ihn einfach im Codebuch nach. Diese Methode behandelt einen Blockverschlüsseler wie einen Code und ist der klassische Ansatz, um Codes zu knacken.

Genau wie häufiger auftretende Buchstaben, treten Worte und Phrasen mit bestimmter Frequenz auf. Dies gilt auch für Blöcke eines Verschlüsselungsalgorithmus. Wenn die Blocklänge klein ist (unter 64 Bytes) und wenn der Schlüssel nicht häufig gewechselt wird, so könnte es möglich sein, ein Codebuch mit Blockinhalten aufzubauen, das ebenso die gewünschten Bedeutungen enthält (den Klartext).

Jede Art von Codebuchattacken wird idealerweise durch Verwendung einer grossen Anzahl Blockwerte und folglich einer großen Blocklänge verhindert. Wenn die Blocklänge 64 Bytes oder mehr beträgt, so kann man erwarten, dass die Einzigartigkeit, die jeder Block beinhaltet, gross genug ist, um die Fähigkeiten eines noch so starken Gegners, ein Codebuch zu erstellen, übersteigt.

Aufgrund der Tatsache, dass die Komplexität jeder Art Codebuchattacke ausschließlich von der Blocklänge abhängt, ist "dreimal" soviel kein geeignetes Mittel um den Schwierigkeitsgrad zu steigern. Im besonderen

sei Triple DES genannt, das dieser Attacke keinesfalls mehr als DES selbst widersteht. Die Komplexität der Transformation spielt hier keine Rolle.

Die polymorphe Verschlüsselungsmethode wird am besten mit einer 1024 Byte Blocklänge und sich blockweise änderndem Maschinencode implementiert. Die Methode ist ferner ideal geeignet, um eine Saat für einen Zufallszahlengenerator zu erzeugen, um den eigentlichen Algorithmus von der letztendlichen Zufallszahlenfolge zu trennen. Weil die polymorphe Methode bei jedem Schlüssel und bei jedem Block eine andere Form hat, wird jede Art Codebuchattacke vor allem Rauschen aufzeichnen und zu sonst nichts zu verwenden sein.

Bekannter Klartext (Known Plaintext)

Erzeuge auf irgendeine Weise eine große Menge verschlüsselten Text aus Klartext mit immer dem gleichen Schlüssel.

Mit dieser Attacke erhält man mit einem Klartext - verschlüsselter Text - Paar ausreichend Informationen, um den vollen Inhalt des Datenspeichers zu rekonstruieren. Um jedoch einen der beiden oder beide Schlüssel zu identifizieren, müssen beide Schlüssel mit einer anderen Methode extrahiert werden.
Da sowohl die Compilereingangsdaten als auch die Schlüssel unbekannt sind, ist es schierig den kompletten internen Zustand des Kryptosystems zu erhalten. Die polymorphe Methode versteckt ungefähr drei Viertel des internen Zustands in den Compilereingangsdaten und damit in einem veränderlichen Algorithmus, wodurch alleine ausreichende Komplexität entsteht und sich die Verschlüsselungsmethode dieser Art Attacke völlig entzieht. Es sei angemerkt, dass ein einziges Klartext - verschlüsselter Text - Paar einen DES-Schlüssel identifizieren könnte!

Codebuch mit bekanntem Klartext (Known-Plaintext Codebook)

Sammle soviele Klartextblöcke mit entsprechendem verschlüsseltem Text als möglich, damit ein auftretender verschlüsselter Text einfach durch das Auffinden im Codebuch entschlüsselt werden kann.

Einfache Blockchiffrierer verhindern Codebuchattacken dadurch, dass sie den Klartext verwürfeln (oft mittels Verknüpfung mit den Daten des letzten Blocks "Cipher Block Chaining"). Auf diese Weise erreicht man, dass die Blockwerte im Klartext gleichverteilt sind.

Codebuchattacken werden idealerweise durch eine große Anzahl Blockwerte vereitelt, wodurch eine grosse Blockgrösse unvermeidlich ist. Um dies auch in Zukunft zu verhindern, wird eine Blockgrösse von 64 Byte als sicher angesehen, sodass die Grösse des Codebuchs nicht zu verarbeiten ist. Die 1024 Byte Blocklänge der polymorphen Methode machen es unmöglich, Erfolg mit dieser Art Attacke zu haben. Aufgrund der Eigenschaft des Algorithmus sich selbst zu verändern, wird eine Tabelle mit aufgezeichneten Blockwerten nur Rauschen enthalten.

Klartextauswahl (Chosen Plaintext)

Ohne den Schlüssel zu kennen, soll beliebiger Klartext verschlüsselt und die verschlüsselten Daten aufgezeichnet werden. Verändere die Daten andauernd, um gezielt bestimmt verschlüsselte Daten zu erhalten.

Hier wird nicht der Code selbst geknackt, sondern der originäre Klartext produziert. Wenn der Gegner ausgewählten Klartext produzieren kann, so kann er eventuell beliebige verschlüsselte Daten zur Dechiffrierung eingeben.

Die Schwäche, die bei diesem Verfahren ausgenutzt wird, liegt manchmal nicht beim Verschlüsselungssystem selbst, sondern an einem Ort mit wenigen internen Zuständen. Die polymorphe Verschlüsselungsmethode verfügt nicht über einen festen Algorithmus und damit auch nicht über konstante Schwächen, sondern eher über viele diskrete Schwächen - soviele es Kombinationen für den Maschinencode gibt. Die Klartextauswahlmethode ist vor allem wegen des dynamisch veränderlichen Maschinencodes nicht erfolgversprechend.

Codebuch mit ausgewähltem Klartext (Chosen-Plaintext Codebook)

Erzeuge so viel verschlüsselten Text aus Klartext wie möglich. Wenn ein bekannter Code auftritt, suche den passenden Klartext im Codebuch.

Diese Attacke ist mit den vorigen Codebuchattacken vergleichbar, diesmal jedoch mit der Fähigkeit, das Codebuch beliebig und mit hoher Geschwindigkeit zu füllen. Wieder hängt der Erfolg von der Blocklänge und der Starrheit des Verschlüsselungssystems ab. Diese Attacke ist ebenso wenig erfolgversprechend wie alle anderen Codebuchattacken und es ist leichter, viele Schlüsselkombinationen durchzuprobieren.

In der Mitte aufeinandertreffen (Meet-in-the-Middle)

Suche bei mehrlagigem Aufbau mit gegebenen oder bekanntem Klartext unter allen Schlüsseln den passenden in der oberen Lage. Verfahre ebenso für die untere Lage.

Bei einem zweilagigen Konstrukt und kurzer Blocklänge können Übereinstimmungen durch wenige Eingangs-/Ausgangsblockpaare überprüft werden. Sind weitere Lagen vorhanden, können diese immer in zwei Teile partitioniert werden, wodurch diese Methode immer anwendbar ist. Die Komplexität dieses Unterfangens kann natürlich beliebig ausfallen.

Jede Lage in einem guten Kryptosystem verwendet einen großen Codeumfang für den Schlüssel. Die polymorphe Methode fügt überdies einen sehr langen Schlüssel für den für Angreifer unbekannten Algorithmus hinzu. Die Anzahl Kombinationen für beide Schlüssel ist gleich dem Produkt der Kombinationen für jeden einzelnen Schlüssel und damit eine ungeheuer grosse Zahl.

Schlüsselbitbeeinflussung (Key Bit Bias)

Durch extensives Ausprobieren vieler möglicher Schlüsel und extensives Verschlüsseln eines festen Klartexts kann es manchmal möglich sein, bestimmte Schlüsselbits über statistische Häufigkeit einiger Bits in den verschlüsselten Daten in Verbindung zu bringen. Konventionelle Algorithmen mit derartigen Schwächen wären schnell geknackt.

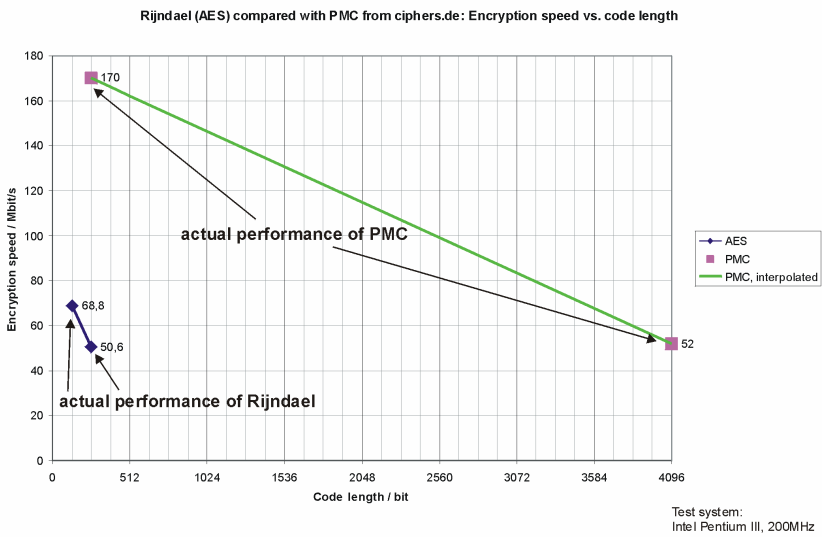
Unterschiedliche Schlüssel erzeugen bei der polymorphen Methode unweigerlich verschiedene Kryptoalgorithmen, wodurch es unmöglich wird, eine statistische Verknüpfung zwischen Eingangs- und Ausgangswerten herzustellen. Jeder Schlüssel hat seine eigene Schwäche, die jedoch nicht konstant über alle möglichen Schlüsselwerte ist.

Differentielle Kryptoanalyse (Differential Cryptanalysis)

Nutze bekannte Eigenschaften bestimmter bekannter Substitutionstabellen aus, um die Anzahl Durchläufe in einem iterierenden Blockverschlüsseler zu reduzieren.

Die ursprüngliche Idee der differentiellen Kryptanalyse bezieht sich hauptsächlich auf iterierende Blockverschlüsseler mit bekannten Tabellen. Nichts davon ist auf die polymorphe Methode anwendbar. In einem iterativen Verschlüsselungsalgorithmus wie DES können statistische Unausgewogenheiten gefunden werden. Dies kann dazu ausgenutzt werden, um auf vorangehende Iterationsschritte zurückzuschliessen.
Bei der polymorphen Methode wird jeder neue Schlüssel einen bestimmten Algorithmus auswählen, wodurch selten ähnliche Transformationen gewählt werden. Es ist substantiell schwierig und sehr ineffizient, eine Transformation anzugreifen, die während der Analyse ihre Struktur vollkommen verändert.

Geschwindigkeit



Die ursprüngliche Implementierung einer Polymorphen Crypto-Engine in BPP ist, verglichen mit neueren Entwürfen, zu langsam.
Eine Variante mit 4096 Bit findet Verwendung in einem Verschlüsselungstool für E-Mails, bei dem mehrfach Teilschlüssel zwischen den Kommunikationspartnern ausgetauscht werden. Diese Teilschlüssel werden zur Verschlüsselung weiterer E-Mails aneinandergereiht, wodurch eine erhebliche Steigerung der Angriffsicherheit erzielt wird. Das hier verwendete Stacked-Diffie-Hellman-Verfahren ist die erste Anwendung, bei der ein symmetrisches Verschlüsselungsverfahren mit mehr als 256 Bit nicht nur sinnvoll, sondern notwendig ist. Die 4096 Bit-Engine erreicht immerhin 52Mbit/s Verschlüsselungsgeschwindigkeit und ist damit ähnlich schnell wie das von der US-Regierung genutzte AES-Verfahren bei 256 Bit.

Je kürzer der Code, desto schneller sind symmetrische Verfahren. Mit einer 256 Bit-Engine sind derart gute Werte erzielbar, daß sogar hochauflösende Videostreams mit günstigen Computern in Echtzeit ver- und entschlüsselt werden können. Die 256 Bit Crypto-Engine wird im Produkt BPP Disk verwendet, einem Festplattenverschlüsselungsprogramm, bei dem ein Softwaretreiber eine oder mehrere Wechselfestplatten nachahmt, die bei Bedarf dem Dateisystem zugeschaltet werden können.

Zusammenfassung

Für das noch im Jahre 1999 in Deutschland als Staatsgeheimnis eingestufte Polymorphe Verschlüsselungsverfahren ist bis heute kein potentiell erfolgreicher Angriff bekannt.
Es gibt keine Möglichkeit, auf bestimmte Schlüssel zurückzuschliessen oder Klartext ohne Kenntnis des passenden Schlüssels zu rekonstruieren, außer durch extensive Durchsuchung aller Schlüsselkombinationen. Das vorgestellte Verfahren verfügt über eine vergleichbare Anzahl Datenschlüssel, jedoch zusätzlich über eine erhebliche Anzahl möglicher Schlüssel für den Algorithmus. Alles in allem bedeutet dies, dass PMC einen extremen Sicherheitsvorsprung gegenüber konventionellen Verfahren und einen spürbaren Geschwindigkeitsvorteil bietet.

Weitere Informationen stehen für Sie bereit unter http://www.ciphers.de/