

Advanced Encryption Standard

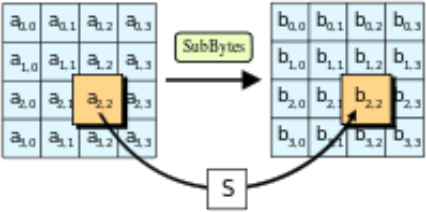
aus Wikipedia, der freien Enzyklopädie

Der **Advanced Encryption Standard (AES)** ist ein symmetrisches Kryptosystem, das als Nachfolger für DES und 3DES im Oktober 2000 vom National Institute of Standards and Technology (NIST) als Standard bekanntgegeben wurde. Nach seinen Entwicklern Joan Daemen und Vincent Rijmen wird er auch *Rijndael-Algorithmus* genannt (gesprochen wie dt. *räindahl*).

Der Rijndael-Algorithmus besitzt variable, voneinander unabhängige, Block- und Schlüssellängen von 128, 160, 192, 224 oder 256 Bit. Rijndael bietet ein sehr hohes Maß an Sicherheit; erst mehr als zehn Jahre nach seiner Standardisierung wurde der erste, theoretisch interessante, praktisch aber nicht relevante Angriff gefunden. AES schränkt die Blocklänge auf 128 Bit und die Wahl der Schlüssellänge auf 128, 192 oder 256 Bits ein. Die Bezeichnungen der drei AES-Varianten AES-128, AES-192 und AES-256 beziehen sich jeweils auf die gewählte Schlüssellänge.

Der Algorithmus ist frei verfügbar und darf ohne Lizenzgebühren eingesetzt, sowie in Soft- und Hardware implementiert werden. AES-192 und AES-256 sind in den USA für staatliche Dokumente mit höchster Geheimhaltungsstufe zugelassen.^[3]

AES



Der Substitutionschritt, einer von 4 Teilschritten pro Runde

| | |
|----------------|--|
| Entwickler | Joan Daemen, Vincent Rijmen |
| Veröffentlicht | 1998, Zertifizierung Oktober 2000 |
| Abgeleitet von | Square |
| Zertifizierung | NESSIE |
| Schlüssellänge | 128, 192 oder 256 Bit |
| Blockgröße | 128 Bit ^[1] |
| Struktur | Substitution |
| Runden | 10, 12 oder 14 (schlüssellängenabhängig) |

Beste bekannte Kryptoanalyse

Der geheime Schlüssel kann bei AES-128 in $2^{126,1}$ Schritten, bei AES-192 in $2^{189,7}$ Schritten und bei AES-256 in $2^{254,4}$ Schritten gefunden werden.^[2]

Inhaltsverzeichnis

- 1 Entstehung
 - 1.1 Auswahl eines DES-Nachfolgers
- 2 Arbeitsweise
 - 2.1 S-Box
 - 2.2 Ablauf
 - 2.3 Schlüsselexpansion
 - 2.4 AddRoundKey
 - 2.5 SubBytes
 - 2.6 ShiftRows
 - 2.7 MixColumns
 - 2.8 Entschlüsselung
- 3 Anwendung
- 4 Schwächen und Angriffe

- 4.1 Kritikpunkte
- 4.2 Biclique-Angriff
- 4.3 XSL-Angriff
 - 4.3.1 Lösen überspezifizierter Systeme
 - 4.3.2 Anwendung auf Blockchiffren
- 4.4 Weitere Angriffe
- 5 Literatur
- 6 Weblinks
 - 6.1 Implementierungen
 - 6.1.1 C++-Bibliotheken
 - 6.1.2 C/ASM
 - 6.1.3 C# /.NET
 - 6.1.4 Delphi
 - 6.1.5 Java
 - 6.1.6 Andere Sprachen
- 7 Einzelnachweise

Entstehung

Bis zum Einsatz von AES war der Data Encryption Standard (DES) der am häufigsten genutzte symmetrische Algorithmus zur Verschlüsselung von Daten. Spätestens seit den 1990er Jahren galt er mit seiner Schlüssellänge von 56 Bit als nicht mehr ausreichend sicher gegen Angriffe mit der Brute-Force-Methode. Ein neuer, besserer Algorithmus musste gefunden werden.

Auswahl eines DES-Nachfolgers

Das amerikanische Handelsministerium schrieb die Suche nach einem Nachfolgealgorithmus am 2. Januar 1997 international aus, federführend für die Auswahl war das US-amerikanische National Institute of Standards and Technology in Gaithersburg, Maryland. Nach einer internationalen Konferenz am 15. April 1997 veröffentlichte es am 12. September 1997 die endgültige Ausschreibung. Die Art der Suche sowie die Auswahlkriterien unterschieden sich damit beträchtlich von der hinter verschlossenen Türen erfolgten DES-Entwicklung. Der Sieger der Ausschreibung, der als Advanced Encryption Standard (AES) festgelegt werden sollte, musste folgende Kriterien erfüllen:

- AES muss ein symmetrischer Algorithmus sein, und zwar eine Blockchiffre.
- AES muss 128 Bit lange Blöcke verwenden (dies wurde erst während der Ausschreibung festgelegt, zu Beginn der Ausschreibung waren auch Blockgrößen von 192 und 256 Bit verlangt, diese wurden nur als mögliche Erweiterungen beibehalten)
- AES muss Schlüssel von 128, 192 und 256 Bit Länge einsetzen können.
- AES soll gleichermaßen leicht in Hard- und Software zu implementieren sein.
- AES soll in Hardware wie Software eine überdurchschnittliche Leistung haben.
- AES soll allen bekannten Methoden der Kryptoanalyse widerstehen können, insbesondere Power- und Timing-Attacken.
- Speziell für den Einsatz in Smartcards sollen geringe Ressourcen erforderlich sein (kurze Codelänge, niedriger Speicherbedarf).
- Der Algorithmus muss frei von patentrechtlichen Ansprüchen sein und muss von jedermann unentgeltlich genutzt werden können.

Die Auswahlkriterien wurden in drei Hauptkategorien unterteilt: Sicherheit, Kosten, sowie Algorithmus- und Implementierungscharakteristiken. Die Sicherheit war der wichtigste Faktor in der Evaluierung und umfasste die Eigenschaften Widerstandsfähigkeit des Algorithmus gegen Kryptoanalyse, Zufälligkeit des Chiffrats, Stichhaltigkeit der mathematischen Basis, sowie die relative Sicherheit im Vergleich zu den anderen Kandidaten.

Kosten, der nächst wichtige Faktor, ist im Sinne des Auswahlverfahrens als Überbegriff zu verstehen: Dieser umfasste Lizenzierungsansprüche sowie rechnerische Effizienz auf verschiedenen Plattformen und Speicherverbrauch. Da eines der wichtigsten Ziele, die das NIST ausgearbeitet hatte, die weltweite Verbreitung auf lizenzfreier Basis war und dass AES von jedermann unentgeltlich genutzt werden kann, wurden öffentliche Kommentare und Anregungen zu Lizenzansprüchen und diesbezügliche potenzielle Konflikte spezifisch gesucht.

Die Anforderung der Geschwindigkeit des Algorithmus auf diversen Plattformen wurde in drei zusätzlichen Zielen unterteilt:

- Die rechnerische Geschwindigkeit mit 128-Bit-Schlüsseln.
- Die rechnerische Geschwindigkeit mit 192-Bit- und 256-Bit-Schlüsseln, sowie die rechnerische Geschwindigkeit verschiedener Hardware-Implementierungen. Der Speicherverbrauch und die Grenzen von Software-Implementierungen der Kandidaten waren weitere wichtige Aspekte.
- Das dritte Ziel, die Algorithmus- und Implementierungscharakteristiken, beinhalteten die Flexibilität, die Eignung für Soft- und Hardware-Implementierungen und die Einfachheit des Algorithmus.

Unter Flexibilität verstand man die Eigenschaften, dass AES die Schlüssel- und Blockgröße über dem Minimum unterstützen musste und, dass er in verschiedenen Typen von Umgebungen, sowie zusätzlich als Stromchiffre und kryptologische Hashfunktion sicher und effizient zu implementieren war.

Die Ausschreibung führte bis zum Abgabeschluss am 15. Juni 1998 zu fünfzehn Vorschlägen aus aller Welt. Diese wurden in der AES-Konferenz vom 20. bis 22. August 1998 in Ventura (Kalifornien) vorgestellt, öffentlich diskutiert und auf die Erfüllung der genannten Kriterien geprüft. Die AES-Konferenz vom 22. und 23. April 1999 in Rom führte zu einer ersten Diskussion der Ergebnisse und Empfehlungen, welche der fünfzehn Algorithmen weiter betrachtet werden sollten. Die fünf besten Kandidaten (MARS, RC6, Rijndael, Serpent, Twofish) kamen in die nächste Runde.

Alle fünf Kandidaten erfüllen die oben genannten Forderungen, daher wurden weitere Kriterien hinzugezogen. Es folgte eine Überprüfung der Algorithmen auf theoretische Schwachstellen, durch die der Algorithmus möglicherweise zu einem späteren Zeitpunkt durch technischen Fortschritt unsicher werden kann. So konnten zum damaligen Stand technisch nicht realisierbare Vorgehensweisen in einigen Jahren anwendbar sein, ein solches Risiko sollte minimiert werden. Die Staffeln der Kandidaten nach Ressourcenverbrauch und Leistung war eindeutiger. Der Rijndael-Algorithmus hatte sich in Hardware- und Software-Implementierung als überdurchschnittlich schnell herausgestellt. Die anderen Kandidaten haben jeweils in unterschiedlichen Bereichen kleinere Schwächen.

Im Mai des Jahres 2000 wurden die Analysen und öffentlichen Diskussionen abgeschlossen und am 2. Oktober 2000 der Sieger schließlich bekannt gegeben: der belgische Algorithmus Rijndael.

Rijndael überzeugte durch seine Einfachheit (die Referenz-Implementierung umfasst weniger als 500 Zeilen C-Code), Sicherheit und Geschwindigkeit, weshalb sich die USA trotz Sicherheitsbedenken für einen europäischen Algorithmus entschieden.

Der Auswahlprozess faszinierte weltweit viele Kryptographen insbesondere durch seine offene Gestaltung. Bis heute ist dieser Wettbewerb als sehr vorbildlich angesehen.

Arbeitsweise

Rijndael ist eine als Substitutions-Permutations-Netzwerk entworfene Blockchiffre. Bei Rijndael können Blocklänge und Schlüssellänge unabhängig voneinander die Werte 128, 160, 192, 224 oder 256 Bits erhalten, während bei AES die Einschränkung der festgelegten Blockgröße von 128 Bit und der Schlüsselgröße von 128, 192 oder 256 Bit gilt. Jeder Block wird zunächst in eine zweidimensionale Tabelle mit vier Zeilen geschrieben, deren Zellen ein Byte groß sind. Die Anzahl der Spalten variiert somit je nach Blockgröße von 4 (128 Bits) bis 8 (256 Bits). Jeder Block wird nun nacheinander bestimmten Transformationen unterzogen. Aber anstatt jeden Block einmal mit dem Schlüssel zu verschlüsseln, wendet Rijndael verschiedene Teile des erweiterten Originalschlüssels nacheinander auf den Klartext-Block an. Die Anzahl r dieser Runden variiert und ist von der Schlüssellänge k und Blockgröße b abhängig (beim AES also nur von der Schlüssellänge):

Anzahl der Runden bei Rijndael. (Die für AES relevanten Werte sind farbig unterlegt.)

| | b = 128 | b = 160 | b = 192 | b = 224 | b = 256 |
|----------------|----------------|----------------|----------------|----------------|----------------|
| k = 128 | 10 | 11 | 12 | 13 | 14 |
| k = 160 | 11 | 11 | 12 | 13 | 14 |
| k = 192 | 12 | 12 | 12 | 13 | 14 |
| k = 224 | 13 | 13 | 13 | 13 | 14 |
| k = 256 | 14 | 14 | 14 | 14 | 14 |

S-Box

Eine Substitutionsbox (S-Box) dient als Basis für eine monoalphabetische Verschlüsselung. Sie ist meist als Array implementiert und gibt an, wie in jeder Runde jedes Byte eines Blocks durch einen anderen Wert zu ersetzen ist. Typischerweise wird die S-Box in Blockchiffren eingesetzt, um die Beziehung zwischen Klar- und Geheimtext zu verwischen (in der kryptologischen Fachsprache *Konfusion* genannt). Die S-Box des AES setzt auch teilweise das Shannon'sche Prinzip der Diffusion um. Die Werte der S-Box und inversen S-Box können dynamisch berechnet werden um Speicher zu sparen oder vorberechnet sein und in einem Array gespeichert werden. Die S-Box besteht aus 256 Bytes, die konstruiert werden, indem zunächst jedes Byte außer der Null, aufgefasst als Vertreter des endlichen Körpers \mathbb{F}_{2^8} , durch sein multiplikatives Inverses ersetzt wird. Die Konstruktion der S-Box unterliegt Designkriterien, die die Anfälligkeit für die Methoden der linearen und der differentiellen Kryptoanalyse sowie für algebraische Attacken minimieren sollen.

Ablauf

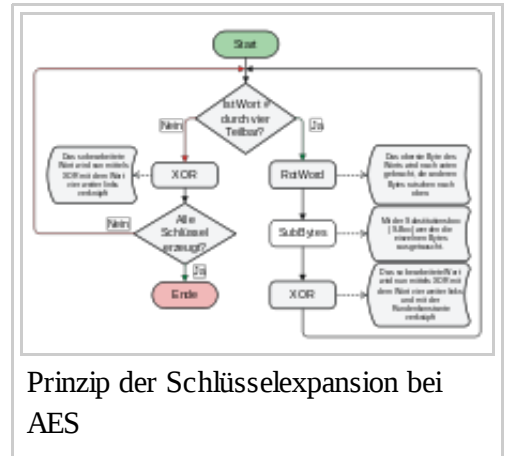
- Schlüsselexpansion
- Vorrunde
 - AddRoundKey(Rundenschlüssel[0])
- Verschlüsselungsrunden ($r = 1$ bis $R-1$)
 - SubBytes()
 - ShiftRows()
 - MixColumns()
 - AddRoundKey(Rundenschlüssel[r])
- Schlussrunde
 - SubBytes()

- ShiftRows()
- AddRoundKey(Rundenschlüssel[R])

(Die Schlusssrunde zählt auch als Runde, also $R = \text{Anzahl Verschlüsselungsrunden} + 1 \text{ Schlusssrunde}$)

Schlüsselexpansion

Zunächst müssen aus dem Schlüssel $R + 1$ Teilschlüssel (auch Rundenschlüssel genannt) erzeugt werden. Die Rundenschlüssel müssen die gleiche Länge wie die Blöcke haben. Somit muss der Benutzerschlüssel auf die Länge $b * (R + 1)$ expandiert werden, wobei b die Blockgröße angibt. Der Schlüssel wird in eine zweidimensionale Tabelle mit vier Zeilen und Zellen der Größe 1 Byte abgebildet. Die ersten Spalten der Tabelle werden mit dem Benutzerschlüssel gefüllt. Die weiteren Spalten werden wie folgt rekursiv berechnet: Um die Werte für die Zellen in der nächsten Spalte zu erhalten, werden die Spalten, welche je nach Blockgröße ein Vielfaches der vierten, sechsten oder achten Spalte sind, nach links rotiert ($[a_0, a_1, a_2, a_3]$ wird zu $[a_1, a_2, a_3, a_0]$) und mit Hilfe der S-Box verschlüsselt. Im Anschluss wird der „vorderste“ Wert der Spalte mit der rcon-Tabelle XOR verknüpft und abschließend die gesamte Spalte mit der um eine Schlüssellänge zurückliegenden Spalte XOR verknüpft. Die rcon-Tabelle ist ähnlich wie die S-Box eine Tabelle in Form eines Arrays, das konstante Werte, in diesem Fall die Zweierpotenzen, enthält. Jede andere Spalte wird aus einer XOR-Verknüpfung mit der Spalte eine Schlüssellänge vorher gebildet. Eine Besonderheit bildet AES-256. Dort wird jede 4. Spalte (also eine Spalte, welche normalerweise ohne Rotation und so weiter auskommt) durch die S-Box ersetzt und dann mit der Spalte eine Schlüssellänge zuvor XOR verknüpft.



Prinzip der Schlüsselexpansion bei AES

AddRoundKey

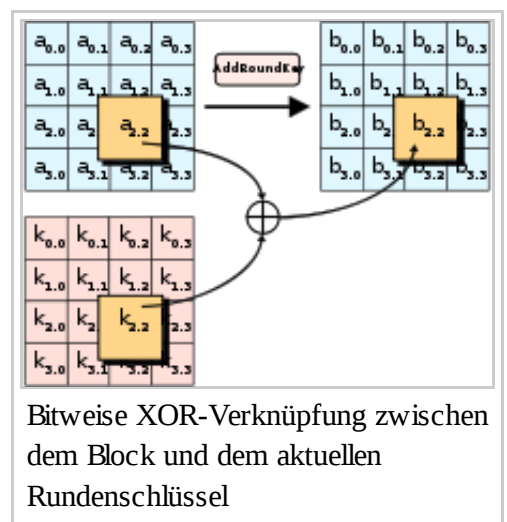
In der Vorrunde und am Ende jeder weiteren Verschlüsselungsrunde wird die KeyAddition ausgeführt. Hierbei wird eine bitweise XOR-Verknüpfung zwischen dem Block und dem aktuellen Rundenschlüssel vorgenommen. Dies ist die einzige Funktion in AES, die den Algorithmus vom Benutzerschlüssel abhängig macht.

SubBytes

Im ersten Schritt jeder Runde wird für jedes Byte im Block ein Äquivalent in der S-Box gesucht. Somit werden die Daten monoalphabetisch verschlüsselt.

ShiftRows

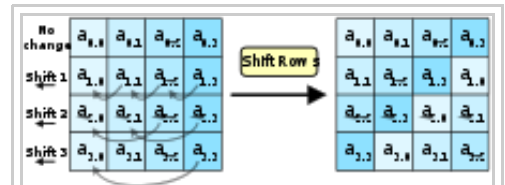
Wie oben erwähnt, liegt ein Block in Form einer zweidimensionalen Tabelle mit vier Zeilen vor. In diesem Schritt werden die Zeilen um eine bestimmte Anzahl von Spalten nach links verschoben. Überlaufende Zellen werden von rechts fortgesetzt. Die Anzahl der Verschiebungen ist zeilen- und blocklängenabhängig:



Je nach Blocklänge b und Zeile in der Datentabelle wird die Zeile um 1 bis 4 Spalten verschoben

Für den AES sind nur die fett markierten Werte relevant

| r | $b=128$ | $b=160$ | $b=192$ | $b=224$ | $b=256$ |
|----------------|----------|---------|---------|---------|---------|
| Zeile 1 | 0 | 0 | 0 | 0 | 0 |
| Zeile 2 | 1 | 1 | 1 | 1 | 1 |
| Zeile 3 | 2 | 2 | 2 | 2 | 3 |
| Zeile 4 | 3 | 3 | 3 | 4 | 4 |



Zeilen werden um eine bestimmte Anzahl von Spalten nach links verschoben

MixColumns

→ Hauptartikel: Rijndael MixColumns^[4]

Schließlich werden die Daten innerhalb der Spalten vermischt. Zur Berechnung eines Bytes b_j der neuen Spalte wird jedes Byte a_j der alten mit einer Konstanten (1, 2 oder 3) multipliziert. Dies geschieht modulo des irreduziblen Polynoms $x^8 + x^4 + x^3 + x + 1$ im Galois-Körper $GF(2^8)$. Dann werden die Ergebnisse XOR-verknüpft:

$$\begin{aligned} b_0 &= (a_0 \cdot 2) \oplus (a_1 \cdot 3) \oplus (a_2 \cdot 1) \oplus (a_3 \cdot 1) \\ b_1 &= (a_0 \cdot 1) \oplus (a_1 \cdot 2) \oplus (a_2 \cdot 3) \oplus (a_3 \cdot 1) \\ b_2 &= (a_0 \cdot 1) \oplus (a_1 \cdot 1) \oplus (a_2 \cdot 2) \oplus (a_3 \cdot 3) \\ b_3 &= (a_0 \cdot 3) \oplus (a_1 \cdot 1) \oplus (a_2 \cdot 1) \oplus (a_3 \cdot 2) \end{aligned}$$

In Matrixschreibweise:

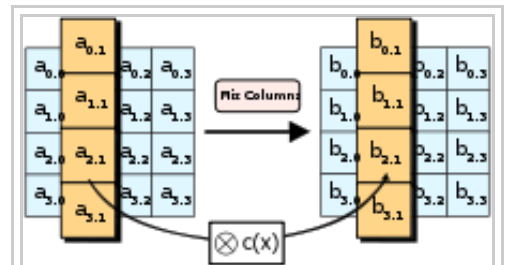
$$\begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix}$$

Nach den Rechengesetzen in diesem Galois-Körper gilt für die Multiplikation:

- $a \cdot 1 = a$
- $a \cdot 2 = \begin{cases} 2a & \text{wenn } a < 2^7 \\ 2a \oplus (11b)_{\text{hex}} & \text{wenn } a \geq 2^7 \end{cases}$
- $a \cdot 3 = (a \cdot 2) \oplus a$

Dabei bezeichnet $2a$ die normale Multiplikation von a mit 2 und \oplus die bitweise XOR-Verknüpfung.

Entschlüsselung



Die Spalten werden vermischt

Bei der Entschlüsselung von Daten wird genau rückwärts vorgegangen. Die Daten werden zunächst wieder in zweidimensionale Tabellen gelesen und die Rundenschlüssel generiert. Allerdings wird nun mit der Schlussrunde angefangen und alle Funktionen in jeder Runde in der umgekehrten Reihenfolge aufgerufen. Durch die vielen XOR-Verknüpfungen unterscheiden sich die meisten Funktionen zum Entschlüsseln nicht von denen zum Verschlüsseln. Jedoch muss eine andere S-Box genutzt werden (die sich aus der originalen S-Box berechnen lässt) und die Zeilenverschiebungen erfolgen in die andere Richtung.

Anwendung

AES wird u. a. vom Verschlüsselungsstandard IEEE 802.11i für Wireless LAN und seinem Wi-Fi-Äquivalent WPA2, bei IEEE802.16 m (WiMAX), sowie bei SSH und bei IPsec genutzt. Auch in der IP-Telefonie kommt AES sowohl in offenen Protokollen wie SRTP oder proprietären Systemen wie Skype^[5] zum Einsatz. Mac OS X benutzt AES als Standardverschlüsselungsmethode für Disk-Images, außerdem verwendet der Dienst *FileVault* AES. Ebenso verwendete die transparente Verschlüsselung EFS in Windows XP ab SP 1 diese Methode. Außerdem wird der Algorithmus zur Verschlüsselung diverser komprimierter Dateiarhive verwendet, z. B. bei 7-Zip und RAR. In PGP und GnuPG findet AES ebenfalls einen großen Anwendungsbereich.

AES gehört zu den vom Projekt NESSIE empfohlenen kryptografischen Algorithmen und ist Teil der Suite B der NSA.

Der AES-Algorithmus wird inzwischen in etlichen CPUs von Intel oder AMD durch zusätzliche spezialisierte Maschinenbefehle unterstützt, wodurch das Verschlüsseln 5 mal und das Entschlüsseln 25 mal schneller als mit nicht spezialisierten Maschinenbefehlen erfolgt.^[6] Damit ist AES auch für mobile Anwendung Akku-schonend benutzbar und für den Masseneinsatz geeignet. Programmier-Softwarebibliotheken wie zum Beispiel OpenSSL erkennen automatisch ob die Hardware AES unterstützt und nutzen dann die Hardware-AES-Implementierung statt der langsameren Softwareimplementierung.

Schwächen und Angriffe

Kritikpunkte

Zum Rijndael-Algorithmus gab es auch kritische Stimmen mancher Kryptographen, welche folgende Punkte betrafen:

- Rijndael überzeugte im AES-Wettbewerb durch seine mathematisch elegante und einfache Struktur, sowie durch seine Effizienz. Allerdings könnte gerade diese einfache Struktur ein Einfallstor für Angriffstechniken sein.
- Je nach verwendetem Schlüssel bestand bei Rijndael nur eine Sicherheitsmarge von 3 (bei 128 Bits Schlüssellänge) bis 5 Runden (bei 256 Bits Schlüssellänge).^[7]
- Als weiterer Kritikpunkt galt die einfache algebraische Beschreibung der S-Boxen, die ihrerseits die einzige nichtlineare Komponente der Chiffre sind. Dadurch lässt sich der gesamte Algorithmus als Gleichungssystem beschreiben.^[7]
- Die Eigenschaft, dass sich durch den einfachen Key Schedule aus Kenntnis eines beliebigen Rundenschlüssels 128 Bit des Verfahrensschlüssels trivial gewinnen lassen können.

Biclique-Angriff

Auf der Rump-Session der Konferenz CRYPTO im August 2011 stellten die Kryptologen Andrey Bogdanov, Dmitry Khovratovich und Christian Rechberger den ersten Angriff auf den vollen AES-Algorithmus vor.^[2] Dieser Angriff ist bei den verschiedenen Schlüssellängen im Schnitt etwa um den Faktor 4 schneller als ein vollständiges Durchsuchen des Schlüsselraumes. Damit zeigt er die prinzipielle Angreifbarkeit von AES, ist aber für die praktische Sicherheit nicht relevant. Der Angriff berechnet den geheimen Schlüssel von AES-128 in $2^{126,1}$ Schritten. Bei AES-192 werden $2^{189,7}$ Schritte, bei AES-256 $2^{254,4}$ Schritte benötigt.

XSL-Angriff

Lösen überspezifizierter Systeme

Im Jahre 1999 zeigten Aviad Kipnis und Adi Shamir, dass ein spezifischer Public-Key-Algorithmus, bekannt als Hidden Field Equations (HFE), zu einem überspezifizierten System quadratischer Gleichungen reduziert werden konnte. Um ein solches System zu lösen, wird eine Technik namens Linearisierung benutzt, mit welcher man jeden quadratischen Term mit einer unabhängigen Variable ersetzt und das daraus folgende Lineare System nach dem gaußschen Eliminationsverfahren löst. Um bei der Linearisierung erfolgreich zu sein, benötigt diese genügend linear unabhängige Gleichungen, welche idealerweise die gleiche Anzahl wie die der Terme aufweisen. Für die Kryptoanalyse des HFE konnten allerdings nicht genügend Gleichungen erstellt werden. Aus diesem Grund schlugen Kipnis und Shamir die re-Linearisierung vor: eine Technik, bei welcher nicht lineare Gleichungen nach der Linearisierung hinzugefügt werden. Das daraus hervorgehende System wird wiederum mit einer zweiten Anwendung der Linearisierung gelöst.

Anwendung auf Blockchiffren

2002 wurde von Courtois und Pieprzyk ein theoretischer Angriff namens XSL („eXtended Sparse Linearization“) gegen Serpent und Rijndael vorgestellt (siehe Serpent). Mit dem XSL-Angriff ist nach Angabe der Autoren eine Komplexität im Bereich von 2^{200} Operationen erreichbar. XSL ist die Weiterentwicklung einer heuristischen Technik namens XL („eXtended Linearization“), mit der es manchmal gelingt, große nichtlineare Gleichungssysteme effizient zu lösen. XL wurde ursprünglich zur Analyse von Public-Key-Verfahren entwickelt. Der Einsatz im Kontext von symmetrischen Kryptosystemen ist eine Innovation von Courtois und Pieprzyk. Grob kann die Technik und ihre Anwendung auf symmetrische Kryptosysteme wie folgt beschrieben werden:

Die Blockchiffre wird als überspezifiziertes System quadratischer Gleichungen in $GF(2)$ beschrieben. Überspezifiziert bedeutet, dass es mehr Gleichungen als Variablen gibt. Variablen und Konstanten können nur die Werte 0 und 1 annehmen. Die Addition entspricht dem logischen exklusiv-Oder (XOR), die Multiplikation dem logischen UND. Eine solche Gleichung könnte wie folgt aussehen:

$$x_1 + x_2x_3 + x_2x_4 = 1 \pmod{2}.$$

Diese Gleichung besteht aus einem linearen Term (der Variablen „ x_1 “), zwei quadratischen Termen („ x_2x_3 “ und „ x_2x_4 “) und einem konstanten Term („1“).

Einige Wissenschaftler zweifeln aber die Korrektheit der Abschätzungen von Courtois und Pieprzyk an:

“I believe that the Courtois-Pieprzyk work is flawed. They overcount the number of linearly independent equations. The result is that they do not in fact have enough linear equations to solve the system, and the method does not break Rijndael ... The method has some merit, and is worth investigating, but it does not break Rijndael as it stands.”

„Ich glaube, dass die Arbeit von Courtois und Pieprzyk fehlerhaft ist; sie schätzen die Anzahl der linear unabhängigen Gleichungen zu hoch ein. Das Resultat ist, dass sie in Wirklichkeit nicht genug lineare Gleichungen erhalten, um das System zu lösen, und die Methode somit Rijndael nicht knackt [...] Die Methode besitzt ihre Vorzüge und ist es wert, weiter untersucht zu werden, allerdings knackt sie in ihrer aktuellen Form Rijndael nicht“

– DON COPPERSMITH^[8]

Diese Art von System kann typischerweise sehr groß werden, im Falle der 128-Bit-AES-Variante wächst es auf 8.000 quadratische Gleichungen mit 1.600 Variablen an, womit der XSL-Angriff in der Praxis nicht anwendbar ist. Das Lösen von Systemen quadratischer Gleichungen ist ein NP-schweres Problem mit verschiedenen Anwendungsfeldern in der Kryptographie.

Weitere Angriffe

Kurz vor der Bekanntgabe des AES-Wettbewerbs stellten verschiedene Autoren eine einfache algebraische Darstellung von AES als Kettenbruch vor. Dies könnte für erfolgreiche Angriffe genutzt werden. Hierzu gibt es einen Videovortrag von Niels Ferguson auf der HAL 2001.^[9]

Im Jahr 2003 entdeckten Sean Murphy und Matt Robshaw eine alternative Beschreibung des AES, indem sie diesen in einer Blockchiffre namens BES einbetteten, welche anstatt auf Datenbits auf Datenblöcke von 128 Bytes arbeitet. Die Anwendung des XSL-Algorithmus auf BES reduziert dessen Komplexität auf 2^{100} , wenn die Kryptoanalyse von Courtois und Pieprzyk korrekt ist.

Im Mai 2005 veröffentlichte Daniel Bernstein einen Artikel über eine unerwartet einfache Timing-Attacke^[10] (eine Art der Seitenkanalattacke) auf den Advanced Encryption Standard.

Die Forscher Alex Biryukov und Dmitry Khovratovich veröffentlichten Mitte des Jahres 2009 einen Angriff mit verwandtem Schlüssel^[11] auf die AES-Varianten mit 192 und 256 Bit Schlüssellänge. Dabei nutzten sie Schwächen in der Schlüsselexpansion aus und konnten eine Komplexität von 2^{119} erreichen. Damit ist die AES-Variante mit 256 Bit Schlüssellänge formal schwächer als die Variante mit 128 Bit Schlüssellänge.^[12] Ende 2009 wurde mit einer Verbesserung des Angriffs eine Komplexität von nur noch $2^{99,5}$ erreicht.^[13] Für die Praxis hat dieser Angriff jedoch wenig Relevanz, denn AES bleibt weiterhin praktisch berechnungssicher.

Im März 2012 wurde bekannt, dass die NSA in ihrem neuen Utah Data Center neben dem Speichern großer Teile der gesamten Internetkommunikation auch mit enormen Rechenressourcen an dem Knacken von AES arbeiten wird.^[14] Die Eröffnung des Rechenzentrums ist im laufenden Jahr 2013 geplant.

Literatur

- Joan Daemen, Vincent Rijmen: *The Design of Rijndael. AES: The Advanced Encryption Standard*. Springer, Berlin u. a. 2002, ISBN 3-540-42580-2 (*Information Security and Cryptography*), (Englisch).

Weblinks

- Offizielle Spezifikation des AES vom NIST (<http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>)

(PDF-Datei; 273 kB)

- Angriffe auf die Sicherheit von AES (<http://www.cryptosystem.net/aes/>)
- Ausführlichere deutsche Erklärung des Algorithmus und Historie des Auswahlverfahrens (<http://www.realtec.de/privat/arbeiten.shtml>) (PDF)
- Beschreibung von Markus Repges der AES-Kandidaten (Finalisten) (<http://www.repges.net/AES-Kandidaten/aes-kandidaten.html>)
- NIST, Report on the Development of the Advanced Encryption Standard (AES), 2. Oktober 2000 (<http://csrc.nist.gov/archive/aes/round2/r2report.pdf>) (PDF-Datei; 383 kB)
- Flashanimation von AES (http://www.formaestudio.com/rijndaelinspector/archivos/Rijndael_Animation_v4_eng.swf) – AES als Flash erklärt und animiert (Animation by Enrique Zabala / Universität ORT / Montevideo / Uruguay)
- AESPipe (<http://loop-aes.sourceforge.net/>) – Komfortable Ver-/Entschlüsselung von Datenströmen für die Linuxshell (StdIn nach StdOut)
- AES Artikel (<http://www.codeplanet.eu/tutorials/cpp/51-advanced-encryption-standard.html>) – Sehr detaillierte deutsche Erklärung des AES mitsamt Rechenbeispielen und Implementierung in der Programmiersprache C
- Applied Crypto++: Block Ciphers (<http://www.codeproject.com/KB/security/BlockCiphers.aspx>) Ein Artikel über Crypto++ auf codeproject.com mit dem Titel *Encrypt Data using Symmetric Encryption with Crypto++*

Implementierungen

C++-Bibliotheken

- Crypto++ (<http://www.cryptopp.com/>) Eine vom NIST offiziell genehmigte sehr umfangreiche quelloffene C++-Kryptographie-Bibliothek mit diversen Verschlüsselungs- und Hash-Algorithmen.
- Chris Lomonts gemeinfreie Version des AES (<http://www.lomont.org/Software/Misc/AES/AES.php>)
- Botan (<http://botan.randombit.net/>) BSD-lizenzierte kryptographische Bibliothek
- Libgcrypt (<http://directory.fsf.org/wiki/Libgcrypt>) Kryptographische Bibliothek unter der GNU General Public License

C/ASM

- OpenSSL (<http://www.openssl.org/>)
- Nettle-Bibliothek (<http://www.lysator.liu.se/~nisse/nettle/>) (unter GPL)
- Polar SSL (<http://polarssl.org/aes-source-code>) (unter GPL)
- Eine kompakte AES-256-Implementierung (<http://www.literatecode.com/2007/11/11/aes256/>) (unter OpenBSD-Lizenz)
- Byte-orientierte gemeinfreie Implementierung (<http://code.google.com/p/byte-oriented-aes/downloads/list>)
- Implementierung von Brian Gladman (http://gladman.plushost.co.uk/oldsite/cryptography_technology/index.php) (unter OpenBSD-Lizenz)
- Implementierung von D.J. Bernstein (<http://cr.yp.to/mac.html>)
- Implementierung von Philip J. Erdelsky (<http://www.efgh.com/software/rijndael.htm>)

C#/.NET

- *Keep Your Data Secure with the New Advanced Encryption Standard*

- (<http://msdn2.microsoft.com/en-us/magazine/cc164055.aspx>) Eine detaillierte Erklärung mit C#-Implementierung von James D. McCaffrey
- In der Version 3.5 des .NET Framework, beinhaltet der System.Security.Cryptography (<http://msdn.microsoft.com/en-us/library/system.security.cryptography.aspx>)-Namespace je eine komplett gemanagte (<http://msdn.microsoft.com/en-us/library/system.security.cryptography.aesmanaged.aspx>) AES-Implementierung und ein gemanagter Wrapper (<http://msdn.microsoft.com/en-us/library/system.security.cryptography.aescryptoserviceprovider.aspx>) um die CAPI AES-Implementierung.
- Bouncy Castle C# API (<http://www.bouncycastle.org/csharp/>)

Delphi

- Martin Offenwangers AES-Implementierung (http://www.dsplayer.de/dspweb/public_downloads/BTAES_0.3.zip) (ZIP; 47 kB) in Delphi geschrieben, GPL-lizenziert
- Arnaud Bouchez' AES-Implementierung (<http://bouchez.info/delphi-crypto.html>) in Delphi und i386-Assembler geschrieben
- DCPcrypt (<http://www.cityinthesky.co.uk/cryptography.html>): OIS-Certified open source David Bartons Delphi-Implementierung, als Teil einer Hash- und Cyphersuite
- Delphi Encryption Compendium (<http://code.google.com/p/delphidec/>) quelloffene Kryptographische Bibliothek mit allen gängigen Verschlüsselungsalgorithmen, Freeware/MIT
- TurboPower Lockbox (<http://sourceforge.net/projects/tplockbox/>) quelloffene AES-Implementierung.
- SongBeamer (<http://www.songbeamer.com/delphi/>), eine Delphi-2010-kompatible Version von TurboPower Lockbox.
- Delphi-Interface zur Implementierung von Brian Gladman (http://gladman.plushost.co.uk/oldsite/cryptography_technology/index.php) für Delphi 2009 (<http://www.rathlev-home.de/sources/download/d2009/aeslib.zip>) (ZIP; 69 kB) und Delphi 7 (<http://www.rathlev-home.de/sources/download/aeslib.zip>) (ZIP; 71 kB)

Java

- Java Cryptography Extension von Sun Microsystems, integriert in der Java-JRE seit Version 1.4.2
- IAIK Java Cryptography Extension (<http://jce.iaik.tugraz.at/>) von IAIK, Technische Universität Graz
- Bouncy castle (cryptography) (<http://bouncycastle.org/>) Offizielle Bouncy Castle Crypto Library

Andere Sprachen

- tcllib zur Sprache Tcl, ab Version 1.8
- mcrypt (<http://php.net/manual/de/book.mcrypt.php>) zur Sprache PHP
- System.Security.Cryptography (<http://msdn.microsoft.com/en-us/library/system.security.cryptography.aspx>) C#, .NET
- Crypt::Rijndael (<http://search.cpan.org/search?query=Crypt-Rijndael&mode=dist>) für Perl
- Crypto-JS (<http://code.google.com/p/crypto-js>) für JavaScript
- Stanford Javascript Crypto Library (<http://crypto.stanford.edu/sjcl/>) für JavaScript
- RijndaelVB (<http://www.freevbcode.com/ShowCode.asp?ID=2389>) für VB6
- PyCrypto (<http://www.dlitz.net/software/pycrypto/>) für Python
- as3crypto (<http://code.google.com/p/as3crypto/>) für ActionScript

- [ruby-aes \(http://raa.ruby-lang.org/project/ruby-aes/\)](http://raa.ruby-lang.org/project/ruby-aes/) für Ruby

Einzelnachweise

1. Im Rijndael-Algorithmus werden Blockgrößen von 128, 160, 192, 224, und 256 Bits unterstützt, im AES-Standard wird aber nur eine 128-bit Blockgröße spezifiziert.
2. Andrey Bogdanov, Dmitry Khovratovich, Christian Rechberger: *Biclique Cryptanalysis of the Full AES*. In: *ASIACRYPT 2011 (= Lecture Notes in Computer Science. 7073)*. Springer, 2011, S. 344–371 (<http://research.microsoft.com/en-us/projects/cryptanalysis/aesbc.pdf>).
3. Committee on National Security Systems: *CNSS Policy No. 15, Fact Sheet No. 1*. 2003, S. 2 (<http://csrc.nist.gov/groups/ST/toolkit/documents/aes/CNSS15FS.pdf>).
4. Laurent Haan: Advanced Encryption Standard (AES) (<http://www.codeplanet.eu/tutorials/cpp/51-advanced-encryption-standard.html>), 17. Februar 2008
5. Tom Berson: *Skype Security Evaluation* (<http://www.skype.com/security/files/2005-031%20security%20evaluation.pdf>) auf skype.com mit Signatur (<http://www.anagram.com/bereson/skyeval.sig>), 18. Oktober 2005, englisch, PDF
6. Oliver Lau (2013): „Spezialkommando. Schnelle AES-Chiffres mit Intrinsic“ in: c't 2013 Heft 14, Seiten 174-177. Zitierte Aussage siehe Seite 176 und 177.
7. Niels Ferguson, Bruce Schneier: *Practical Cryptography*. Wiley Publishing, Indianapolis 2003, ISBN 0-471-22357-3, S. 56.
8. <http://www.schneier.com/crypto-gram-0210.html#8>
9. ftp://ftp.ccc.de/events/hal2001/video/hal2001_cryptoanalis_of_rijndael_48.mp4
10. Cache-timing attacks on AES (PDF-Version; 426 kB) (<http://cr.yp.to/antiforgery/cachetiming-20050414.pdf>)
11. Related-key Cryptanalysis of the Full AES-192 and AES-256 (PDF) (<http://cryptolux.uni.lu/mediawiki/uploads/1/1a/Aes-192-256.pdf>)
12. FAQ zum Angriff (https://cryptolux.org/FAQ_on_the_attacks)
13. Biryukov, Alex; Khovratovich, Dmitry (4. Dezember 2009): „Related-key Cryptanalysis of the Full AES-192 and AES-256“ (<http://eprint.iacr.org/2009/317>)
14. http://www.wired.com/threatlevel/2012/03/ff_nsadatacenter/all/

Von „http://de.wikipedia.org/w/index.php?title=Advanced_Encryption_Standard&oldid=121727795“

Kategorie: Blockverschlüsselung

-
- Diese Seite wurde zuletzt am 21. August 2013 um 02:42 Uhr geändert.
 - Abrufstatistik

Der Text ist unter der Lizenz „Creative Commons Attribution/Share Alike“ verfügbar; Informationen zum Lizenzstatus eingebundener Mediendateien (etwa Bilder oder Videos) können im Regelfall durch Anklicken dieser abgerufen werden. Möglicherweise unterliegen die Inhalte jeweils zusätzlichen Bedingungen. Durch die Nutzung dieser Website erklären Sie sich mit den Nutzungsbedingungen und der Datenschutzrichtlinie einverstanden.

Wikipedia® ist eine eingetragene Marke der Wikimedia Foundation Inc.