**Advanced Encryption Standard** (AES)

Bisher war der Data Encryption Standard (DES) der am häufigsten genutzte symmetrische Algorithmus zum Verschlüsseln von Daten. Trotzdem wurde er oft stark kritisiert. Da er lange Zeit nicht vollständig veröffentlicht war, wurden sogar Hintertüren der NSA vermutet, die allerdings nie gefunden wurden. Zweifel an der Sicherheit von DES waren jedoch begründet (schließlich benutzt DES nur einen 56 Bit langen Schlüssel) und wurden durch die Electronic Frontier Foundation bestätigt, die 1998 eine Maschine zur Dechiffrierung von DES mittels Brute-Force-Verfahren bauten. 3DES bot sich als eine vorübergehende Lösung an, ist jedoch recht langsam und die verwendete Blocklänge von 64 Bit ist ein potenzielles Sicherheitsrisiko. Es musste also ein neuer Verschlüsselungsstandard entwickelt werden, der nicht die Fehler seiner Vorgänger besitzt.

**Auswahl eines Nachfolgers**
Das US-amerikanische National Institute of Standards and Technology (NIST) hat Anfang 1997 zu einem offenen Wettbewerb aufgerufen, dessen Sieger als Advanced Encryption Standard (AES) festgelegt werden soll. Dabei wurden folgende Kriterien aufgestellt, die von den Algorithmen zu erfüllen sind:

- AES muss ein symmetrischer Algorithmus sein, und zwar ein Blockalgorithmus.
- AES muss mindestens 128 Bit lange Blöcke verwenden und Schlüssel von 128, 192 und 256 Bit Länge einsetzen können.
- AES soll gleichermaßen leicht in Hard- und Software zu implementieren sein.
- AES soll in Hard- wie Software eine überdurchschnittliche Performance haben.
- AES soll allen bekannten Methoden der Kryptanalyse (die Kunst, einen Geheimtext ohne Kenntnis des Schlüssels zu dechiffrieren) 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 darf von jedermann unentgeltlich genutzt werden.

Im August 1998 sind 15 Algorithmen beim NIST eingegangen und werden nun öffentlich diskutiert und auf die Erfüllung der genannten Kriterien geprüft. Die engere Wahl ist im April 1999 beendet und die fünf besten Kandidaten (MARS, RC6, Rijndael, Serpent, Twofish) kommen in die nächste Runde.

Alle fünf Kandidaten erfüllen die Forderungen. Daher werden ab nun weitere Kriterien hinzugezogen: Die Algorithmen sollen auf theoretische Schwachstellen überprüft werden, durch die ein Algorithmus möglicherweise irgendwann einmal unsicher werden kann. Dies klingt sehr weit hergeholt, ist aber notwendig. Denn leider kann man heute nicht sagen, wie sich der Technologie-Fortschritt in den nächsten Jahren entwickelt. Vorgehensweisen die heute als unmöglich erklärt sind können in einigen Jahren schon alltäglich sein. Ein solches Risiko darf nicht eingegangen werden.

Eindeutiger ist jedoch die Staffelung der Kandidaten nach Ressourcenverbrauch und Performance. Als einziger Algorithmus ist Rijndael als Hard- und Software-Implementierung überdurchschnittlich schnell. Andere Kandidaten haben jeweils in unterschiedlichen Bereichen Schwächen.

Im Mai 2000 wurden die Analysen und öffentlichen Diskussionen abgeschlossen und schließlich am 2. Oktober 2000 der Sieger bekannt gegeben: Der belgische Algorithmus Rijndael wird neuer Standard. Sicherlich entspricht das nicht der üblichen Sicherheitspolitik in den USA, dass man sich künftig auf einen europäischen Algorithmus berufen soll. Aber Rijndael hat insbesondere durch seine Einfachheit (die Referenz-Implementierung umfasst weniger als 500 Zeilen C-Code) und Performance überzeugt.

**Blockverschlüsselungsalgorithmen**
Symmetrische Verschlüsselungsalgorithmen sind oft Blockverschlüsselungsalgorithmen. Bevor sie Daten verschlüsseln, werden diese zunächst in Blöcke unterteilt, dessen Größe durch den Algorithmus vorgegeben ist und bei AES sogar variabel ist. Die Blöcke werden anschließend jeder für sich verschlüsselt. Die Verschlüsselungsroutine setzt jedoch ganze Blöcke voraus, somit muss meistens der letzte Block mit Zufallsdaten aufgefüllt werden (bei einer Blockgröße von 128 Bit werden die verschlüsselten Daten somit maximal 16 Byte größer als der Klartext). Wichtige Parameter eines Blockchiffre sind somit die Blocklänge und die Schlüssellänge.

**XOR-Verknüpfung**
Viele Verschlüsselungsalgorithmen arbeiten mit einer XOR-Verknüpfung. XOR ist die Bezeichnung für ein exklusives "oder". Praktisch entspricht dies der Addition zweier Bits modulo 2:

| XOR | 0 | 1 |
|---|---|---|
| **0** | 0 | 1 |
| **1** | 1 | 0 |

Der Vorteil der XOR-Verknüpfung ist, dass sie umkehrbar ist. Somit kann beim Verschlüsseln die gleiche Funktion verwendet werden, wie beim Entschlüsseln:
101 XOR 11 = 110
110 XOR 11 = 101
110 XOR 101 = 11

**S-Box**
Eine Substitutionsbox (S-Box) dient als Basis für eine monoalphabetische Verschlüsselung. Sie ist meist als Array implementiert und gibt an, welches Byte wie getauscht wird. Die S-Box in AES basiert auf einem mathematischen Zusammenhang und ist somit fest im Algorithmus implementiert. Doch dies ist kein Schwachpunkt, da die S-Box lediglich zur Vermischung der Bytes in Kombination mit weiteren Operationen und dem Schlüssel genutzt wird.

**Die Arbeitsweise von AES**
AES (in Zukunft ist mit AES der Algorithmus Rijndael gemeint) ist, wie bereits erwähnt, ein Blockchiffre, dessen Blocklänge und Schlüssellänge unabhängig voneinander die Werte 128, 192 oder 256 Bit erhalten kann. Jeder Block wird zunächst in eine zweidimensionale Tabelle mit vier Zeilen geschrieben, dessen Zellen ein Byte groß sind. Die Anzahl der Spalten variiert je nach Blockgröße von 4 (128 Bit) bis 8 (256 Bit). Jeder Block wird nun nacheinander bestimmten Transformationen unterzogen. Aber anstatt jeden Block einmal mit dem Schlüssel zu Chiffrieren, wendet AES verschiedene Teile des Schlüssels nacheinander auf den Klartext-Block an. Die Anzahl dieser Runden (r) variiert und ist von Schlüssellänge (k) und Blockgröße (b) abhängig:

| r | b = 128 | b = 192 | b = 256 |
|---|---|---|---|

| | | | |
|---|---|---|---|
| **k = 128** | 10 | 12 | 14 |
| **k = 192** | 12 | 12 | 14 |
| **k = 256** | 14 | 14 | 14 |

**Ablauf von AES**

- Schlüsselexpansion
- Vorrunde
  - KeyAddition ()
- Verschlüsselungsrunden (wiederhole solange runde<r)
  - Substitution()
  - ShiftRow()
  - MixColumn()
  - KeyAddition()
- Schlussrunde
  - Substitution()
  - ShiftRow()
  - KeyAddition()

**Schlüsselexpansion**
Zunächst muss der Schlüssel in r+1 Teilschlüssel (auch Rundenschlüssel genannt) aufgeteilt werden. Die Rundenschlüssel müssen die gleiche Länge, wie die Blöcke erhalten. Somit muss der Benutzerschlüssel auf die Länge b * (r +1) expandiert werden. Die Schlüssel werden wieder in zweidimensionalen Tabellen mit vier Zeilen und Zellen der Größe 1 Byte gebildet. Der erste Rundenschlüssel ist identisch mit dem Benutzerschlüssel. Die weiteren Schlüssel werden wie folgt berechnet:
Um die Werte für die Zellen in der ersten Spalte eines Rundenschlüssels zu erhalten, wird zunächst das Byte, welches sich in der letzten (je nach Blockgröße: vierten, sechsten oder achten) Spalte befindet und drei Zeilen zurückliegt, durch die S-Box verschlüsselt und anschließend mit dem um eine Schlüssellänge zurückliegendem Byte XOR verknüpft.
Befindet sich das zu berechnende Byte an einer Position, die ein ganzzahliger Teiler der Schlüssellänge ist, so wird das berechnete Byte zusätzlich mit einem Eintrag aus der rcon-Tabelle XOR verknüpft. Hierfür wird als Index eine fortlaufende Nummer verwendet. Die rcon-Tabelle ist ähnlich wie die S-Box eine Tabelle in Form eines Arrays, das konstante Werte enthält die auf einem mathematischen Zusammenhang beruhen.
Jedes weitere Byte (das sich nicht in der ersten Spalte befindet) wird aus einer XOR-Verknüpfung des vorherigen Bytes ($w_{i-1}$) mit dem Byte einer Schlüssellänge vorher ($w_{i-k/32}$) gebildet.

**KeyAddition**
In der Vorrunde und nach 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.

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

**ShiftRow**
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:

| | **b = 128** | **b = 192** | **b = 256** |
|---|---|---|---|
| **Zeile 0** | 0 | 0 | 0 |
| **Zeile 1** | 1 | 1 | 1 |
| **Zeile 2** | 2 | 2 | 3 |
| **Zeile 3** | 3 | 3 | 4 |

**MixColumn**
Schließlich werden die Spalten vermischt. Es wird zunächst jede Zelle einer Spalte mit einer Konstanten multipliziert und anschließend die Ergebnisse XOR verknüpft. Hinter dieser Vorgehensweise steckt ein komplizierter mathematischer Zusammenhang, der hier nicht näher erläutert wird. Die Konstante wird folgendermaßen bestimmt:
Zeile 1: 2
Zeile 2: 3
Zeile 3: 1
Zeile 4: 1

**Entschlüsselung**
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 original S-Box berechnen lässt) und die Zeilenverschiebungen erfolgen in die andere Richtung.

Kristof Hamann, 2002