Implementierung einer Smartphone-Anwendung zum Austausch verschlüsselter Daten mit einer Cloud

22. Juli 2014

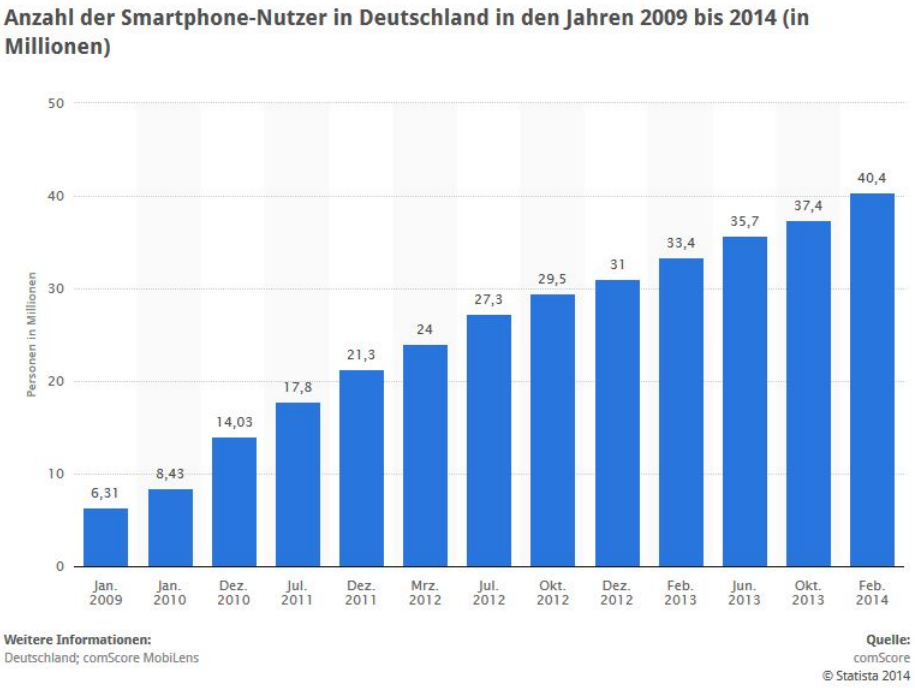
"If you think technology can solve your security problems, then you don't understand the problems and you don't understand the technology."

Bruce Schneier

Inhaltsverzeichnis

1 Einleitung

"Die Computer- und Internetnutzer in Deutschland setzen seit Bekanntwerden der geheimdienstlichen Abhöraktionen häufiger Verschlüsselungsverfahren ein.Äus der Pressemitteilung der BITKOM geht weiterhin hervor, dass von Juli 2013 auf November 2013 insgesamt 1,1 Millionen mehr Bundesbürger ihre persönlichen Dateien verschlüsseln. Besonders wichtig ist der Aspekt der Sicherheit, wenn es sich bei den Daten um relevante oder firmeninterne Informationen handelt, wie es z. B. am in Hamburg der Fall ist. Auch der Austausch von Daten von mobilen Endgeräten wie oder Tables spielen eine immer größere Rolle wie die Entwicklung der letzten Jahre zeigt (siehe Grafik).



Herkömmliche Verfahren zum Austausch von Daten reichen oftmals nicht mehr aus, wenn man den Aspekt der Sicherheit näher beleuchtet.

1.1 Motivation

Am Deutschen Elektronen Synchrotron, im folgenden DESY, werden bisher wichtige und sensible Dokumente über ein Programm Namens Dropbox gesichert und verwaltet. Dropbox bietet eine plattformunabhänginge Möglichkeit Dokumente Online abzuspeichern und von einem anderen Standort über ein internetfähiges Gerät wieder zu öffnen [https://www.dropbox.com/]. Auch wenn Dropbox nach eigenen

Angaben den Advanced Encryption Standard (AES) verwendet, bevor die Daten gespeichert werden, liegen die dafür notwendigen Schlüssel in Händen der Betreiber selbst, die somit vollen Klartextzugriff auf die Nutzerdateien haben. Dropbox begründet diesen Zugriff wie folgt: "Wie die meisten Online-Dienste verfügt auch Dropbox über einen kleinen Mitarbeiterstamm, dem aus in unserer Datenschutzrichtlinie dargelegten Gründen Zugriffsrechte auf Nutzerdaten gewährt werden muss [...]".
Da das DESY über eine eigene Cloud-Infrastruktur verfügt, sollen in Zukunft alle wichtigen Daten nicht nur in dieser Cloud gespeichert werden, sondern auch zusätzlich durch eine Verschlüsselung gesichert werden. Die Cloud am DESY stellt im Hintergrund ein Rechnernetz zum Abspeichern von Daten zur Verfügung. Durch das Programm dCache, welches das Rechnernetz im Hintergrund steuert und verwaltet, ist es dem Anwender möglich Daten in das System zu speichern, ohne dessen Struktur zu kennen. dCache sorgt dafür dass die Daten, je nach Bedarf, mehrfach abgelegt werden und bei einem Zugriff schnell zur Verfügung stehen. Die Dateien selbst werden im Hintergrund auf verschiedene Datenträger, wie z. B. SSD-Festplatten, Magnetbänder, Tapes o. ä., abgelegt. Das System sorgt dafür, dass bei reger Anfrage die Daten, sofern möglich, auf ein schnelleres Medium repliziert werden. Die genaue Struktur und Vorgehensweise des Programmes ist jedoch nicht Teil dieser Arbeit, da das hier zu entwickelnde Programm nur die Schnittstelle des dCache-Servers verwendet.

1.2 Zielsetzung

Ziel dieser Arbeit ist es aus diesem Grund einen Prototyp zu entwickeln, der einerseits mit dem Cloud-System des DESY Kommunizieren kann um dort Dateien hoch- und herunter zu laden, andererseits diese Daten auch in angemessener Form (siehe Kapitel Validierung) zu Verschlüsseln.
In der ersten Version dieser Arbeit wird ein Programm entwickelt, welches auf Android-Betriebssystemen zum Einsatz kommen kann. Darüber hinaus ist es wichtig, dass die entsprechenden Schlüssel zum entschlüsseln der Daten nicht zusammen mit den Daten abgelegt werden, sondern ausschließlich den Parteien des Datenaustauschs bekannt sein soll. Dies bedeutet, das selbst die Betreiber am DESY nicht die Möglichkeit haben die abgelegten Daten zu entschlüsseln.
Zum Ver- und Entschüsseln der Daten sollen Verfahren verwendet werden, die in der heutigen Zeit als sicher angesehen werden und Smartphones im Bezug auf Performance und Akkuverbrauch nicht zu stark belasten. Um diese Faktoren zu Validieren wird eine Testanwendung geschrieben, die mit bestimmten Faktoren die verschiedenen Verfahren untereinander überprüfen (siehe Kapitel Validierung).

1.3 Verwandte Arbeiten

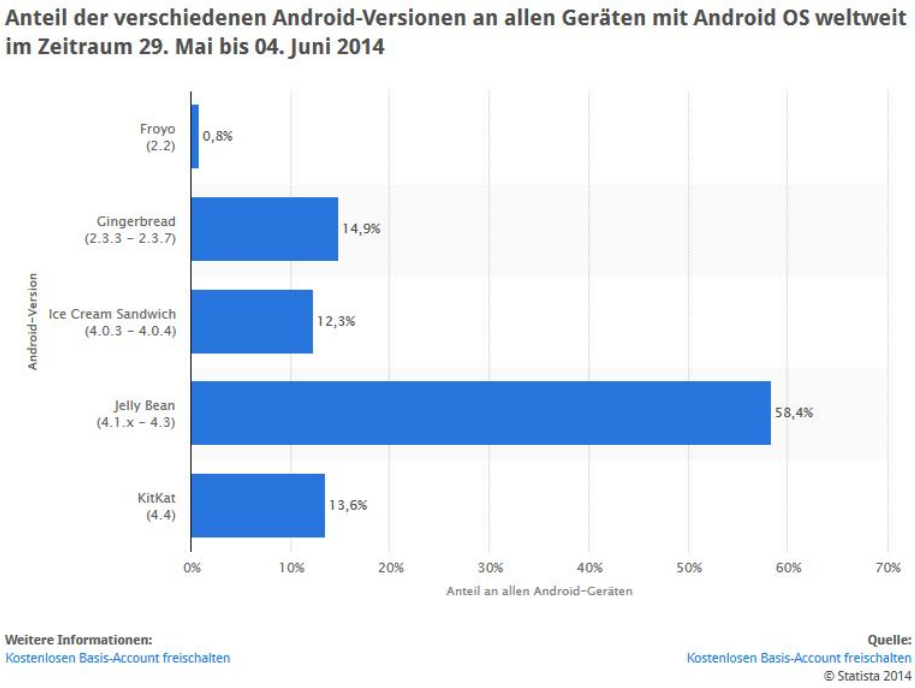
1.4 Verwandte Programme

1.5 Diese Arbeit

1.5.1 Inhaltlicher Aufbau

2 Grundlagen Android

Android ist ein Betriebssystem für Smartphones und Tablets, welches von der open handset alliance entwickelt wird. Das Konsortium besteht aktuell aus 84 Unternehmen, die an der Entwicklung des Betriebssystems arbeiten. In diesem Kapitel wird kurz darauf eingegangen, welche Kryptografischen Aspekte Android in den verschiedenen Versionen zur Verfügung stellt um diese im darauffolgenden Kapitel genauer zu Untersuchen. Aufgrund der Tatsache, dass die Android Version Gingerbread (2.3.3) im Juni 2014 noch einen Marktanteil von knapp 15% hält, ist dies auch die niedrigste vom Programm unterstützte Version.



Bei der Analyse wird darauf geachtet, dass alle Funktionalitäten die im Programm entwickelt werden, von dieser Version unterstützt werden. Die, während des Schreibens dieser Arbeit, aktuellste Version der Android API ist KitKat (4.4), bei der darauf geachtet wird, dass die eingesetzten Funktionen auch in dieser Version noch zur Verfügung stehen und nicht mit *deprecated (veraltet)* markiert sind.

2.1 Zusammenhang Kryptographie

Beim Thema Sicherheit im Zusammenhang mit Android ist erstmals der Begriff der Sandbox zu nennen. Eine Anwendung wird abgekapselt in einer eigenen Umgebung mit eigenem Prozess, eigenem Betriebssystem-

User, eigener Dalvik-VM, eigenem Heap und eigenem Dateisystem ausgeführt. Dieses abgekapselte Konstrukt wird Sandbox bezeichnet. Dadurch ist es dem Betriebssystem möglich unerlaubten Zugriff auf Ressourcen oder andere Programme zu beschränken, hierbei wird das Berechtigung- und Prozess-Management-System von Linux verwendet. Um dennoch verschiedene Zugriffe zu erlauben muss in der sogenannten Manifest-Datei der Anwendung die Berechtigung festgelegt und vom Benutzer bei der Erstinstallation bestätigt werden. Auch wenn dieses Konzept Daten zur Laufzeit innerhalb einer Anwendung schützt, ist es möglich Dateien auch auf einer SD-Karte zu speichern, in das Internet zu verschicken oder über andere Wege auszutauschen. Diese Daten sind dann außerhalb der Anwendung und gegen externe Zugriffe nicht mehr geschützt. Dennoch gibt es die Möglichkeit in Android diese Daten zusätzlich mit einer Verschlüsselung zu versehen - hierfür stellt Java, seit der Version 1.4, die *Java Cryptography Extension (JCE)* innerhalb von Android zur Verfügung. Innerhalb der Erweiterung (engl. extension) sind verschiedene Provider eingebunden, die dem Programmierer die Möglichkeit geben Kryptografische Verfahren aufzurufen, ohne die genaue Implementierung kennen zu müssen. In Java-Anwendungen gibt es diverse Implementierungen von Sun, die jedoch aus Datenschutzrechtlichen Gründen nicht in der Android Java-API vorhanden sind. der Provider Bouncy-Castle stellt eine Alternative zur Implementierung von Sun darf und wird in Android zur Verfügung gestellt. Innerhalb von Android wurde das Paket so geändert, dass es den Richtlinien der JCE entspricht. Bouncy-Castle ist einer der von Android zur Verfügung gestellten Provider - jedoch gibt es noch weitere Provider, die selbige oder andere Implementierungen zur Verfügung stellen. Mit folgendem Codeabschnitt ist es möglich die einzelnen Provider mit den unterstützten Verfahren auszulesen und untereinander zu vergleichen. Dieser Code wurde auf verschiedenen Versionen ausgeführt, um die Unterschiede der einzelnen Versionen hervorzuheben.

```
Provider[] providers = Security.getProviders();
for (Provider provider : providers) {
    Log.i("CRYPTO","provider: "+provider.getName());
    Set<Provider.Service> services = provider.getServices();
    for (Provider.Service service : services) {
        Log.i("CRYPTO","  algorithm: "+service.getAlgorithm());
    }
}
```

Im Vergleich stehen folgende Android-Versionen:

- 2.3.3 (Gingerbread) : die niedrigste vom Programm unterstützte Version
- 4.1.1 (Jelly Bean) : die Version des Entwickler-Gerätes
- 4.4.4 (KitKat) : aktuellste auf dem Markt verfügbare Android-Version
- "L" : zukünftige Version, welche bereits zu Testzwecken als Entwickler-Version freigeschaltet ist. Der Codename "L" zeigt auf, dass es sich in der Folge der Süßigkeiten (Gingerbread, HoneyComb, Ice Cream Sandwich, Jelly Bean, KitKat) vermutlich alphabetisch fortsetzen wird - die Versionsnummer ist bis dato nicht bekannt.

Im Vergleich der Ausgabe eines Gerätes mit Android 2.3.3 und eines mit 4.1.1 bzw. 4.4.4 und "L" liegt der Hauptunterschied in der Unterstützung von Elliptischen Kurven für das Diffie-Hellmann-Verfahren

(ECDH) und den Digital-Signature-Algorithm (ECDSA), welche in der Version 2.3.3 nicht unterstützt werden. Des Weiteren ist ab der Version 4.4.x der Provider OpenSSL und deren Algorithmen spezifischer dargestellt. Folgende Verschlüsselungsverfahren werden sowohl von der Version 2.3.3 als auch von der Version 4.1.1 , 4.4.4 und L unterstützt und werden im nachfolgenden Kapitel näher erläutert:

Verschluesselung	**Hash-Funktion**
AES	MD5
ARC4	SHA1
Blowfish	SHA256
DES	SHA384
3DES	SHA512
PBE	
RSA	

Des Weiteren wird das Key-Wrap-Verfahren (AES und 3DES), der Authentifizierungsalgorithmus HMAC und der Standard X509 beschrieben. Die Vollständige Liste aller Unterstützen Algorithmen mit dessen Providern befindet sich im Anhang. Welcher Provider für welchen Algorithmus besser geeignet ist, kann man nicht pauschalisieren und auch nicht auf einen spezifischen Anwendungsfall verallgemeinern. Im Kapitel Validierung werden Geschwindigkeitsaspekte beider großen Provider (OpenSSL und Bouncy-Castle), sofern möglich, gegenübergestellt um für jeden Algorithmus den geeigneten Provider zu wählen. Falls es innerhalb eines Providers zu größeren Sicherheitslücken von einem verwendeten Algorithmus kommt, ist es durch das JCE möglich diesen ohne weitere Code-Änderungen zu wechseln.

3 Grundlagen Kryptologie

Das Wort Kryptologie stammt aus dem Griechischen *kryptós* für verstecken und *lógos* für die Lehre.Dieser Zweig umfasst die Kryptographie - die Wissenschaft die sich mit der Absicherung von Daten beschäftigt, die Kryptoanalyse - welche für das Aufbrechen von Geheimnachrichten zuständig ist sowie der Mathematik. Im Bereich der Kryptologie ist es das Ziel eine Nachricht, welche aus lesbaren Zeichen (Klartext) besteht unverständlich zu machen (Verschlüsseln) und daraus einen Geheimtext (Chiffretext) zu erzeugen. Dieses Verfahren wird mit mathematischen Funktionen und einem Schlüssel (Key) durchgeführt. Die Umkehrung von Chiffretext in Klartext (Entschlüsselung) wird ebenfalls durch eine mathematische Funktion und einen Schlüssel durchgeführt. Ziel dieser Ver- und Entschlüsselung ist es Nachrichten zwischen einem Sender und Empfänger so auszutauschen, dass ein Angreifer diese nicht mitlesen, oder im verschärftem Sinne nicht verändern kann. Hierbei besteht eine Nachricht in der Informatik immer aus binären Daten und kann eine Textdatei, ein Bild, ein Video oder vieles mehr darstellen. Ver- und Entschlüsselung sind mathematische Funktionen, die auf den Klartext, bzw. auf den Chiffretext angewendet werden.

Terminologie

Um die Lesbarkeit zu erhöhen wird Klartext im folgenden mit M (engl. Message), Chiffretext mit C (engl. Chiffre), die Verschlüsselungsfunktion mit E (engl. Encoding), die Entschlüsselungsfunktion mit D (engl. Decoding) und dem Schlüssel K (engl. Key) beschrieben. Zum Verschlüsseln kommt also folgende Funktion zum Einsatz:

$\mathrm{E}_K(\mathrm{M}) = \mathrm{C}$

Um den Chiffretext wieder zu Entschlüsseln wird die umgekehrte Richtung angewandt:

$\mathrm{D}_K(\mathrm{C}) = \mathrm{M}$

Zusammengefasst muss also gelten: das Verschlüsseln einer Nachricht und das darauffolgende Entschlüsseln des erzeugten Chiffretextes, mit der dazugehörigen Funktion und korrektem Schlüssel, muss wieder den Klartext ergeben. Mathematisch beschrieben ist das wie folgt:

$\mathrm{D}_K(\mathrm{E}_K(\mathrm{M})) = \mathrm{M}$

Um einen sicheren Kanal zwischen Sender und Empfänger zu gewährleistet, reicht es nicht allein die Nachricht zu verschlüsseln. Authentifizierung, Integrität und Verbindlichkeit müssen darüber hinaus gewährleistet sein um sicher zu Kommunizieren.

Authentifizierung beschreibt hierbei das Verfahren indem sich die Identität einer Person beweisen lässt. Im Umkehrschluss bedeutet das, dass sich ein Angreifer nicht als eine andere Person ausgeben kann. Aus der Authentifizierung folgt dann die **Autorisierung**, also das Prüfen, ob der Benutzer die Rechte hat, die er fordert.

Integrität bedeutet, dass sichergestellt werden kann, dass eine Nachricht bei der Übermittlung zwischen Sender und Empfänger nicht durch einen Angreifer verändert wurden ist.

Verbindlichkeit beschreibt dass der Sender nicht leugnen kann, dass eine Nachricht gesendet wurde.

Kerhoff's Maxime

Ein Aspekt in der Kryptographie sind die Kerkhoffs' Maxime, die folgendes Aussagen: "the security of the encryption scheme must depend only on the secrecy of the Key K_e, and not on the secrecy of the algorithms." [Schneier, 1997] Übersetzt bedeutet es, dass die Sicherheit eines Kryptographischen Verfahrens auf der Geheimhaltung des Schlüssels beruhen muss und nicht auf derer des Verschlüsselungsalgorithmus.

Verfahren

Prinzipiell unterteilt man Kryptographie in zwei Verschiedene Verfahren. Die symmetrischen Verfahren und die asymmetrischen, auch public key infrastructure genannt. Generell lässt sich über das "bessere Verfahren" keine Aussage treffen, da es für beide Verfahren Vor- und Nachteile gibt. Bruce Schneier fasste es wie folgt zusammen:
"Symmetrische Kryptographie eignet sich am besten zur Verschlüsselung von Daten. Sie ist um Größenordnungen schneller und nicht anfällig für chosen-ciphertext-Angriffe. Public-Key-Kryptographie schafft Dinge, die außerhalb des Einsatzbereichs symmetrischer Kryptographie liegen und eignen sich am besten für die Schlüsselverwaltung und eine Vielzahl der Protokolle [...]."
Der im Zitat verwendete Ausdruck, chosen-ciphertext-Angriff beschreibt einen Angriff auf ein Kryptosystem, bei dem der Kryptoanalytiker verschiedene Chiffretexte zur Entschlüsselung auswählen kann und entsprechend Zugriff auf den dazugehörigen Klartext besitzt. Die Aufgabe bei dieser Art des Angriffes besteht darin, den entsprechenden Schlüssel herauszufinden. Neben der chosen-ciphertext-Angriffe gibt es weitere Angriffsszenarien auf Kryptosysteme, wie z. B. ciphertext-only, known-plaintext, chosen-plaintext, chosen-key und weitere. Da es sich bei dieser Arbeit nicht um eine Kryptoanalyse eines Systems handelt, werden diese Szenarien nicht näher erläutert. Es wird davon ausgegangen, dass wenn eines dieser Szenarien zum knacken des Systems führt, dieses kryptographische Verfahren bereits heute als unsicher angesehen wird.

3.1 Symmetrische Verfahren

Bei symmetrischen Verschlüsselungsverfahren existiert ein Schlüssel, der jeweils für Ver- und Entschlüsselung verwendet wird. Dieser Schlüssel muss bereits beiden Parteien bekannt sein, bevor ein verschlüsselter

Kanal aufgebaut werden kann. Eines der Probleme bei symmetrischen Verfahren ist der Austausch des Schlüssels, den man von Sender zu Empfänger, bereits vor der sicheren Kommunikation, übertragen muss (siehe Kapitel Schlüsselvereinbarung). Symmetrische Verfahren unterteilt man in zwei Grundtypen, die Block- und Stromchiffrierung. Bei der Blockchiffrierung wird der Klartext in Blöcke, mit fester Größe, aufgeteilt und innerhalb des Blockes werden die mathematischen Funktionen angewandt. Bei der Stromchiffrierung werden die Daten nicht in Blöcken zusammengefasst, sondern jedes einzelne Klartextbit wird in ein Chiffrebit überführt. [Schneier 1996, Seite 223]

3.1.1 Betriebsmodi

3.1.2 DES, 3DES

"Its restricted key size of 56 bits and small block size of 64 bits make it unsuitable for today's fast computers and large amounts of data. It survives in the form of 3DES, which is a block cipher built from three DES encryptions in sequence. This solves the most immediate problem of the small key size, but there is no known fix for the small block size. [...] we do not recommend using either DES oder 3DES in new designs." [Ferguson 2003, Seite 51] Niels Ferguson weißt darauf hin, dass DES aufgrund seiner Schlüsselänge von 56 bits und der Blockgröße von 64 bits ungeeignet für heutige Systeme ist. Weiterhin beschreibt er, dass auch durch 3DES das Problem der geringen Blockgröße nicht behoben wird und er schlussfolgert dass man in heutigen neuen Systemen beide Verfahren nicht verwenden sollte.
Da das System als unsicher angesehen ist, wird auf eine nähere Untersuchung und Erläuterung der mathematischen Funktionen verzichtet. DES und 3DES wird in der zu entwickelnden Anwendung nicht implementiert.

3.1.3 AES

Der Advanced Encryption Standard, im folgenden AES genannt, ist ein Verschlüsselungsverfahren welches auf Blockchiffrierung beruht und seit 2002 ein offizieller Standard ist. Entwickelt wurde der neue Standard mit dem Namen Rijndael bei einer Ausschreibung für einen neuen Sicherheitsstandard durch J. Daemen und V. Rijmen, die sich gegen 14 andere Konkurrenten durchsetzen konnten. Auch unter den besten 5 dieser Ausschreibung waren die Verfahren MARS, RC6, Serpent und Twofish, wobei keiner dieser fünf eine Sicherheitsschwäche aufwies. Rijndael konnte letztendlich durch seine einfache Struktur und gute Performance im Software-, sowie im Hardwarebereich überzeugen. Darüber hinaus hat die US National Security Agency (NSA) AES für interne Dokumente bis zum Sicherheitsstatus TOP SECRET, mit einer Schlüssellänge von 192 oder 256 und für den Sicherheitsstatus SECRET mit einer Schlüsselänge von 128 Bit erlaubt, was verdeutlicht, dass selbst Kryptographen von Geheimdiensten diesen Standard als sicher ansehen.

Funktionsweise

AES arbeitet mit einer Blockgröße von 128 Bit, also 16 Byte welcher intern als 2-Dimensionale Matrix (4x4) abgespeichert wird und auf der mathematische Funktionen angewandt werden. Alle Funktionen innerhalb von AES werden Byteweise ausgeführt (8 Bit-Blöcke). Die interne Nachricht bezeichnet man als *state*, also den aktuellen Status des Blockes.

A_0	A_4	A_8	A_{12}
A_1	A_5	A_9	A_{13}
A_2	A_6	A_{10}	A_{14}
A_3	A_7	A_{11}	A_{15}

Die Schlüssellänge des Verfahren ist entweder 128, 192 oder 256 Bit, wovon auch die auszuführende Rundenzahl abhängt (10 Runden bei 128 Bit, 12 bei 192 und 14 bei 256 Bit Schlüssellänge). In jeder dieser Runden werden Verfahren angewandt um den Klartext weiter zu verschlüsseln. Bei AES sind das *Byte-Substitution, ShiftRow, MixColumns* und *KeyAddition*.

Ausgenommen von dieser Regel ist die letzte Runde, in der MixColumns ausgelassen wird. Zusätzlich wird vor der ersten Runde die Funktion KeyAddition angewendet.

ByteSubstitution

In der Funktion ByteSubstitution wird eines der beiden Verfahren zum Verbergen von Redundanz angewendet - die Konfusion. Die Konfusion sorgt dafür, dass der Zusammenhang zwischen Klartext und Chiffrat verschleiert wird und möglichst aus einer kleiner Änderung im Klartext eine große Änderung im Chiffrat erzeugt wird. Hierzu wird jedes Byte in eine sogenannte S-Box eingegeben, wobei diese wiederum ein Byte aus Ausgabewert hat (Dieses Verfahren wird für alle 16 Byte eines Blockes angewandt). Die S-Box selbst ist eine 16x16 Matrix, mit der zu jeder eingegebenen Bit-Reihenfolge eine neue Ausgabe-Reihenfolge erzeugt wird. Ziel ist es, durch minimale Veränderung des Eingabewertes eine maximale Veränderung des Ausgabewertes zu erzeugen. Darüber hinaus ist die in AES verwendete S-Box nicht linear - das bedeutet, dass die Addition zweier einzelner Ausgabewerte nicht das selbe Ergebnis liefert wie die Addition zweier Eingabewerte:

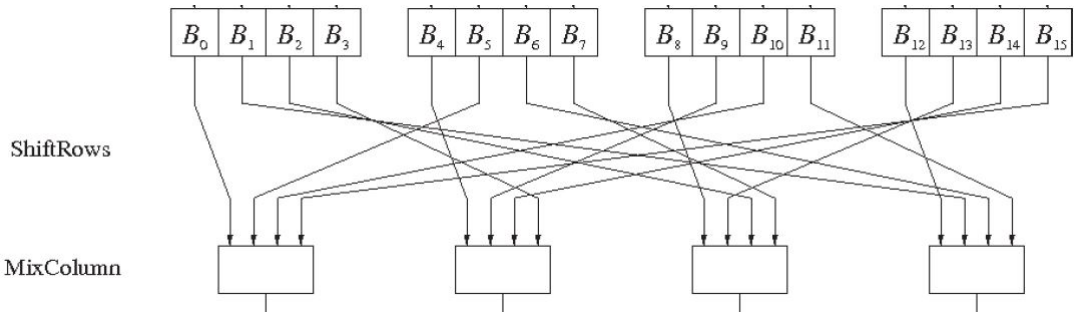
$$S(A) + S(B) \neq S(A + B)$$

Zusätzlich ist die S-Box bijektiv, es existiert also zu jeder Bitreihenfolge genaue eine eindeutige Zuweisung - Bitreihenfolgen die zwei Ausgabewerte erzeugen können existieren nicht. Im Umkehrschluss bedeutet das, dass jedes Ausgabebyte der S-Box wieder durch eine inverse S-Box zurück transformiert werden kann (Wird bei der Entschlüsselung verwendet). Die S-Boxen innerhalb von AES sind alle identische, sodass 16x pro Runde immer die selbe Matrix verwendet wird. Das hat zur Folge, dass sie in den den meisten Softwareimplementierungen durch fixe Tabellen realisiert werden, anstatt sie jedes mal neu zu Berechnen. Die Berechnung einer S-Box erfolgt durch Endliche Körper und eine fixe Multiplikation und Addition um die Logik der endlichen Körper zu verwischen. [c. Paar, Seite 102

ShiftRow

Das zweite Verfahren zum Vergeben von Redundanz ist die Diffusion, bei der die Redundanz verteilt wird (Einfachster Anwendungsfall ist das Vertauschen der Klartextbuchstaben in eine neue Reihenfolge).

Eine Funktion die innerhalb von AES die für Diffusion sorgt, ist das ShiftRow-Verfahren. Hierbei werden die Bytes innerhalb einer Spalte des state-Blockes auf alle anderen Spalten aufgeteilt. Eine Änderung innerhalb einer Spalte (A_0 bis A_3 der state-Matrix) hat somit Auswirkung auf alle anderen Spalten (Auswirkung auf komplette State-Matrix). Folgende Grafik soll das verdeutlichen, wobei B_0, B_1, ..., B_{15} jeweils die Bytes A_0, A_1, ..., A_{15} nach der Transformation durch die S-Box sind. In der Grafik ist die interne 4x4 Matrix (state) hier Spaltenweise nebeneinander abgebildet. (Vergleiche state-Matrix)



Die in der Grafik gezeigten Linien, die die Verschiebung darstellen sollen, ist innerhalb der state-Matrix durch einfaches Shifting realisiert. Hierbei wird in der ersten Zeile keine Verschiebung durchgeführt, in der zweiten Zeile wird jedes Byte um 1 nach Links rotiert, in der dritten 2 nach Links und in der vierten Zeile 3 nach Links.

Input matrix

B_0	B_4	B_8	B_{12}
B_1	B_5	B_9	B_{13}
B_2	B_6	B_{10}	B_{14}
B_3	B_7	B_{11}	B_{15}

Output matrix

B_0	B_4	B_8	B_{12}	no shift
B_5	B_9	B_{13}	B_1	← one position left shift
B_{10}	B_{14}	B_2	B_6	← two positions left shift
B_{15}	B_3	B_7	B_{11}	← three positions left shift

MixColumns

Die MixColumns-Funktion ist die zweite Funktion in AES die für die Diffusion sorgt - sie bewirkt, dass die Änderung eines einzigen Eingabebytes in die Funktion alle Ausgabebytes verändert. Hierbei wird jede Spalte (als Vector dargestellt) mit einer festen 4x4 Matrix multipliziert.

$$\begin{pmatrix} C_0 \\ C_1 \\ C_2 \\ C_3 \end{pmatrix} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \cdot \begin{pmatrix} B_0 \\ B_5 \\ B_{10} \\ B_{15} \end{pmatrix}$$

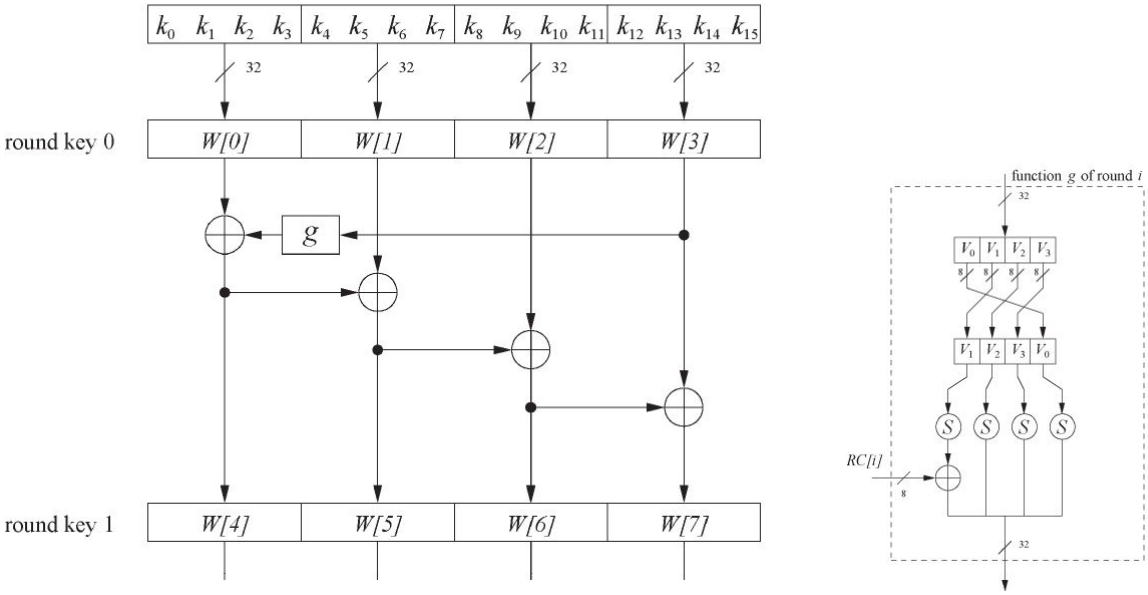
In der Grafik beschreibt der Vector B_0, B_5, B_{10}, B_{15} genau die erste Spalte nach der Verschiebung durch ShiftRow. Durch die starke Diffusion die durch die Verteilung der Bytes von einer Spalte auf alle Spalten in der Funktion ShiftRow und die Vermischung aller Bytes durch die MixColumns-Funktion erreicht wird, ist es dem Verfahren AES mögilch in 3 Runden jedes Byte des Klartextes von allen 16Byte der state-Matrix abhängig zu machen. Wenn also die zu verschlüsselnde Nachricht aus einer 1 und restlichen Nullen besteht wird diese 1 in nur 3 Runden auf alle anderen Nullen Auswirkung zeigen.

KeyAddition

Beim KeyAddition wird jeweils der aktuelle Block (4x4 state-Matrix) mit dem aktuellen Rundenschlüssel (16 byte) via XOR Bitweise verknüpft.

Rundenschlüssel

AES erzeugt für die verschiedenen Runden, die bei der Ver- und Entschlüsselung durchlaufen werden Rundenschlüssel (1 Schlüssel mehr als Runden die durchlaufen werden), welche in 4x 32-Bit große Blöcke (Word-oriented) abgespeichert werden. In der ersten Runde entspricht der Rundenschlüssel dem Original AES-Schlüssel (W[0] bis W[3] = 4x32Bit = 128Bit Schlüssellänge). Der letzte Word-Block einer Runde, wird dann durch eine Funktion gegeben und mit den anderen Blöcken XOR-Verknüpft.



Die Funktion g rotiert hierbei jeweils die Eingabebytes und führt sie durch eine nichtlineare S-Box. Am Ende dieses Verfahrens wird noch die Rundennummer mittels XOR dem linken Teilblock zugefügt. Das Ergebnis dieser Durchführung (W[4] bis W[7]) ist dann der Rundenschlüssel für die erste Runde. Dieses Verfahren wird wiederholt bis alle Rundenschlüssel entsprechend berechnet wurden. Die Anzahl der benötigten Rundenschlüssel und damit verbunden mit den benötigten Word-Blöcke erhöht sich mit der Erhöhung der Schlüssellänge von AES.

Entschlüsselung

Für die Entschlüsselung eines AES-Chiffretextes müssen alle Funktionen in umgekehrter Reihenfolge und umgekehrter Logik (Inverse Funktionen) ausgeführt werden. So hat man z. B. in der letzten Runde der Verschlüsselung die Funktion MixColumns nicht ausgeführt - so wird man in der ersten Runde der Entschlüsselung diese Funktion ebenfalls nicht ausführen. Darüber hinaus muss man für alle fixen Matrizen, die verwendet wurden eine inverse Matrix erstellen (S-Boxen, MixColumns-Matrix). Das Shifting in der Funktion ShiftRow erfolgt bei der Entschlüsselung dann entsprechend nach Rechts, anstatt nach Links

wie bei der Entschlüsselung. Ausgenommen von der Umgekehrten Logik ist die Berechnung der Rundenschlüssel - da man in der ersten Runde der Entschlüsselung den letzten Rundenschlüssel benötigt, der bei der Verschlüsselung eingesetzt wurde, müssen zu Beginn der Entschlüsselung erstmals alle Rundenschlüssel berechnet werden um diese dann zu verwenden. Die Berechnung der Rundenschlüssel selbst ist identisch.

3.1.4 ARC4

RC4, oder auch ARC4 (Arcfour) genannt ist eine Stromverschlüsselung, wird also Bitweise ent- und verschlüsselt. Nach dem Aufdecken geheimer Informationen der NSA durch den Whistleblower Edward Snowden, hat der Kryptograph Jacob Appelbaum (Mitentwickler des Sicherheitsnetzwerkes Tor und Unterstützer von WikiLeaks) auf Twitter einen Post geteilt in dem er sagt, dass mit RC4 verschlüsselte Daten von der NSA in Echtzeit entschlüsselt werden können: "RC4 is broken in real time by the #NSA - stop using it." Diese Behauptung wird auch von Bruce Schneier (Experte für Kryptographie, Entwickler der Verfahren Blowfish und Twofish, Mitglied in mehreren Verbänden) in seinem offiziellen Blog als plausibel bestätigt: "Someone somewhere commented that the NSA's "groundbreaking cryptanalytic capabilities" could include a practical attack on RC4. I don't know one way or the other, but that's a good speculation." Darüber hinaus warnen verschiedene Seiten, wie Golem und Heise, die sich mit Informatik beschäftigen vor der Verwendung von RC4. Selbst das Bundesamt für Sicherheit in der Informationstechnik (BSI) schreibt in einer technischen Richtlinie für Kryptographische Verfahren Anfang 2014: "Der Verschlüsselungsalgorithmus RC4 in TLS weist erhebliche Sicherheitsschwächen auf und darf nicht mehr eingesetzt werden." Das BSI gibt zusätzlich an, dass das Verschlüsselungsverfahren AES verwendet werden soll. Durch diese gezeigten Publikationen wird das Verfahren RC4 als unsicher angesehen und in dieser Arbeit nicht verwendet.

3.1.5 PBE

3.1.6 Blowfish

3.2 Asymetrische Verfahren

3.2.1 RSA

3.3 Hash-Funktionen

3.4 Digitale Signature

3.4.1 Public Key Infrastruktur

3.5 Schlüsselvereinbarung

3.5.1 Diffie Hellmann

3.5.2 ElGamal

3.6 Schlüsselgenerierung

3.7 Authentifizierung

3.7.1 Zwei-Faktor-Authentifizierung

4 Validierung

4.1 Verschlüsselungsverfahren

4.2 Hashfunktionen

5 Implementierung

5.1 Entwurf

6 Test

6.1 Validierung

6.2 Testverfahren

7 Zusammenfassung und Ausblick

7.1 Zusammenfassung

7.2 Ausblick

Index