fhwedel

UNIVERSITY OF APPLIED SCIENCES

Seminararbeit

in der Fachrichtung
Wirtschaftsingenieurwesen

Symmetrische Verschlüsselung: Blockchiffre, Stromchiffre, Modi: ECB, CBC, CFB, Aktuelle Chiffren: Rijendael, AES Ausschreibung, Kryptoanalyse, Schlüssellängen und „Adversary Advantage“

Erstellt von:	Moritz Gadow Mat-Nr.: 101705 E-Mail: wing101705@fh-wedel.de
Erarbeitet im:	6. Semester
Abgegeben am:	28.05.2018
Betreuender Dozent:	Prof. Dr. Michael Anders Fachhochschule Wedel

Inhaltsverzeichnis

Einleitung 1
Grundprinzipien der symmetrischen Verschlüsselung 2
Substitutionschiffrierung 2
Transpositionschiffrierung/Permutation 3
Stromchiffre 4
Synchrone Stromchiffrierung 5
Selbstsynchronisierende Stromchiffrierung 5
Blockchiffre 6
Padding 6
Betriebs-Modi 6
Electronic Code Book Mode 7
Cipher Block Chaining Mode 7
Cipher Feedback Mode 8
Data Encryption Standard 8
Advanced Encryption Standard 9
Kryptoanalyse 11
Kryptoanalyse-Methoden 12
Häufigkeitsanalyse 12
Brute-Force-Angriff 12
Seitenkanal-Angriff 13
Lineare Kryptoanalyse 13
Differentielle Kryptoanalyse 13
Adversary Advantage 14
Sicherheit 15

Einleitung

Wirft man einen Blick auf die Historie der Kryptologie, so ist die symmetrische Verschlüsselung die älteste und bis in die siebziger Jahre auch die einzige Verschlüsselungsart auf die Menschen zurückgegriffen haben. Schon Caesar soll diese Art der Verschlüsselung genutzt haben, um wichtige Botschaften geheim übermitteln zu können. Dazu verschob er jeden Buchstaben im Alphabet um drei Stellen[1]. Somit ergab der reine Text für den Leser keinen Sinn, es sei denn er wusste, dass jeder Buchstabe um drei Stellen im Alphabet zurückverschoben werden musste.

Die symmetrische Verschlüsselung ist eine, in ihren Grundzügen einfache Verschlüsselungsmethode, da die Ver- und Entschlüsselung mit demselben Schlüssel erfolgt. Somit muss der Empfänger den selben Schlüssel haben wie der Absender. Daher ist es bei diesem Verfahren wichtig, dass der Schlüssel sicher übermittelt wird. Früher geschah dies meist physisch, d.h. es wurde eine Bote von dem Absender zu dem Empfänger gesandt, der den Schlüssel überbringen sollte. Genau dieses Element der Verschlüsselung stellt die Schwachstelle der symmetrischen Kryptosysteme dar: gerät der Schlüssel während der Übermittlung in falsche Hände, kann die Nachricht ohne weiteren Aufwand entschlüsselt werden.

Generell lässt sich die symmetrische Verschlüsselung in zwei Untergruppen einteilen: Stromchiffren und Blockchiffren. Bei Stromchiffren wird jedes Zeichen des Texts einzeln verschlüsselt und nach demselben Verfahren entschlüsselt. Eine andere Methode ist das Blockchiffre-Verfahren: hierbei werden einzelne Zeichen zu Blöcken zusammengefasst und so ver- und entschlüsselt. Des Weiteren spielt bei Bockchiffren der Betriebsmodus eine entscheidende Rolle, da erst über diesen beliebig lange Texte verschlüsselt werden können.

[1] https://de.wikipedia.org/wiki/Caesar-Verschlüsselung

Grundprinzipien der symmetrischen Verschlüsselung

In den frühen Jahren der Verschlüsselung wurde bei der Art der Verschlüsselung auf einfaches Vertauschen oder Versetzen von Buchstaben gesetzt. Sichere Algorithmen beinhalteten damals beide Methoden, die mehrmals nacheinander angewandt wurden. Mit der Zeit des Computers wurde diese Verfahren etwas komplizierter, blieben aber vom Prinzip dieselben. Es wurde nicht mehr Buchstaben vertauscht, sondern Bits.
In der kryptographischen Fachsprache wird das Ersetzen von Zeichen oder Bits Substitution genannt, die Methode des Vertauschens wird als Transposition oder Permutation bezeichnet. Auf diese beiden Varianten möchte ich in den folgenden Absätzen weiter eingehen. Ich werde mich im weiteren Text auf Bits oder Zeichen beziehen, all die Verfahren sind simultan auf das jeweils anderen anzuwenden.

Substitutionschiffrierung

Übersetzt man das englische Wort „Substitution“, so wäre es im deutschen gleichbedeutend mit „Austauschen“. Genau nach diesem Prinzip arbeiten Substitutionschiffren: es wird ein Bit durch ein bestimmtes oder mehrere bestimmte Bits ausgetauscht. Hierbei kann in vier verschieden Varianten der Substitutionsverschlüsselung aufgeteilt werden:

1. Die monoalphabetische Chiffrierung ist die einfachste Art der Substitution: es wird jedem Buchstaben eines Alphabets ein anderer Buchstabe zugewiesen. Somit wird jeder Buchstabe eines Klartexts durch den dazugehörigen Buchstaben ersetzt und bildet somit den Chiffrentext. Ein Beispiel für dieses Verfahren wäre die Caesar-Chiffre, bei der jeder Buchstabe im Alphabet um jeweils drei Stellen verschoben wird. Dementsprechend wird aus einem „A“ ein „D“ und aus einem „R“ ein „U“. Dieses Verfahren ist jedoch anfällig für die Häufigkeitsanalyse (siehe Kryptoanalyse Methoden) und gilt somit als nicht sicher.
2. Die homophone Substitutions-Chiffrierung ähnelt der monoalphabetischen. Anstatt, dass jedem Buchstaben ein bestimmter Buchstabe zugewiesen wird, gibt es in diesem Fall mehrere zugehörige andere Zeichen, so kann ein der Buchstabe „E“ z.B. durch die Zeichen „4“, „?“ und „93“ ausgedrückt werden. Somit ist die homophone Substitutions-Chiffrierung gegen die einfache Häufigkeitsanalyse nicht so unsicher wie die monoalphabetische Chiffrierung.

3. Bei der polygraphischen Substitution-Chiffrierung werden Buchstabenblöcke verschlüsselt. Auch hier entsprechen einem Buchstabenblock mehrere Chiffrenblöcke. D.h. der Block „VER“ kann im Chiffrentext durch „KDJ“, „ELF“ oder „MDA“ ausgedrückt werden.
4. Die letzte Methode ist die polyalphabetische Substitution-Chiffrierung, diese entsteht aus einer Mehrfachanwendung der monoalphabetischen Chiffrierung. Wie häufig diese Verschlüsselung auf einen Buchstaben angewendet wird ist von der Stelle im Klartext abhängig. Beispielsweise kann übermittelt werden, dass die erste Stelle des Klartexts drei monoalphabetische Chiffrenschritte verschlüsselt werden soll. So wird aus dem ersten Buchstaben „D“ in der ersten Runde ein „R“, in der zweiten Runde wird aus dem „R“ ein „%“ und in der letzten Runde aus dem „%“ eine „0“. Bei dem zweiten Buchstaben könnten dann z.B. 7 Runden durchlaufen werden. [2]

Transpositionschiffrierung/Permutation

Gegenüber zur Substitutionschiffrierung bleiben bei der Transpositionschiffrierung alle Buchstaben aus dem Klartext in dem Chiffrat erhalten, was sich jedoch ändert sind die Positionen der einzelnen Zeichen. Die einfachste Abwandlung ist die einfache Transpositionschiffrierung.

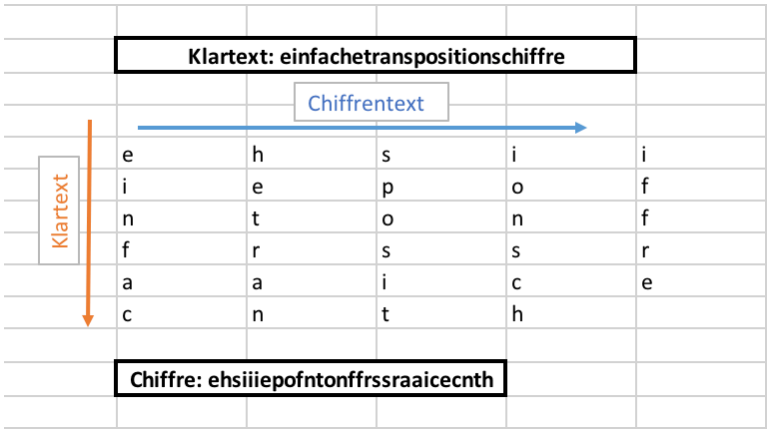

Abbildung 1: einfache Transpositionschiffre

Wie in Abbildung 1 zu erkennen ist, wird der Klartext Zeichen für Zeichen vertikal in eine Tabelle übernommen. Um den Chiffrentext zu erhalten, ist die Tabelle spaltenweise auszulesen. Für den Empfänger ist das Chiffrat zu entschlüsseln, wenn er weiß, wie viele

[2] Bruce Schneier, „Angewandte Kryptographie – Protokolle, Algorithmen und Sourcecode in C“, 1. korrigierter Nachdruck 1997, S.11-12

Zeilen die Tabelle hat. Die Anzahl der Zeilen ist somit der Schlüssel. Dieses Verfahren ist heutzutage recht einfach zu „knacken“ und dient nur zur Veranschaulichung des Prinzips. Permutationen die Computer nutzen sind weitaus komplizierter und daher auch sicherer.[3]

Stromchiffre

Eine Stromchiffre ist eine Art der Chiffrierung, bei der der Klartext Zeichen für Zeichen in den Chiffrentext übersetzt wird. Hierfür wird vorausgesetzt, dass ein Schlüsselstrom vorhanden ist, der mit dem Klartext zusammen ein Chiffrat erzeugen kann. Dies geschieht durch eine XOR-Verknüpfung des Klartextes mit dem Schlüsselstrom:

$$C_i = S_i \oplus M_i$$

C_i=Chiffrentext (ciphertext)
S_i=Schlüsselstrom (key stream)
M_i=Klartext (message)

Praktisch funktioniert dieses Verfahren so, dass die i-te Stelle des Klartexts mit der i-ten Stelle des Schlüsselstroms verknüpft wird, daraus ergibt sich die i-te Stelle Chiffrentexts.
Die Sicherheit des Chiffrats beruht auf der Zufälligkeit des Schlüsselstroms: sieht der Schlüsselstrom aus wie einfache Zufallsbits, so sieht auch das Chiffrat nach reinen Zufallsbits aus. Je zufälliger der Schlüsselstromgenerator den Schlüsselstrom erzeugt, desto sicherer wird das Chiffrat sein. Der Schlüsselstromgenerator ist ein Algorithmus, der durch die Eingabe des Schlüssels den Schlüsselstrom generiert. Bei der Erstellung des Schlüsselstroms ist darauf zu achten, dass der Schlüsselstrom immer mindestens so lang sein sollte wie der Klartext, da sich der Schlüssel sonst ab einer bestimmten Stellen wiederholen und somit eine Angriffsfläche für Attacken bilden würde.
Nachdem das Chiffrat erfolgreich verschlüsselt und verschickt wurde, kann der Empfänger mit Hilfe des Schlüssels und des Schlüsselstromgenerators denselben Schlüsselstrom wie der

[3] Bruce Schneier, „Angewandte Kryptographie – Protokolle, Algorithmen und Sourcecode in C“, 1. korrigierter Nachdruck 1997, S.13-14

Absender erzeugen und durch eine XOR-Verknüpfung des Chiffrats mit dem Schlüsselstrom die Nachricht dechiffrieren.[4]

Stromchiffren lassen sich in zwei Untergruppen aufteilen, einerseits gibt es synchrone Stromchiffrierung und andererseits selbstsynchronisierende Stromchiffrierung.

Synchrone Stromchiffrierung

Betrachtet man die synchrone Stromchiffrierung, so wird der Schlüsselstrom unabhängig von Vom Klartextstrom generiert, die einzigen Faktoren wovon er abhängt ist der Schlüssel und der Schlüsselstromgenerator. Vorteilhaft an diesem Prinzip ist, dass der Schlüsselstrom vor dem eigentlichen Ablauf der Verschlüsselung erzeugt werden kann, somit ist mehr Rechnerleistung für den Algorithmus verfügbar. Auf der anderen Seite ist dieses Prinzip anfällig für eingefügte oder verlorene Bits im Klartext, weil der Schlüsselstrom und der Klartext nur über die Position zusammengefügt werden. Folglich wird das Chiffre aber der Stelle des eingefügten oder verlorenen Bits bis zum Ende der Nachricht falsch dechiffriert. Unproblematisch ist dagegen die falsche Übermittlung eines Bits: wird ein Bit falsch übermittelt und mit dem Schlüsselstrom dechiffriert ist lediglich das falsch übermittelte Bit falsch dechiffriert.[5]

Selbstsynchronisierende Stromchiffrierung

Im Gegensatz zur synchronen hängt die selbstsynchronisierende Stromchiffrierung von einer bestimmten Anzahl n vorangegangener Chiffrenstrombits ab. Das bedeutet, dass der Klartext an der Stelle i nicht nur mit dem Schlüsselstrom an der Stelle i verknüpft wird, sondern auch mit dem Chiffrenstrom der Stellen i-1, i-2 und i-3. Daraus folgt, dass verlorenen oder eingefügte Bits nur eine falsche Dechiffrierung von n Bits nach sich zieht. Danach hat sich die Chiffrierung wieder synchronisiert.[6]

[4]Bruce Schneier, „Angewandte Kryptographie – Protokolle, Algorithmen und Sourcecode in C“, 1. korrigierter Nachdruck 1997, S.232-233
[5] Vgl. http://www.koram.de/doks/sa/node17.html
[6] Vgl. http://www.koram.de/doks/sa/node17.html

Blockchiffre

In der heutigen Zeit werden in der symmetrischen Verschlüsselung meistens Blockchiffre verwendet. Dies hat den Grund, dass diese variabler sind und ich unterschiedlichen Modi (siehe Betriebs-Modi) genutzt werden können. Strom- und Blockchiffre unterscheiden sich darin, dass bei der Blockchiffrierung die Bits nicht mehr einzeln nacheinander, sondern blockweise verschlüsselt werden. Ein Block besteht aus mehreren Bits (häufig 64 oder 128).[7]

Padding

Ein Problem, dass bei der Blockchiffrierung auftreten kann, ist, dass die letzten Bits des Klartexts keinen kompletten Block füllen können und somit theoretisch nicht chiffriert werden können. In diesem Fall werden die restlichen freien Stellen durch sogenanntes „Padding" aufgefüllt. Dies bringt zudem den Vorteil mit sich, dass die Länge des Chiffrats keine Auskunft über die Länge des Klartexts gibt.
Es gibt zwei Abwandlungen des Paddings:

1. Es werden alle freien Bits im letzten Block durch Nullen aufgefüllt und auf die letzte Stelle wir die Anzahl er angehängten Nullen geschrieben, sodass erkennbar ist, wo die Nachricht aufhört.
2. Es wird ein vereinbartes Trennzeichen hinter das letzte Bit der Nachricht gesetzt und alle anderen Zeichen mit Nullen aufgefüllt.[8]

Betriebs-Modi

Der Betriebs-Modus bei Blockchiffren kann nach unterschiedlichen Kriterien gewählt werden. Für einen Anwendungsbereich kann es von Vorteil sein, dass die Verschlüsselung möglichst schnell abläuft, für andere liegt die Priorität auf dem Verhindern vom Einfügen oder Löschen von Bits. Nachfolgend möchte ich drei unterschiedliche Betriebs-Modi vorstellen.

[7] Wolfgang Ertel, „Angewandte Kryptographie", 4.,überarbeitete und ergänzte Auflage 2012, S.59/S.74
[8] Bruce Schneier, „Angewandte Kryptographie – Protokolle, Algorithmen und Sourcecode in C", 1. korrigierter Nachdruck 1997, S.223-224

Electronic Code Book Mode

Der Electronic Code Book Mode (ECB) ist der einfachste Betriebs-Modus der Blockchiffrierung. In diesem Modus werden die Blöcke unabhängig voneinander verschlüsselt. Dies bringt die vorteilhafte Eigenschaft mit sich, dass Fehler die beim Übertragen in einem Block entstehen, sich nicht auf spätere Blöcke auswirken. Allerdings werden gleiche Klartextblöcke jedes Mal wieder gleich verschlüsselt, sprich die Chiffrate der Blöcke sind identisch. So können in einem Chiffrentext Schlüsselwörter schneller erkannt und analysiert werden. Zudem ist der ECB nicht sicher gegen „Replay"-Angriffe.[9]

Cipher Block Chaining Mode

Der Cipher Block Chaining Mode (CBC) unterscheidet sich vom ECB darin, dass die Chiffrenblöcke untereinander verknüpft sind. In der Praxis wird dies so umgesetzt, dass die Klartextblöcke mit dem vorherigen Chiffrentextblock XOR-verknüpft werden. Aus diesem Grund verteilen sich die statistischen Eigenschaften über die komplette Länge des Chiffrats und die Verschlüsselung ist somit sicherer gegen die Kryptoanalyse. Bei dem CBC-Mode ist zu beachten, dass beim Entschlüsseln die Struktur und Reihenfolge der Blöcke gleich bleibt, da sonst die Beziehungen der Blöcke durcheinander gebracht werden und sich dadurch Fehler durch die komplette Chiffre ziehen können.
Ein weiterer Aspekt, den man beachten sollte, ist der Initialisierungsvektor. Wenn man z.B. eine E-Mail verschlüsseln möchte und man immer mit dem selben Schlüssel verschlüsselt, würden die Anfänger der Chiffren gleich aussehen. Denn die Absender-Zeile fängt immer gleich an. Damit der Angreifer keine Rückschlüsse auf die Art der Nachricht ziehen kann, sollte man immer einen Initialisierungsvektor nutzen. Der Initialisierungsvektor ist ein Block an Zufallsvariablen, den man vor den ersten Block der Nachricht stellt. Das sich im CBC-Mode die Blöcke auf einander beziehen, wird der Block der nach dem Initialisierungsvektor folgt, verändert und sieht somit jedes Mal anders aus.[10]
Trotzdem ist dieser Modus nicht sicher gegen das Anhängen von Blöcken. Diese würde den Inhalt der eigentlichen Nachricht zwar nicht verändern, können aber in manchen Situationen

[9]Vgl. https://itsecblog.de/verschluesselungsmodus-im-detail-empfehlung/
[10] Bruce Schneier, „Angewandte Kryptographie – Protokolle, Algorithmen und Sourcecode in C", 1. korrigierter Nachdruck 1997, S.229

störend sein, deshalb ist es sinnvoll das Ende einer Nachricht für den Empfänger immer deutlich erkennbar zu machen.[11]

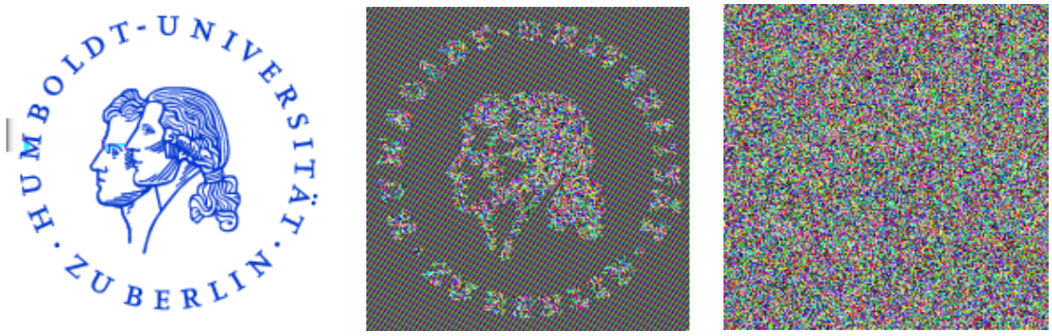

Abbildung 2: grafischer Unterschied zwischen ECB-Mode und CBC-Mode; Bild1: unverschlüsselt, Bild2: ECB-verschlüsselt, Bild3: CBC-verschlüsselt

Quelle: https://www.mathematik.hu-berlin.de/~wilhelm/openssl-notizen.pdf

Cipher Feedback Mode

Als letztes Beispiel möchte ich den Cipher Feedback Mode (CFB) vorstellen. Dieser Modus ist eine Kombination aus Strom- und Blockchiffrierung, bei dem innerhalb der Blöcke stromchiffriert wird. Dies kann manchmal wichtig sein, z.B. wenn Bits direkt verschlüsselt werden sollen und nicht erst, wenn ein kompletter Block gefüllt ist, demzufolge wäre der CBC-Mode ungeeignet. Im CFB-Mode können einzelne Bits des Klartextes mit Bits des Schlüsselstroms verknüpft werden. Auch in diesem Modus werden die Blöcke miteinander verknüpft, damit er sicher gegen das Einfügen oder Entfernen von Blöcken ist.[12]

Data Encryption Standard

Der Data Encryption Standard (DES) ist ein 1977 vorgestellter Algorithmus, der von dem „National Bureau of Standards“ und IBM veröffentlicht und mit Hilfe der NSA entwickelt wurde. Er diente als Verschlüsselungs-Standard für US-Behörden und andere Instanzen, bis er als unsicher eingestuft wurde. Entwickelt wurde der DES als Blockchiffre mit 64-Bit-

[11] Vgl. https://www.itwissen.info/Schluesselblock-Kette-cipher-block-chain-CBC.html
[12] Bruce Schneier, „Angewandte Kryptographie – Protokolle, Algorithmen und Sourcecode in C“, 1. korrigierter Nachdruck 1997, S.235

Blöcken und einem 56-Bit-Schlüssel, der für heutige Verhältnisse zu kurz und damit zu unsicher ist. Zur Zeit der Veröffentlichung des Standards, wurde ein 56-Bit-Schlüssel als ausreichend sicher angesehen, jedoch war die Zeit, in der diese Schlüssellänge geknackt werden könne, absehbar. Infolgedessen kam Kritik auf und es wurde vermutet, dass die NSA schon damals in der Lage gewesen sein soll, einen 56-Bit-Schlüssel knacken zu können und sich deshalb für eine Schlüssellänge von 56 Bit eingesetzt haben soll. Zudem wurden die in dem Algorithmus verwendeten S-Boxen von Analytikern untersucht und es wurden nicht erklärbare Strukturen entdeckt, die scheinen den Algorithmus eher zu schwächen als ihn zu stärken. Deshalb wurde auch hier vermutet, dass sich die NSA „Hintertüren“ in die S-Boxen gebaut haben könnte.[13]

Advanced Encryption Standard

1997 wurde vom “National Institute of Standards and Technology” (NIST), der alten „National Bureau of Standrads“, eine Ausschreibung veröffentlicht, die einen Nachfolger für den veralteten DES finden sollte. Das NIST stellte selbst keinen Bewerber, da es als neutraler Gutachter dienen wollte. Nachdem viel Kritik an dem DES aufgekommen war, wollte die amerikanischen Behörden das Auswahlverfahren und auch den Algorithmus an sich durchsichtiger gestalten. Die Ausschreibung galt weltweit und stellte folgende Bedingungen:

- Er sollte eine symmetrische Blockchiffrierung sein.
- Die Blocklänge sollte 128 Bit betragen.
- Er sollte mit Schlüsseln der Längen 128 Bit, 192 Bit und 256 Bit gearbeitet werden können.
- Zudem sollte der Algorithmus leicht in Hard- und Software implementierbar sein und
- lizenzfrei und unentgeltlich genutzt werden können.
- Er sollte „unknackbar“ gegenüber bekannten Kryptoanalyse-Methoden, besonders gegen Power- und Timing-Attacken.

[13] https://de.wikipedia.org/wiki/Data_Encryption_Standard

- Für den Einsatz in Chipkarten sollte der Code möglichst kurz sein und wenig Speicherplatz in Anspruch nehmen.

Bewertet werden sollte die Ausschreibung nach drei Hauptkriterien:

1. Sicherheit
2. Kosten
3. Algorithmus- und Implementierungseigenschaften

Am 20. August 1998 fand die „First AES Candidate Conference“ statt und es wurde 15 Algorithmen aus aller Welt vorgestellt. Anschließend sollte diese in Fachkreisen, in Bezug auf die gestellten Bedingungen diskutiert und analysiert werden. Ein Jahr später im März 1999 wurde in Ventura, Kalifornien die „Second AES Candidate Conference“ abgehalten, auf der die bis dato erfolgten Analysen weiter diskutiert wurden. Zum Ende der Konferenz wurde fünf Bewerber ausgewählt, die in die letzte Runde einziehen sollten. Zu den Finalisten zählte der Algorithmus „MARS“, der von Mitarbeitern von IBM entwickelt worden war. Außerdem die beiden amerikanische Codes „RC6“ und „“Twofish“, sowie der „Serpent“, der von einem englischen, einem israelischen und einem dänischen Kryptographen entwickelt wurde. Der letzte Finalist war der belgische Algorithmus „Rijndael“ der von den beiden Wissenschaftlern Joan Daemen und Vincent Rijmen vorgestellt wurde. In der letzten Runde wurde die Codes weiter auf Implementierbarkeit, „Knackbarkeit“ gegenüber möglichen technischen Fortschritten in der Zukunft und anderen Kriterien geprüft.

Im April 2000 fand die „Third AES Candidate Conference” statt zur letzten Besprechung der Algorithmen. Im Oktober 2000 wurde der Gewinner bekanntgegeben: Es wurde der belgische Algorithmus „Rijendael“. Er schlug die Konkurrenz unter anderem in den Kategorien der schnellen Implementierbarkeit in Hard- und Software, in der Kürze des Codes und bestach durch eine einfache und elegante mathematische Theorie.[14,15]

[14] Vgl. https://secorvo.de/publikationen/advanced-encryption-standard-fox-1999.pdf

[15] Vgl. https://csrc.nist.gov/projects/cryptographic-standards-and-guidelines/archived-crypto-projects/aes-development

Kryptoanalyse

Das Gegenstück zur Kryptographie ist die Kryptoanalyse. Sie befasst sich mit der Analyse der Kryptographie und damit mit dem Umgehen von kryptographischen Systemen und dem Aushebeln der Schutzfunktionen. Sie soll Schwachstellen in Algorithmen aufzeigen um diese zu verbessern, oder beim Scheitern der Kryptoanalyse nachweise, dass das System sicher ist. Nach Kerckhoffs‘ Prinzip soll die Sicherheit eines kryptographischen Systems nicht auf der Geheimhaltung des Algorithmus beruhen, sondern auf der Geheimhaltung der Schlüssels.[16] Daher zielen die meisten Angriffe auch darauf ab, an den Schlüssel zu gelangen und mit diesem ein Chiffrat entschlüsseln zu können. Bei den meisten kryptographischen Systemen ist der im Hintergrund arbeitende Algorithmus offengelegt und somit eine Angriff auf diesen sinnlos.

In der Kryptoanalyse gibt es vier grundlegende Angriffe, die ich auf dieser Seite weiter erläutern möchte.

1. Der am wenigsten effektive Angriff ist ein Ciphertext-Only-Angriff. Hierbei kennt der Angreifer keinen Klartext, sondern nur den Chiffrentext und versucht über Probieren den richtigen Schlüssel herauszufinden.
2. Etwas effektiver ist der Known-Plaintext-Angriff, hierbei sind dem Angreifer ein Klartext oder Teile davon bekannt und der dazugehörige Chiffrentext. Folglich hat der Angreifer mehr Angriffsfläche auf das System.
3. Bei dem Chosen-Plaintext-Angriff kann der Angreifer eine Klartext frei wählen und erhält nach der Verschlüsselung den dazugehörigen Chiffrentext. Auf Grund dieser Voraussetzung kann er versuchen den Schlüssel zu berechnen.
4. Der effektivste Angriff ist der Chosen-Ciphertext-Angriff. Dem Angreifer stehen mehrere Chiffrentexte zur Verfügung, zu welchen er den dazugehörigen Klartext generieren kann.

Die Sicherheit von kryptographischen Systemen kann in Gruppen gegliedert werden. Ein System gilt als uneingeschränkt sicher, wenn unabhängig von der Menge des vorhandenen Chiffrentexts der Klartext nicht eindeutig identifiziert werden kann. Das einzige System, das als uneingeschränkt sicher gilt, ist das One-Time-Pad. Dieses findet in der Realität jedoch kaum Anwendung, da die Verwendung zu aufwändig ist. Als berechnungssicher gilt ein

[16] Vgl. https://de.wikipedia.org/wiki/Kerckhoffs’_Prinzip

System dann, wenn es nicht unter vernünftigem Zeitaufwand und mit vorhandenen Computern gebrochen werden kann.[17]

Kryptoanalyse-Methoden

In der Kryptoanalyse gibt es viele unterschiedliche Ansätze, wie ein System gebrochen werden kann. Im Folgenden möchte ich ein paar Kryptoanalyse-Methoden vorstellen. Wobei zu beachten ist, dass die Häufigkeitsanalyse heutzutage keine wirklich effektive Analysemöglichkeit darstellt, sie die zur Verdeutlichung des Prinzips und zeigt, wie Kryptoanalyse in ihrer Anfangszeit funktioniert hat.

Häufigkeitsanalyse

Die Häufigkeitsanalyse benutzt die statistischen Eigenschaften einer Sprache um einen Chiffrentext knacken zu können. In jeder Sprache treten Buchstaben mit einer bestimmten Wahrscheinlichkeit auf. Sind diese Häufigkeiten bekannt, können darüber Rückschlüsse auf den Klartext geschlossen werden. In der deutschen Sprache tritt z.B. der Buchstabe „E“ mit 17,4% am häufigsten auf. Analysiert man einen Chiffrentext und notiert die Häufigkeiten jedes einzelnen Buchstaben, so sollte jeder Wahrscheinlichkeit eines Buchstaben aus dem Chiffrentext einer Wahrscheinlichkeit eines Buchstaben des Alphabets zugewiesen werden können. Diese Art der Kryptoanalyse kann unter anderem auf monoalphabetische Chiffrierung angewandt werden.[18]

Brute-Force-Angriff

Die trivialste Herangehensweise ist wohl ein Brute-Force-Angriff, was so viel bedeutet, wie ein Angriff mit brachialer Gewalt. Das Prinzip besteht darin alle möglichen Schlüssel auszuprobieren. Hierfür werden teilweise auf die Verfahren angepasste Computer entwickelt, mit einer großen Anzahl an Kryptochips, die den Schlüssel in möglichst kurzer Zeit knacken sollen.[19]

[17] Vgl. https://www.kryptowissen.de/kryptoanalyse.html
[18] Vgl. https://www.symbcomp.fim.uni-passau.de/fileadmin/files/lehrstuhl/kreuzer/Artikel/kryptographie-LFB.pdf
[19] https://www.datenschutz.org/brute-force/

Seitenkanal-Angriff

Ein Seitenkanal-Angriff greift weder direkt den Schlüssel, noch den Algorithmus an. Bei dieser Art des Angriffs geht es eher darum ein kryptographisches Gerät während des Verschlüsselungsprozesses zu beobachten, um daraus Informationen gewinnen zu können. Dazu zählen z.B. Timing-Attacken, über die man Rückschlüsse auf den Algorithmus ziehen kann, anhand der Dauer, die das Gerät zum Chiffrieren benötigt. Ein weiterer Seitenkanal-Angriff wäre eine Power-Attacke, wobei der Stromverbrauch während des Prozesses aufgezeichnet wird. Experten können auch über diese Daten Auskunft über den Algorithmus geben. Es gibt noch viele andere Seitenkanal-Angriffe, die sich u.a. auf die Speichernutzung oder Schallwellen beziehen, auf die ich nicht weiter eingehen möchte.[20]

Lineare Kryptoanalyse

Das Prinzip, auf dem die lineare Kryptoanalyse basiert, ist eine lineare Annäherung an die Funktion einer Blockchiffrierung. Ich werde das Grundprinzip dieses Ansatzes erklären, da die Details recht kompliziert sind. Vorausgesetzt wird, dass Angreifer mindestens einen Known-Plaintext-Angriff durchführen kann. Als erstes werden einige Bits des Klartextes miteinander XOR-verknüpft, das selbe soll mit ein paar Bits des Chiffrentexts durchgeführt werden. Im nächsten Schritt sollen diese beiden Teile dann auch XOR-operiert werden. Es wird nun davon ausgegangen, dass diese Bits zu einer bestimmten Wahrscheinlichkeit einigen XOR-verknüpften Bits des Schlüssels entsprechen. Ist diese Wahrscheinlichkeit asymmetrisch verteilt, sprich nicht 0,5, so lassen sich anhand der Wahrscheinlichkeit Schlüsse ziehen, wie der Schlüssel aussehen könnte. Hierbei gilt: Je mehr Klar- und Chiffrentext verarbeitet werden kann, desto näher kann sich der approximierte Schlüssel an den tatsächlichen Schlüssel annähern.[21]

Differentielle Kryptoanalyse

Die letzte Methode, die ich in der Kryptoanalyse vorstellen möchte, ist die differentielle Kryptoanalyse. Diese betrachtet bestimmte paare von Chiffrentexten, deren Klartexte bestimmte Differenzen aufweisen. Im Prinzip werden ganze Teile eines Algorithmus, mit

[20] Vgl. https://de.wikipedia.org/wiki/Seitenkanalattacke
[21] Bruce Schneier, „Angewandte Kryptographie – Protokolle, Algorithmen und Sourcecode in C“, 1. korrigierter Nachdruck 1997, S.338-340

unterschiedlichen Eingaben durch eine XOR-Operation miteinander verknüpft. Da somit auch der Schlüssel XOR-verknüpft wird, „löst“ sich dieser auf. Darüber können nun z.B. im DES Differenztabellen für alle S-Boxen aufgestellt, werden. Es werden also die Bits vor und nach der S-Box betrachtet und ihre Differenzen vermerkt. Mit Hilfe dieser Differenztabellen können möglichen Schlüsseln Wahrscheinlichkeiten zugewiesen werden. Auch hier gilt das selbe wie für die lineare Kryptoanalyse: Je mehr Daten verarbeitet werden können, desto höher ist die Wahrscheinlichkeit den richtigen Schlüssel herauszufinden.[22]

Adversary Advantage

Dieser Ansatz beschreibt den Optimalzustand einer Chiffre, der gegeben ist, wenn der Angreifer keinen Unterschied ausmachen kann zwischen einer Chiffre und einem Block aus Zufallsbits. Es wird sich also die Frage gestellt: Mit welcher Wahrscheinlichkeit erkennt der Angreifer das Chiffrat, bei einer Auswahl von einem Chiffrat und einem Text Zufallsdaten? Wenn der Angreifer keinen Unterschied erkennen kann, sich also mit einer Wahrscheinlichkeit von 0,5 für das Chiffrat entscheiden würde, wird es „Zero Adversary Advantage“ genannt. Dies kommt daher, dass die Differenz zwischen den Wahrscheinlichkeiten Null beträgt. Der andere Fall wäre der „Negigible Adversary Advantage“, also ein vernachlässigbarer Vorteil für den Angreifer. Dafür probiert der Angreifer Schlüssel aus und schaut sich die daraus entschlüsselten Daten an. Wenn er sagen kann, dass ein Schlüssel, in Kombination mit dem entschlüsselten Text wahrscheinlicher ist als ein anderer, verändert sich die Wahrscheinlichkeit, dass er das Chiffrat richtig wähle würde minimal. Problematisch ist, dass im Kopf eines Chiffrats häufig die Art des Algorithmus angegeben wird, daher wäre das Chiffrat sofort als solches zu erkennen. Es müsste also der Kopf des Chiffrats nochmals verschlüsselt werden, um dieses Problem zu beseitigen.[23,24]

[22] Bruce Schneier, „Angewandte Kryptographie – Protokolle, Algorithmen und Sourcecode in C“, 1. korrigierter Nachdruck 1997, S.332-337
[23] Vgl. https://cs.nyu.edu/courses/spring04/V22.0480-003/handout1.pdf
[24] Vgl. https://www.fh-wedel.de/~an/crypto/accessories/The_Nada_Cap_5.pdf

Sicherheit

Die Sicherheit eines Kryptosystems sollte immer von der Geheimhaltung des Schlüssels Abhängen (Kerckhoffs' Prinzip), also somit indirekt auch von der Länge des Schlüssels. Geht man davon aus, dass ein Algorithmus perfekt ist, so wäre der bestmögliche Angriff eine Brute-Force-Attacke. Mit einem 8-Bit-Schlüssel gäbe es 2^8=256 möglich Schlüssel, dieser wäre folglich nicht sicher, 256 Schlüssel könnte man, wenn man wirklich wollte, auch noch per Hand ausprobieren. Betrachten wir nun einen 56-Bit-Schlüssel, wie z.B. den des DES. Hier gäbe es $2^{56}\approx$70 Billarden mögliche Schlüssel. Diese Zahl erscheint im ersten Moment sehr hoch, bezieht man jedoch in Betracht, dass ein normaler Computer 170 Millionen Passwörter pro Sekunde testen kann, erscheint auch dieser Schlüssel nicht wirklich sicher. Genau aus diesem Grund wurde der DES als Standard schon in den neunziger Jahren von dem AES abgelöst. Baut man spezielle Rechner, die darauf ausgelegt sind einen Schlüssel eines bestimmten Verfahrens zu knacken sind noch viel höhere Leistungen möglich. Im Jahr 1998 wurde ein Rechner mit über 1000 Kryptochips entwickelt, der den DES in höchstens sieben Stunden knacken konnte und 88 Milliarden Schlüssel pro Sekunde testen konnte. Dieser Rechner hat damals ca. 250.000 USD gekostet. Somit wurde die These der US-Regierung widerlegt, dass der DES nur mit millionenschweren Investitionen geknackt werden könne.[25]

Ein weiterer Aspekt der in die Auswahl eine Schlüssels mit einfließen sollte, ist die Art der zu verschlüsselnden Daten. Die ENISA (European Union Agency for Network and Information Security) empfiehlt bei symmetrischen Schlüsseln für kurzfristige Daten, z.B. Transaktionen, eine Schlüssellänge von mindestens 80 Bit. Bei mittelfristig gespeicherten Daten wird ein Schlüssel mit einer Länge von 128 Bit empfohlen und bei langfristig gespeicherten Daten ein 256-Bit-Schlüssel. Hierfür empfehlen würde sich der AES, da dieser bei der Schlüssellänge variieren kann bis hin zu einem 256-Bit-Schlüssel.[26]

[25] Vgl. http://cryptography.wikia.com/wiki/EFF_DES_cracker

[26] European Union Agency for Network and Information Security, "Algorithms, key size and parameters report 2014", 2014

Symmetrische Verfahren	DES	RC4	3DES	AES	Camellia
Schlüssellänge (Bit)	56	128	168	128 / 192 / 256	128 / 192 / 256
Aufwand zum Brechen	2^{39}	2^{15}	2^{112}	$2^{126,1}$ / $2^{169,7}$ / $2^{254,4}$	2^{128} / 2^{192} / 2^{256}
Sicherheit	niedrig	extrem niedrig	mittel	hoch / sehr hoch / maximal	hoch / sehr hoch / maximal

Abbildung 3: Sicherheit symmetrischer Schlüssel, Quelle: https://www.elektronik-kompendium.de/sites/net/2006261.htm