Facharbeit im Grundkurs

Informatik

Untersuchung des Wired Equivalent Privacy-Verschlüsselungsprotokoll

Verfasser: Alexander Neuwirth

Kursleiter: Manuel Rosemann

Schuljahr 2014/15

Gymnasium Wolbeck

Inhaltsverzeichnis

Einleitung

Das Thema dieser Facharbeit ist die Untersuchung des bei „Wireless Local Area Network(s)“ (WLAN) verwendeten „Wired Equivalent Privacy“-Verschlüsselungsprotokoll (WEP). Mit Hilfe einer Computersimulation sollen Funktionsweise und Sicherheit des Verschlüsselungsprotokolls veranschaulicht werden.

Mein Interesse an kabellosen Netzwerken wuchs durch den täglichen Kontakt zu Computertechnologie mit WLAN-Schnittstellen, besonders aber durch mein Jugend-forscht-Projekt, in dem ein WLAN zur Kommunikation verwendet wird.

Das Ziel der Facharbeit ist eine Analyse der Sicherheit von WEP Verschlüsselungen, da Datensicherheit und der damit verbundene Schutz der Privatsphäre, auch aufgrund den NSA- und GCHQ-Skandalen in den letzten Jahren, in der öffentlichen Meinung an Relevanz zugenommen haben.

Geschichte

Wireless Local Area Network

Die erste kabellose Kommunikation zwischen Computern wurde 1971 von Norman Abramson einem Professor der Universität von Hawaii entwickelt. Die kabellosen Netzwerke wurden 1997, also 26 Jahre später, durch das „Institute of Electrical and Electronics Engineers“ zu IEEE 802.11 bzw. Wi-Fi standardisiert. Die heutzutage meist verwendeten Standards sind IEEE 802.11 a/b/g/n. Diese unterscheiden sich untereinander hauptsächlich in den verwendeten Frequenzen:

Standard	Frequenzband	Datum
802.11(ursprünglich)	2.5 GHz	1997
802.11b/g	2.5 GHz	1999/2003
802.11a	5 GHz	1999
802.11n:	2.5 GHz + 5 GHz	2009

Diese genannten IEEE-Normen schreiben bestimmte Frequenzen vor. Die Frequenzen sind 2.5 GHz bzw. 5 GHz, da diese in vielen Ländern ohne Lizenz nutzbar sind. Dies ermöglicht es Einzelpersonen eigene kabellose Netzwerke einzurichten, ohne sich um Lizenzierungen kümmern zu müssen.

Wired Equivalent Privacy

Das Verschlüsselungsprotokoll WEP wurde zeitgleich mit IEEE 802.11b/a eingeführt. Ziel des Protokolls ist es eine „verdrahteten (Systemen) entsprechende Privatsphäre“ (deutsche Übersetzung von WEP) zu gewährleisten. Aufgrund von Sicherheitslücken, auf welche im Folgenden genauer eingegangen wird, wurde WEP im Jahre 2003 durch „Wi-Fi Protected Access“ (WPA) abgelöst.

Funktionsweise der WEP-Verschlüsselung

Kontravalenz

Der Verschlüsselungsalgorithmus WEP basiert auf Kontravalenz (XOR). Der XOR-Operator berechnet aus zwei Bits ein resultierendes Bit:

A	B	$A \oplus B$
0	0	0
1	0	1
0	1	1
1	1	0

Einige Eigenschaften des exklusiven Oders sind:

1. $M \oplus 0 = M$
2. $M \oplus M = 0$
3. $(M \oplus K) \oplus K = M \oplus (K \oplus K) = M \oplus 0 = M$ (Assoziativgesetz)

Speziell letztere Eigenschaft bietet sich für einen Verschlüsselungsalgorithmus an, da die Nachricht (M), die mittels eines Schlüssel (K) verschlüsselt wird, in verschlüsselter Form ($M \oplus K$) übermittelt und anschließend wieder mit demselben Schlüssel (K) entschlüsselt werden kann.

Zyklische Redundanzprüfung

Die WEP Verschlüsselung verwendet einen CRC32-Algorithmus (engl. „cyclic redundancy check“) um sicher zu stellen, dass die Nachricht nicht durch äußere Einflüsse oder durch einen Angreifer verändert wurde. Beim CRC32-Algorithmus werden Nachrichtenteile (32-bit) durch ein Polynom (CRC-Polynom) dividiert. Dieses ist so gewählt, dass der Rest der Division die Prüfsumme ergibt. Diese reicht folglich von 0 bis zum Wert des Polynoms.
Der zu übertragenden Nachricht wird das Ergebnis des CRC32-Algorithmus angehängt. Nach dem die Nachricht verschlüsselt, übermittelt und entschlüsselt wurde, wird aus der Nachricht erneut eine Prüfsumme berechnet. Wenn beide CRC32-Algorithmus Ergebnisse gleich sind, wurde die Nachricht erfolgreich übermittelt.[1]

[1] Vgl. (CRC_Wikipedia)

River Cipher 4 (RC4)

Die Stromverschlüsselung (engl. „stream cipher") RC4 wird von WEP verwendet, um die Nachrichten mit Prüfsumme mittels der XOR-Operation zu verschlüsseln. RC4 benötigt einen Schlüssel, um Bits für die Verschlüsselung zu berechnen. Der Schlüssel muss dem Sender und dem Empfänger bekannt sein.

Der RC4 besteht aus zwei Algorithmen:

Key Scheduling Algorithm (KSA)

Der „Key Scheduling Algorithm" mischt ein zuvor geordnetes Array der Länge 255 (im Pseudocode S[] mit Werten von 0 bis 255) mittels des übermittelten Schlüssels (im Pseudocode K[]).

```
K[] = Array des Schlüssels
KL = Länge von K[]
S[] = Array der Länge 255

für i=0 bis 255
    S[i]=i
j=0
für i=0 bis 255
    j = (j + S[i] + K[i modulo KL]) modulo 256
    tausch S[i] und S[j]
```

Abbildung 1: KSA Pseudocode

Das gemischte Array (S[]) wird an den „Pseudo-Random Generation Algorithm" übergeben.

Pseudo-Random Generation Algorithm (PRGA)

Der „Pseudo-Random Generation Algorithm" erstellt aus dem übergebenen Array ein Array (im Pseudocode O[]) vorgegebener Länge (im Pseudocode ML), das verwendet wird, um die Nachricht (M[]) bitweise mit dem XOR-Operator zu verschlüsseln. [2]

```
S[] = gemischtes Array
ML = Länge der zu verschlüsselnden Nachricht (mit CRC32)
O[] = Ergebniss Array der Länge ML
i=0
j=0
für n=0 bis ML-1
    i = (i+1) modulo 256
    j = (j + S[i]) modulo 256
    tausch S[i] und S[j]
    t = (S[i]+S[j]) modulo 256
    O[n] = S[t]
```

Abbildung 2: PRGA (eingeschränkt) Pseudocode

[2] Vgl. (Erickson, 2008) (Lashkari, et al., 2009) (RC4_Wikipedia)

Initialization Vector

Beim WEP-Protokoll wird ein Initialization Vector (IV) verwendet. Dieser besteht aus 3 Bytes, also 24 Bits, die dem Schlüssel vorangesetzt werden, bevor dieser an den KSA übergeben wird. Der Algorithmus verschlüsselt eine Nachricht (M) in die Nachricht (C). Da der IV für jede Nachricht zufällig bestimmt wird, wird der Algorithmus wenn er die gleiche Nachricht (M) erneut verschlüsselt, mit hoher Wahrscheinlichkeit eine andere verschlüsselte Nachricht (C´) generieren.

Damit die Nachricht mit IV vom Empfänger entschlüsselt werden kann, benötigt dieser den IV. Deshalb wird der IV der verschlüsselten Nachricht vorangestellt, sodass er mit übermittelt wird.

Zusammenfassung

Verschlüsselung:

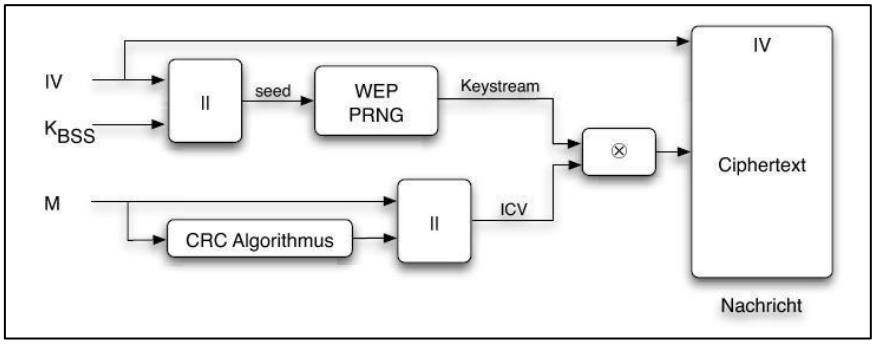

Abbildung 3: WEP Verschlüsselung[3]

1. An die zu verschlüsselnde Nachricht (M) wird das Ergebnis des CRC-Algorithmus angehängt (ICV).
2. An RC4 (WEP PRNG) wird als „seed“ IV und Schlüssel(K_{BSS}) übermittelt. Durch RC4 wird dann der „Keystream“ generiert.
3. Mittels der XOR-Operation von Keystream und ICV entsteht die kodierte Nachricht (Ciphertext).
4. Damit die kodierte Nachricht wieder dekodiert werden kann, wird ihr noch der IV vorangestellt.

Entschlüsselung:

[3] (WEP_Wikipedia)

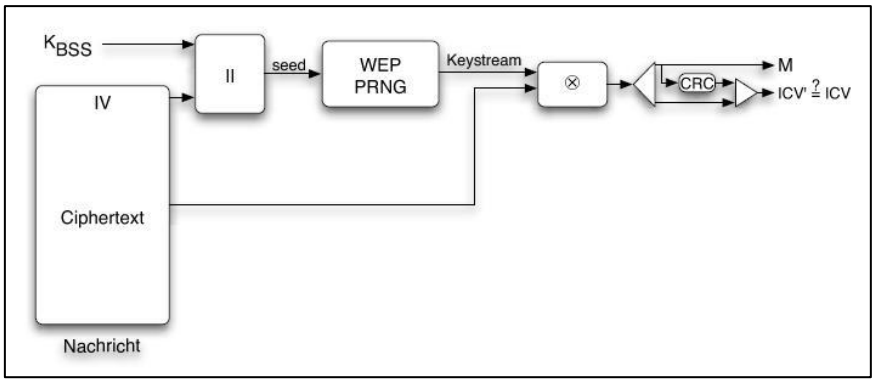

Abbildung 4: WEP Entschlüsselung[4]

1. Aus IV, aus der verschlüsselten Nachricht, und dem Schlüssel(K_{BSS}) wird der „seed“ wieder zusammengesetzt. Mittels des RC4 Algorithmuses (WEP PRNG) wird derselbe Keystream wie bei der Verschlüsselung erzeugt.
2. Durch das exklusive Oder erhält man aus Keystream und Ciphertext wieder die Originalnachricht mit der Prüfsumme.
3. Der CRC-Algorithmus erzeugt wieder mit der dekodierten Nachricht eine Prüfsumme. Wenn übermittelte und neu berechnete Prüfsumme übereinstimmen, ist die Nachricht erfolgreich übermittelt worden.

[4] (WEP_Wikipedia)

Sicherheitsproblematik

Im Folgenden werden die bekanntesten Sicherheitsprobleme im WEP Verschlüsselungsprotokoll erläutert.

1. Nachrichten erneut senden

Ein Angreifer kann beliebige gesendete Nachrichten aufzeichnen und erneut senden. Problematisch wäre zum Beispiel die Überweisung einer Geldsumme ohne weitere Sicherheitsmaßnahmen von einem Konto auf ein anderes.

Der Angreifer kann, ohne Kenntnis über den Inhalt der Nachricht, diese erneut senden und so die überwiesene Summe verdoppeln.

2. Gleicher Initialization Vector

Nachrichten mit gleichem IV haben folglich auch denselben Keystream. Wenn ein Angreifer zwei Nachrichten (M_1, M_2) mit gleichem IV mitgeschnitten hat und den Inhalt einer Nachricht (M_1) kennt, kann er mit dieser Formel auch den Inhalt der anderen Nachricht (M_2) erhalten:

$$(M_1 \oplus K) = C_1$$
$$(M_2 \oplus K) = C_2$$
$$(C_1 \oplus C_2) = (M_1 \oplus M_2) \oplus (K \oplus K) = (M_1 \oplus M_2)$$
$$(C_2 \oplus (C_1 \oplus M_1)) = M_2$$

In der Realität könnte ein Angreifer ein ihm bekanntes Protokoll (z.B. E-Mail: SMTP/IMAP) nutzen und eine eigene Nachricht (M_1) über das Internet in das Netzwerk schicken. Die verschlüsselte Nachricht (C_1) kann er mitschneiden und kann dann alle Nachrichten mit gleichem IV entschlüsseln.

3. Nachrichten manipulieren

Da der CRC32-Algorithmus linear ist, lässt sich eine Nachricht so manipulieren, dass die Prüfsumme wieder dieselbe ist. Für die Manipulation des Inhalts ist allerdings Vorwissen über den Inhalt notwendig.

Beispiel: Manipulation einer IP-Adresse: Ein Angreifer muss lediglich wissen wie die IPs im Netzwerk organisiert sind (meistens ist die Organisation: 192.168.0.X). Die Position der IP-Adresse ist durch andere Protokolle (TCP/IP) immer an derselben Position der Nachricht. Um die IP-Adresse zu manipulieren wird vom Assoziativgesetz des exklusiven Oders Gebrauch gemacht:

- M_U: die ursprüngliche Nachricht
- K: die Verschlüsslungsbits des RC4
- C_U : die verschlüsselt ursprüngliche Nachricht

- M_M: die neue Nachricht
- C_M: die neue verschlüsselt Nachricht

$$(M_U \oplus K) = C_U$$

$$(M_U \oplus K) \oplus (M_U \oplus M_M) = \Big(K \oplus \big(M_M \oplus (M_U \oplus M_U)\big)\Big) = (K \oplus M_M) = C_M$$

Eine Nachricht/Paket an den WLAN-Router (IP-Adresse meistens: 192.168.0.1) kann an einen beliebigen andern Computer im Netzwerk weitergeleitet werden, in dem man die Bits der Nachricht mittels der oben genannten Formel manipuliert. Damit die Prüfsumme wieder gleich ist, muss an anderer Stelle der Nachricht (z.B. IP-Adresse des Ursprungs) entsprechend geändert werden. Diese ist ebenso wie die des Ziels durch TCP/IP an einer festen Position.

4. Fluhrer, Mantin and Shamir Attack (FMS)

Die FMS-Attacke wurde 2001 durch Scott Fluhrer, Itsik Mantin und Adi Shamir veröffentlicht. Diese nutzt eine Schwachstelle im RC4 in Kombination mit IVs aus, um den Schlüssel zu errechnen.

Voraussetzung ist jedoch dass die IVs eine bestimmte „schwache“ Struktur haben und dass das erste Byte des durch RC4 erzeugten Keystreams bekannt ist. Das erste Byte eines im WLAN (IEEE 802.11) zu verschlüsselnden Pakets ist immer der „Subnetwork Access Protocol“-Header (SNAP-Header: 0xAA). Das erste Byte eines übermittelten Paketes XOR dem SNAP-Header ergibt das benötigte erste Byte des Keystreams. Das heißt, das erste Byte des Keystreams lässt sich errechnen und ist somit implizit immer bekannt.

„Schwache“ IVs sehen wie folgt aus:

$$\{(A + 3);\ 0xFF;\ X\}$$

Sie haben als erstes Byte A+3. A ist die Stelle des Bytes im Schlüssel, die ermittelt werden soll. Das zweite Byte ist immer 255. Das letzte Byte kann einen beliebigen Wert annehmen.

Der Algorithmus durchläuft die Schleife des „Key Scheduling Algorithm“ nach der Initialisierung (A+3)-fach:

```
K[] = Array des Schlüssels
KL = Länge von K[]
S[] = Array der Länge 255
O = Erstes Byte des Keystreams
R = Mögliches Schlüssel-Byte

für i=0 bis 255
    S[i]=i
j=0
für i=0 bis (A+3)
    j = (j + S[i] + K[i modulo KL]) modulo 256
    tausch S[i] und S[j]
falls j>1
    R=O-j-S[A+3] modulo 256
```

Abbildung 5: FMS - KSA Pseudocode

Auf den ersten Blick erscheint der Algorithmus unvollständig, da K[] nicht bekannt ist. Allerdings sind die zur Berechnung notwendigen Teile, zu Beginn die drei Bytes des IVs und danach die zuvor ermittelten Bytes des Schlüssels, bekannt. Im Algorithmus ist auch erkennbar das auf die noch nicht ausgefüllten Werte von K[] nicht zugegriffen wird.
Folglich muss der Wert von A zu Beginn Null sein, da für den Algorithmus das erste Byte des Schlüssels notwendig ist um das zweite zu ermitteln, für das dritte wiederum das zweite usw.
Außerdem darf im letzten Durchlauf der Schleife weder S[0] noch S[1] getauscht werden. Dies wird im Pseudocode mittels j größer 1 überprüft, da i im letzten Durchlauf immer größer als 2 ist. Das Byte des Schlüssels erhält man durch die Subtraktion von j und S[A+3] von dem ersten Byte des Keystreams. Die Wahrscheinlichkeit, dass das Ergebnis dem echten Byte des Schlüssels entspricht, beträgt jedoch nur 5 Prozent. Um dennoch zuverlässig ein Byte des Schlüssels zu ermitteln, müssen mehrere Pakete mit „schwachen“ IVs verarbeitet werden. Mit mehr als 60 verschiedenen IVs (verschiedener X Wert) lässt sich diese Wahrscheinlichkeit auf über 50 Prozent steigern. Sollte jedoch ein Byte des vermuteten Schlüssels falsch sein, sind höchstwahrscheinlich auch alle folgenden Bytes falsch.[5]

[5] Vgl. (Erickson, 2008) (Lashkari, et al., 2009) (Scott Fluhrer, 2001) (FMS_Wikipedia)

WEP-Simulation

Modellierung

Um die Anwendung einer WEP-Verschlüsselung zu simulieren, benötigt man ein modellhaftes WLAN. Einerseits sollten sich Computer über das WEP im WLAN anmelden können, andererseits ist die Funkkommunikation nicht abhörsicher, so dass auch Computer ohne das Passwort die Kommunikation mitschneiden und Nachrichten/Pakete schicken können.
Eine Kommunikation zwischen zwei Computern besteht in der Simulation lediglich aus dem Antworten auf eine Nachricht mit derselben Nachricht. Deshalb wird das ständige Verschicken der gleichen Nachricht durch eine erste Nachricht eingeleitet.

Implementation

Der gesamte Sourcecode ist unter der Adresse: https://github.com/APN-Pucky/WEPSimulation hinterlegt.
Die Simulation wurde unter Windows 7 mit Eclipse, Git und Java (1.8) programmiert.

Jeder bereits aufgeführte Algorithmus (*CRC32*, *RC4*, *KSA*, *PRGA*) wurde in einer eigenen Klasse implementiert, wobei die RC4 Klasse lediglich KSA und PRGA kombiniert. Die Größe des WEP-Schlüssels beträgt in der Simulation 5 Byte. Die Größe des Schlüssels lässt sich allerdings schnell in der WEP-Klasse ändern (*WEP.key_size = n*).
Das oben modellhaft beschriebene WLAN wurde in der Klasse *Router* implementiert. Mit der Funktion *void:addListener(l : Listener)* kann ein Objekt, welches das Interface *Listener* implementiert, dem Netzwerk des *Routers* hinzugefügt werden, d.h. es empfängt alle gesendeten Pakete/Nachrichten. Die Funktion *void:sendTo(msg : byte[], from : byte[], to : byte[])* übermittelt die zuvor verschlüsselte Nachricht (*msg*) an den Adressaten (*to*, ebenfalls verschlüsselt), jedoch nur wenn dieser sich mit dem *Router* verbunden hat (nicht nur als *Listener* zuhört). Um sich zu Verbinden muss an den Router eine aus Computer- und Router-Name zusammengefügte Nachricht geschickt werden. (Dies entspricht nicht dem wirklichen Authentifikation Verfahrens bei WEP-WLANs).
Das Anfügen des SNAP-Headers an Nachrichten ist für das Funktionieren der FMS-Attacke fundamental. Deshalb wird er in der Simulation verwendet.

Der FMS-Algorithmus wurde in der Klasse *Attacker* implementiert. Das *Attacker*-Objekt fügt sich wie ein Computer dem WLAN als *Listener* hinzu, um gesendete Pakete abzufangen.

Diese Pakete werden mit Hilfe der zuvor erläuterten Methode verwendet um den originalen Schlüssel zu berechnen. In die *Attacker*-Klasse wurde zusätzlich ein Algorithmus, der aus allen zuvor errechneten möglichen Schlüssel-Bytes, den am häufigsten vorkommenden herausfiltert, entwickelt. Der Algorithmus sortiert das Array. Danach wird das Array erneut durchlaufen und es wird gezählt welcher Wert am häufigsten vorkommt. Dieser Wert ist das Schlüssel-Byte. Dies erfolgt für jedes Schlüssel-Byte. Wenn der Schlüssel komplett ermittelt wurde, endet das Programm mit der Ausgabe des Schlüssels.

Probleme und Lösungen

Ein Problem, das sehr früh erkannt wurde, ist, dass in Java alle primitiven Datentypen „signed“ sind, das heißt sie umfassen sowohl den negativen als auch den positiven Wertebereich. Da der Algorithmus der WEP-Verschlüsselung oft den Modulo 256 Operator benutzt, erhält man, wenn man Javas Byte-Datenstruktur (von -128 bis 127) verwendet, nicht das gewünschte Ergebnis. Lösung des Problems war die Verwendung eines alternativen primitiven Datentyps der den Wertebereich von 0 bis 255 einschließt (z.B. int oder short). In der Simulation wurden die Algorithmen mit Integer-Datenstruktur implementiert.

Ein Paket, das zunächst nicht verwendet werden kann, um das gesuchte Byte (A) des Schlüssels zu berechnen (weil IV noch nicht die geforderte Struktur hat), wird in der *Attacker*-Klasse gespeichert, damit es später für die Berechnung eines anderen Bytes des Schlüssels herangezogen werden kann.

Anwendung

Die Simulation kann mit Hilfe der statischen Methode *main* der Klasse *Main* gestartet werden.

Die versendete Nachricht ist „Hallo, PC in der WEP-Simulation“. Der Inhalt der Nachricht ist aber irrelevant, da zum Entschlüsseln nur der SNAP-Header verwendet wird.

Die Ausgabe besteht aus einer Liste der möglichen Bytes des Schlüssels sowie des am häufigsten vorkommenden Bytes. Die Anzahl der verschickten Pakete sowie die Dauer der Entschlüsselung des Schlüssels und der Schlüssel selbst werden am Ende der Ausgabe aufgelistet.

Die Durchführung mit zufälligen IVs führte erwartungsgemäß zu sehr unterschiedlichen Laufzeiten. Um die Laufzeiten für verschiedene Passwörter zu vergleichen wurden lineare IVs von {0; 0; 0} bis {0xFF; 0xFF; 0xFF} verwendet (siehe exemplarische Ausgabe im Anhang).

Fazit

Das WLAN-Verschlüsselungsprotokoll WEP ist keine sichere Verschlüsslungsmethode, da der verwendete Schlüssel aus gesendeten Nachrichten durch den FMS-Algorithmus abgeleitet werden kann. Wie die Simulation gezeigt hat, ist dies in sehr kurzer Zeit möglich.
Im Beispiel mit linearen IVs und einem Passwort der Länge fünf, betrug die durchschnittliche Laufzeit etwa zwei Minuten.
Außerdem garantiert WEP nicht, dass die Nachricht nur vom Empfänger entschlüsselt werden kann. Noch gravierender ist aber, dass eine Manipulation bzw. Vervielfältigung von Nachrichten nicht ausgeschlossen ist.
Die heutzutage vermutlich sicherste und auch am meisten genutzte Verschlüsselung von kabellosen Netzwerken ist WPA2, dies ist die zweite Version des Protokolls WPA, welches WEP abgelöst hat. Für WPA2 sind bisher keinerlei Sicherheitslücken bekannt.

Anhang

Exemplarische Ausgaben

Beispiel 1:

```
Potential 0. Key Bytes:
      '&',  ' ',  'z',  'q',  '?',  '_',  '!',  '?',
      '&',  '(',  'V',  '?',  '?',  'G',  '?',  '_',
      'G',  'd',  '?',  '?',  '_',  'e',  '?',  '?',
      '?',  '?',  'd',  '?',  's',  'd',  '2',  '?',
      '_',  '?',  ')',  '?',  '<',  '?',  'Y',  '?',
      'd',  'R',  'F',  'd',  '"',  '\r', '?',  '?',
      '!',  'X',  '_',  '_',  '?',  '?',  'D',  '8',
      '~',  'J',  '/',  'i',
      |
      '-> Key-Byte: d

Potential 1. Key Bytes:
      [...]
      |
      '-> Key-Byte: a

Potential 2. Key Bytes:
      [...]
      |
      '-> Key-Byte: s

Potential 3. Key Bytes:
      [...]
      |
      '-> Key-Byte: P

Potential 4. Key Bytes:
      [...]
      |
      '-> Key-Byte: W

After 1.8207886885833333 minutes
And 1048173 packets
------------------------------------
575669.7669379374 packets/minute
------------------------------------
Full WEP-Key: 'dasPW'
```

Beispiel 2:

```
Potential 0. Key Bytes:
       [...]
       |
       '-> Key-Byte: I

Potential 1. Key Bytes:
       [...]
       |
       '-> Key-Byte: n

Potential 2. Key Bytes:
       [...]
       |
       '-> Key-Byte: f

Potential 3. Key Bytes:
       [...]
       |
       '-> Key-Byte: o

Potential 4. Key Bytes:
       [...]
       |
       '-> Key-Byte: !

After 2.1221564038 minutes
And 1048173 packets
------------------------------------
493918.8262105038 packets/minute
------------------------------------
Full WEP-Key: 'Info!'
```

Abkürzungsverzeichnis

Abkürzung	**Begriff**
CRC	Cyclic Redundancy Check
FMS	Fluhrer, Mantin and Shamir Attack
GCHQ	Government Communications Headquarters
IEEE	Institute of Electrical and Electronics Engineers
IV	Initialization Vector
KSA	Key Scheduling Algorithm
NSA	National Security Agency
PRGA	Pseudo-Random Generation Algorithm
RC4	River Cipher 4
SNAP	Subnetwork Access Protocol
WEP	Wired Equivalent Privacy
WLAN	Wireless Local Area Network
WPA	Wi-Fi Protected Access
XOR	Kontravalenz/ exlusives Oder/ exclusive or

Abbildungsverzeichnis

Literaturverzeichnis

Erickson Jon Hacking: The Art of Explotation [Book]. - San Francisco : No Starch Press, Inc, 2008. - 2. - ISBN 978-1-59327-144-2.

Fluhrer, Mantin and Shamir Attack [Online] // Wikipedia. - März 18, 2015. - https://en.wikipedia.org/wiki/Fluhrer,_Mantin_and_Shamir_attack.

IEEE 802.11 [Online] // Wikipedia. - März 17, 2015. - https://de.wikipedia.org/wiki/IEEE_802.11.

Institute of Electrical and Electronics Engineers IEEE 802.11TM WIRELESS LOCAL AREA NETWORKS [Online]. - März 17, 2015. - http://www.ieee802.org/11/.

Kontravalenz [Online] // Wikipedia. - März 17, 2015. - https://de.wikipedia.org/wiki/Kontravalenz.

Krüger Guido and Hansen Heiko Handbuch der Java Programmierung [Book]. - München : Addison-Wesley Verlag, 2012. - ISBN 978-3-8273-2751-2.

Lashkari Arash Habibi, Danesh Mir Mohammad Seyed and Samadi Behrang A Survey on Wireless Security protocols (WEP,WPA and WPA2/802.11i) [Conference] // IEEE Conference. - Beijing, China : [s.n.], 2009. - ISBN 978-1-4244-4519-6.

RC4 [Online] // Wikipedia. - März 18, 2015. - https://en.wikipedia.org/wiki/RC4.

Scott Fluhrer Itsik Mantin, Adi Shamir Weaknesses in the Key Scheduling Algorithm of RC4 [Online] // Crypto. - 2001. - März 18, 2015. - http://www.crypto.com/papers/others/rc4_ksaproc.pdf.

Ullenboom Christian Java ist auch eine Insel [Book]. - Bonn : Galileo Press GmbH, 2005. - ISBN 3-89842-526-6.

Wired Equivalent Privacy [Online] // Wikipedia. - März 17, 2015. - https://en.wikipedia.org/wiki/Wired_Equivalent_Privacy; https://de.wikipedia.org/wiki/Wired_Equivalent_Privacy.

Wireless LAN [Online] // Wikipedia. - März 17, 2015. - https://en.wikipedia.org/wiki/Wireless_LAN.

Zyklische Redundanzprüfung [Online] // Wikipedia. - März 17, 2015. - https://de.wikipedia.org/wiki/Zyklische_Redundanzpr%C3%BCfung.

Erklärung über die selbstständige Anfertigung der Arbeit

Erklärung:

Hiermit erkläre ich, dass ich die vorliegende Facharbeit selbstständig und ohne fremde Hilfe verfasst und keine anderen als die im Literaturverzeichnis angegebenen Hilfsmittel verwendet habe. Darüber hinaus versichere ich, dass ich alle **wörtlichen** und **sinngemäßen** Übernahmen aus anderen Werken oder dem Internet als solche **kenntlich** gemacht habe.

Ich erkläre mich damit einverstanden, dass die von mir verfasste Facharbeit Dritten zugänglich gemacht wird.

_____ den _____ _____

Ort Datum Unterschrift