Sicherheit in der Cloud

rc4b

Inhalt

Artikel

Quellennachweise

Artikellizenzen

One-Time-Pad

```
CIHJT UUHML FRUGC ZIBGD BQPNI PDNJG LPLLP YJYXM
DCXAC JSJUK BIOYT MWQPX DLIRC BEXYK VKIMB TYIPE
UOLYQ OKOXH PIJKY DRDBC GEFZG UACKD RARCD HBYRI
DZJYO YKAIE LIUYW DFOHU IOHZV SRNDD KPSSO JMPQT
MHQHL OHQQD SMHNP HHOHQ GXRPJ XBXIP LLZAA VCMOG
AWSSZ YMFNI ATMON IXPBY FOZLE CVYSJ XZGPU CTFQY
HOVHU OCJGU QMWQV OIGOR BFHIZ TYFDB VBRMN XNLZC
```

Beispiel eines One-Time-Pads

Das **One-Time-Pad** (Abkürzung: **OTP**, deutsch: **Einmalverschlüsselung** oder **Einmalschlüssel-Verfahren**, wörtlich *Einmal-Block*, nicht zu verwechseln mit dem *Einmal-Passwort-Verfahren*) ist ein symmetrisches Verschlüsselungsverfahren zur geheimen Nachrichtenübermittlung. Kennzeichnend ist, dass ein Schlüssel verwendet wird, der (mindestens) so lang ist wie die Nachricht selbst. Das OTP ist informationstheoretisch sicher und kann nachweislich nicht gebrochen werden – vorausgesetzt, es wird bestimmungsgemäß verwendet.

Geschichte

Joseph Mauborgne

Das Verfahren geht auf den amerikanischen Kryptologen Gilbert Vernam (1890–1960) zurück, der die Idee dazu im Jahre 1918 geäußert hat. Der Amerikaner Joseph O. Mauborgne (1881–1971) setzte diese Idee um und nannte das Verfahren *„One-Time Pad“* (deutsch: *Einmal-Block*). Kurz darauf arbeiteten auch die Deutschen Werner Kunze, Rudolf Schauffler und Erich Langlotz an dieser Methode. Sie schlugen im Jahr 1921 vor, Blöcke, die mit zufällig erstellten Ziffern bedruckt waren, zur Überschlüsselung der damaligen diplomatischen Codes zu verwenden und bezeichneten diese als *i-Wurm* (individueller Wurm). Diese Methode wurde vom diplomatischen Dienst der Weimarer Republik auch tatsächlich eingesetzt.

Seit dieser Zeit bis zum heutigen Tag, speziell auch während der Zeit des Kalten Krieges, wird dieses Verfahren verwendet. Beispielsweise war der „Heiße Draht“ (auch als das „Rote Telefon“ bekannt), also die hochsichere direkte Fernschreibverbindung zwischen dem amerikanischen Präsidenten und dem sowjetischen Generalsekretär durch ein Einmalschlüssel-Verfahren geschützt.[1]

Verfahren

Beschreibung

Das One-Time-Pad gehört zu den polyalphabetischen Substitutionsverfahren, bei denen die einzelnen Buchstaben (oder Zeichen) in jeweils andere Buchstaben (oder Zeichen) umgewandelt (verschlüsselt) werden. Kennzeichnendes Merkmal der Einmalverschlüsselung ist die *einmalige* Verwendung eines *zufälligen* Schlüssels, der (mindestens) die gleiche Länge wie die zu verschlüsselnde Nachricht aufweist.

Bei dem One-Time-Pad sind sowohl der Klartext, als auch der Schlüssel und das Chiffrat Zeichenketten der Länge n. Die Menge der verwendeten Zeichen, auch Alphabet genannt, ist in der modernen Kryptologie üblicherweise die Menge der Bits $\{0,1\}$, in der klassischen Kryptographie die Menge aller Großbuchstaben. Zur Verschlüsselung wird der Schlüssel zeichenweise modulo der Alphabetgröße auf den Klartext addiert. Bei der Verwendung von Bits entspricht das einer Exklusiv-Oder-Verknüpfung (XOR) der einzelnen Bits. Bei der Verwendung von

Großbuchstaben werden die Buchstaben wie bei der Caesar-Verschlüsselung addiert. Dazu werden sie zuerst auf die Zahlen von 0 bis 25 abgebildet, dann modulo 26 addiert (es wird vom Ergebnis 26 abgezogen, falls es größer ist als 25) und das Ergebnis wieder in Buchstaben umgewandelt.

Grundlegende Voraussetzungen für die Sicherheit des Einmalschlüssel-Verfahrens sind: Der Einmalschlüssel muss

- genauso lang sein wie die Nachricht,
- gleichverteilt zufällig gewählt werden,
- geheim bleiben und
- darf nicht wiederverwendet werden, auch nicht teilweise .

Damit erfüllt das Einmalschlüsselverfahren Kerckhoffs' Prinzip, nach dem die Sicherheit eines Kryptosystems nicht von der Geheimhaltung des Algorithmus abhängen darf, sondern nur von der Geheimhaltung des Schlüssels.

Das Einmalschlüsselverfahren ist gut geeignet für eine maschinelle Realisierung. Im Gegensatz zu vielen modernen kryptographischen Methoden (wie DES, PGP oder AES), die wegen ihrer Komplexität auf Computer angewiesen sind, eignet sich das One-Time Pad jedoch ebenso gut zur manuellen Durchführung der Ver- und Entschlüsselung.

Eng verwandt mit dem One-Time-Pad ist die Stromverschlüsselung, bei welcher der Schlüssel pseudozufällig erzeugt wird. Andere historische und auch aktuelle kryptographische Verfahren verwenden Schlüssel, die in der Regel deutlich kürzer sind als die Länge des zu verschlüsselnden Klartextes.

Sicherheit

Die vier Bedingungen der Schlüsselwahl stellen zusammen mit der Beschreibung des Verfahrens sicher, dass die folgenden Voraussetzungen erfüllt sind.

- Es gibt genausoviele Schlüssel wie mögliche Chiffrate.
- Zu jedem Klartext-Chiffratpaar gibt es genau einen Schlüssel, der auf den Klartext angewendet das Chiffrat ergibt.

Unter diesen Bedingungen ist das Verfahren informationstheoretisch sicher (auch *perfekte Sicherheit* genannt) und kann auch mit beliebig hohem Rechenaufwand nicht gebrochen werden.[2] Allein mit Kenntnis des Schlüssels kann der Geheimtext entschlüsselt werden. Diesen zu raten ist praktisch unmöglich, da jeder mögliche Schlüssel mit der gleichen Wahrscheinlichkeit gewählt wurde. Zudem gibt es keine Möglichkeit, herauszufinden ob ein Schlüssel richtig geraten wurde oder nicht. Bei einer anderen Schlüsselwahl, beispielsweise der Verwendung von Textpassagen, wäre diese Eigenschaft nicht gegeben.

Beispiel

Eine einfache Handmethode zur Verschlüsselung ist beispielsweise die buchstabenweise Addition von Klartext und Schlüssel. Hierzu ersetzt man zunächst mithilfe einer beliebigen Substitutionstabelle die Buchstaben des Klartextalphabets durch Zahlen. Im einfachsten Fall ordnet man den 26 Großbuchstaben des lateinischen Alphabets Zahlen zu, die ihrer Position im Alphabet entsprechen. Mit anderen Worten, man nummeriert das Alphabet wie folgt durch:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26

Jetzt ist eine buchstabenweise Addition leicht möglich. Beispielsweise ergibt die Addition von **A** und **F** den Buchstaben **G**, entsprechend ihren Platznummern **1 + 6 = 7**. Falls die Summe den Wert 26 überschreiten sollte, so zieht man einfach 26 ab (Modulo-Operation) und erhält so wieder einen der 26 Alphabetbuchstaben. Beispielsweise **X** plus **U** ist numerisch **24 + 21** = 45, nach abziehen von 26 ergibt sich **19** und damit der Buchstabe **S**, also **X + U = S**.

Die Zusammenhänge bei der Addition von Buchstaben lassen sich an der folgenden Tabelle, die Ähnlichkeit mit einer klassischen *Tabula recta* hat, übersichtlich darstellen:

	A	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

In der kursivgedruckten oberen Zeile und in der ersten Spalte am linken Rand sind die beiden zu addierenden Summanden angegeben. Im Kreuzungspunkt innerhalb der Tabelle lässt sich die Summe ablesen, also das Ergebnis der Summation von Klartext-Buchstaben und Schlüsselbuchstaben. Dies ist der entsprechende Buchstabe des Geheimtextes.

Zur Verschlüsselung wird man einen zufälligen Schlüssel benutzen, der in diesem Beispielfall passenderweise ebenfalls aus den 26 Großbuchstaben zusammengesetzt ist und dessen Länge (mindestens) der Länge des zu verschlüsselnden Klartextes entspricht. Entscheidend für die Sicherheit der Verschlüsselung ist, dass die einzelnen Buchstaben des Schlüssels wirklich zufällig verteilt sind, unvorhersagbar sind und in keinerlei Zusammenhang untereinander stehen. Als Beispiel für einen zufälligen Schlüssel dient die Buchstabenfolge unten. Sie wurde mit einem aufwändigen Verfahren generiert, das in der Lage ist, sehr gute Zufallsfolgen zu erzeugen:

```
S = WZSLXWMFQUDMPJLYQOXXB
```

Der Schlüssel S ist in diesem Beispiel recht kurz, er umfasst nur 21 Buchstaben, und ist bei bestimmungsgemäßer Verwendung sehr schnell „verbraucht“, nämlich bereits nach Verschlüsselung eines Textes aus 21 Buchstaben.

Beispielsweise soll der folgende Klartext K verschlüsselt werden:

```
K = ANGRIFFIMMORGENGRAUEN
```

Zur Verschlüsselung werden Klartext K und Schlüssel S, wie oben erläutert, buchstabenweise addiert. Als „Summe“ (K + S = G) erhält man nach der so durchgeführten Einmalverschlüsselung den Geheimtext G:

```
G = XNZDGCSODHSEWOZFIPSCP
```

Der im Ergebnis erhaltene Geheimtext G ist von einem Zufallstext nicht zu unterscheiden und kann prinzipiell mit keiner noch so gearteten kryptanalytischen Angriffsmethode (weder jetzt noch in Zukunft) entziffert werden. Allein die Kenntnis des Schlüssels S erlaubt es, aus dem Geheimtext G durch Subtraktion des Schlüssels wieder den Klartext K zu gewinnen. Ohne den Schlüssel kann man prinzipiell alle denkbaren und mehr oder weniger sinnvollen Buchstabenkombinationen aus 21 Buchstaben „konstruieren“. Theoretisch könnte ein Angreifer dies probieren. Er würde so eine Unmenge an Sätzen erhalten, die in beliebigen Sprachen beliebige Informationen verkünden würden, beispielsweise

```
K' = WIKIPEDIAFINDENWIRGUT
```

mit dem dazu „passenden“ Schlüssel, der der Differenz zwischen Geheimtext G und dem konstruierten Pseudo-Klartext K' entspricht (S' = G - K'):

```
S' = AEOUQXOFCBJQSJLIZXLHV
```

Dieser Schlüssel S' erfüllt die Bedingung, dass die buchstabenweise Summe von ihm mit dem oben erzeugten (falschen) Klartext K' genau den gleichen Geheimtext ergibt wie die Summe aus dem echten, aber dem Angreifer unbekannten Schlüssel S mit dem echten, aber dem Angreifer ebenso unbekannten Klartext K. So kann der Angreifer eine unübersehbare Fülle von denkbaren Klartext-Schlüsselpaaren konstruieren, die in (buchstabenweiser) Summe alle den gleichen echten Geheimtext ergeben. Er hat jedoch keine Möglichkeit, daraus auf den echten Klartext zurückzuschließen. Auch Brute Force, also das erschöpfende Durchprobieren aller möglichen Schlüssel, führt nicht zum Erfolg. Der Angreifer kann zwar so – sogar ohne überhaupt den Geheimtext zu kennen – alle denkbaren Texte mit 21 Buchstaben erzeugen, unter denen natürlich auch der ursprüngliche sein wird. Es gibt jedoch keinerlei Anhaltspunkte zu entscheiden, welcher aus der Unmenge der durchaus sinnvollen Texte der tatsächliche Klartext ist. Solange ihm der Schlüssel nicht in die Hände fällt, bleibt der Klartext auf ewig ein Geheimnis.

Um diese Sicherheit nicht zu gefährden, sollte der Einmalschlüssel nach bestimmungsgemäßem Gebrauch unwiederbringlich vernichtet werden.

Angriffsmöglichkeiten

Bei korrekter Anwendung des Einmalschlüsselverfahrens ist es, wie der amerikanische Wissenschaftler Claude Shannon in den 1940er-Jahren zeigte [3], nachweislich sicher und kann nicht gebrochen werden. In der praktischen Anwendung können jedoch Fehler passieren, die einen Einbruch möglich machen. Ursache dafür ist die Verletzung einer oder mehrerer der im Abschnitt „Beschreibung“ formulierten grundlegenden Sicherheits-Voraussetzungen.

Ausspähen des Schlüssels

Handelsübliche Datenträger zur Speicherung von Einmalschlüsseln

Ein möglicher Fehler ist, den Einmalschlüssel nicht geheim zwischen Sender und Empfänger auszutauschen, so dass er durch Dritte ausgespäht werden kann. Auch eine sichere Aufbewahrung des Schlüssels ist essentiell. Im Gegensatz zu Passwörtern oder Kennwörtern lässt sich ein Einmalschlüssel kaum merken. Er muss auf irgendeinem Medium, sei es Papier, Datendiskette, CD-ROM oder USB-Stick gespeichert sein. Dieses Medium kann kopiert werden und der Schlüssel ist dann kompromittiert. Ebenso darf nicht vergessen werden, den Einmalschlüssel nach Gebrauch sicher und unwiederbringlich zu vernichten. Gelingt es dem Angreifer, den Schlüssel nachträglich zu finden oder zu restaurieren, so ist die Kommunikation entlarvt.

Nicht ausreichend zufälliger Schlüssel

Ein ebenso schon mehrfach begangener Fehler ist die unzureichende Zufälligkeit und Unabhängigkeit der einzelnen Zeichen des Schlüssels. Ein Kardinalfehler ist, als Einmalschlüssel keine zufällige Buchstabenfolge, sondern eine Textpassage zu benutzen. Selbst, wenn der verwendete Text einmalig ist und niemandem (außer den beiden Kommunikationspartnern) bekannt ist, weisen Buchstabenfolgen, die aus einem „sinnvollen“ Text stammen, im Gegensatz zu zufälligen Buchstabenfolgen (Zufallstexten) statistisch auswertbare Abhängigkeiten auf, die eine Entzifferung möglich machen können. Selbst *manuell* erzeugte pseudozufällige Zeichenfolgen genügen häufig nicht höchsten Ansprüchen an die Sicherheit, denn sie weisen eine Reihe von durchaus auffälligen Merkmalen auf, die ein Angreifer zu seinem Vorteil nutzen könnte. Beispielsweise scheuen viele Menschen bei der Kreation von möglichst zufälligen Texten vor Buchstabenverdopplungen und erst recht -verdreifachungen zurück, wohingegen echte Zufallstexte nicht selten zwei oder auch mehr identische Buchstaben in direkter Folge oder in kurzem Abstand enthalten können.

Der folgende Zufallstext (der, wenn er nicht hier in der Wikipedia veröffentlicht wäre, sehr gut als Einmalschlüssel geeignet wäre) wurde nach einem speziellen Verfahren maschinell erzeugt. Er besteht aus 1000 Buchstaben, die tatsächlich „gut“ zufällig verteilt sind. Interessanterweise erkennt man jedoch in der sechsten Zeile in kurzem Abstand hintereinander fünfmal den Buchstaben **Z** (rot hervorgehoben).

```
RBFALAZJGPUCWFCMPQUCEDTCXGXRGUHXATFYVLVCMQRJSDXRYU
XMIOPUNYGTQOMQKRNCBPTBSCQZOICFQENNHWJFAAUTEXDYOMKY
HPJVENSEFLKNYQMWKXFQEKPSYBCTRZSNRDPGOJLLDQNYSPMWBX
XKFBEPJBUCIOLUGHBAUTXUSBAXSCTLDTFHSOQOIHAZAVQFINLG
NFSVFWIIGDVCAPDWBRWIKJCUVQCDHPSBNUYCHALZRNTNFHMOOY
GMQDAQASBKSGIZIZZTZCLNZNCMTVJBCFUBIEQSQNLPUMBCDBEH
FSXGJADMYGDLXRKFKWSVCYQDGBILEBIEXEKGEUTJAOBZAKPJEJ
YBNAWUCSPJZBTNMVHLKCDCPHSVNKWRGGWRAZLWSYCDXOEYKMRR
NTOSHBFEXKKSKOSOPFRBYNZSDXTSJMIYNQFEHLBEEZSXRLPEWM
QHKYWALNGMPJMYEXLPVXZYDMSABJAHLVPEESQYRHNBCQSBOUVR
XARCSSNFRTWGHNWPRODKXHROCWZWZGJFVLEWYVDJANGJXWEIGT
BHATMVYSCISMEPMORPWVQJRBFVZLSNMTKMEVYQYJQRDEHGPGVM
GNPGCCBIBNJNVSHQZWODDYQEBAHKVOTCIUAAIOAERRPSEWIIOP
OBDMMDXTRYWOFSWYLNDFQDDRPJFDPEOBAYWAOYDMUMOEWZVNMW
ZGIAHASQJSGUXXBCURDQLSQLXONHNHRIFVMCKQLUYATXPCCQGG
XHVFDALBJVRVEJGASEAXAWZZWJDAGASQHXINCADLBEOKNEDRJD
UPGXDTYEIQWIHRYZAMQHDVNAFQKQDWIDMDXLDYZZFWSTAPAPQL
VYUTAYKGPBOHHVGIVUOTEOVZTRGMWPZCKYCKVKXZJMTMUKQGVQ
TUGCOEJHUCMROKQWMPTLADKNCFPYEXCBSPHVWYRQSJFTGQGJHJ
YIVSQGIAUJJIBYUPCPWLKRLFIZYKVCYZXDSYPJLWHRIEXVQZGR
```

Ein Mensch würde – vor die Aufgabe gestellt, eine möglichst zufällige Buchstabenfolge zu erzeugen – wohl davor zurückschrecken, in so kurzem Abstand fünfmal auf die gleiche Taste zu tippen. Er würde die Buchstaben eher gleichmäßiger verteilen – was tatsächlich aber ein Fehler wäre und keine gute Zufälligkeit ergäbe, denn er hätte dabei den Grundsatz verletzt: „Der Zufall hat kein Gedächtnis!“

Ebenso wäre es ein Fehler, als Einmalschlüssel eine Buchstabenfolge zu verwenden, die von einem Pseudozufallszahlengenerator erzeugt wurde, so wie er von fast allen Hochsprachen angeboten wird (meist RND von engl. *random* für „zufällig“ genannt). Hier muss davon ausgegangen werden, dass ein Angreifer den verwendeten Algorithmus kennt, und es ihm dann genügt, den zumeist nur wenige Byte langen Startwert (genannt: „die Saat“) des Generators zu finden, um den Schlüssel zu rekonstruieren. Ein weiterer (eher theoretischer) Nachteil von Pseudozufallszahlengeneratoren ist ihre beschränkte Periodenlänge, nach der sich die Pseudozufallsfolge wiederholt.

Möglich ist, einen PC mit einer speziellen Hardware zur Erzeugung von Zufallsereignissen zu erweitern. Hierzu dienen beispielsweise Rauschgeneratoren. Einfacher beschränkt man sich auf gewisse nichtdeterministische Ereignisse, die von der Interaktion des PC mit dem Menschen herrühren, wie beispielsweise die Zeitabstände

zwischen Tasteneingaben oder Mausbewegungen. Auch kann man Signale von einem angeschlossenen Mikrofon benutzen. Allerdings ist bei diesen einfachen Methoden die „Menge“ an Zufallsereignissen relativ klein, so dass sie nicht ausreichen, um lange Zufallsschlüssel mit guter Qualität zu erzeugen (siehe auch: Entropietests).

Mehrfachverwendung des Einmalschlüssels

Ein in der praktischen Anwendung (Beispiele: Lorenz-Schlüsselmaschine und Venona) schon häufig passierter Fehler ist, mehr als nur die beiden allein für Sender und Empfänger bestimmten Kopien des Einmalschlüssels herzustellen und zu verteilen oder den Schlüssel mehr als einmal zur Verschlüsselung zu verwenden. Schon eine zweimalige Verwendung eines Einmalschlüssels genügt, um die Kommunikation erfolgversprechend angreifen zu können. Dies kann an einem Beispiel illustriert werden: Nachdem der oben erwähnte Klartext K mit dem Schlüssel S verschlüsselt worden ist, soll am nächsten Tag ein neuer Befehl, nämlich der Klartext K2 (mit völlig anderem Inhalt), verschlüsselt übermittelt werden.

```
K2 = RUECKZUGVONDENHUEGELN
```

Fehlerhafterweise wird zur Verschlüsselung der bereits „verbrauchte“ Einmalschlüssel S vom Vortag ein zweites Mal benutzt.

```
 S = WZSLXWMFQUDMPJLYQOXXB
```

Wieder werden zur Verschlüsselung Klartext und Schlüssel buchstabenweise addiert. Als „Summe“ erhält man in diesem Fall den Geheimtext G2:

```
G2 = OUXOIWHMMJRQUXTTVVCJP
```

Für sich allein ist auch der Geheimtext G2 unknackbar. Hat der Angreifer jedoch zusätzlich den Geheimtext G vom Vortag abgefangen, so kann er beide miteinander kombinieren. (Zur Deutlichkeit und besseren Unterscheidung zum Geheimtext G2 wird der erste Geheimtext G im Folgenden mit G1 bezeichnet.)

```
G1 = XNZDGCSODHSEWOZFIPSCP
G2 = OUXOIWHMMJRQUXTTVVCJP
```

Der Angreifer geht dabei von folgender Überlegung aus: Angenommen, der Absender der beiden Geheim-Nachrichten G1 und G2 hat in beiden Fällen (versehentlich) denselben Schlüssel verwendet, dann gelten die folgenden Beziehungen zwischen den vorliegenden beiden Geheimtexten G1 und G2, dem unbekannten, aber in beiden Fällen identischen Schlüssel S und den beiden unbekannten Klartexten K1 und K2:

```
G1 = K1 + S
G2 = K2 + S
```

Durch Differenzbildung der beiden Geheimtexte kann der Angreifer nun den Schlüssel eliminieren!

```
D = G1 − G2 = (K1 + S) − (K2 + S) = K1 − K2
```

Die Differenz D der beiden Geheimtexte, die der Angreifer ja kennt, ist identisch zur Differenz der beiden Klartexte (K1 - K2):

```
D = ISBOXFKBQXANBQFLMTPSZ
```

Dies ist der Angriffspunkt zur Entzifferung. Klartexte und folglich auch Differenzen von Klartexten zeigen nämlich im Gegensatz zu Zufallstexten eine Reihe von Auffälligkeiten, die statistisch ausgewertet und zur Entzifferung ausgenutzt werden können.

So zeigt das Häufigkeitsgebirge von Differenztexten ebenso charakteristische Auffälligkeiten wie beispielsweise das von deutschen Klartexten oder von monoalphabetisch verschlüsselten Texten. Die folgende Tabelle zeigt als

Ergebnis einer Auszählung der Differenz von zwei deutschen Texten aus jeweils einer Million Buchstaben die relativen Häufigkeiten der Buchstaben in ‰ (Promille). Bei einer Gleichverteilung würde man jeden der 26 Buchstaben mit einer Häufigkeit von 1/26, also mit 38,5 ‰ erwarten.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
43	31	33	48	34	30	33	31	42	39	37	37	50	38	38	39	44	30	33	32	33	44	32	32	41	77

Bei Differenztexten – falls sie von zwei Klartexten oder von zwei nach dem Einmalschlüssel-Verfahren mit identischem Schlüssel verschlüsselten Geheimtexten stammen – dominiert der Buchstabe Z mit etwa 77 ‰, der durch Koinzidenzen von identischen Buchstaben in beiden Klartexten und damit auch (bei zweifacher Verwendung desselben Einmalschlüssels) in beiden Geheimtexten auftritt. Ein Nebenmaximum tritt beim Buchstaben M mit 50 ‰ auf. Ursache dafür sind Koinzidenzen der in deutschsprachigen Texten häufigen Buchstaben E und R beziehungsweise R und E, die beide die gleiche Differenz, nämlich 13, aufweisen. Falls der Angreifer diese Auffälligkeiten am Differenztext entdeckt, bestätigt dies seinen Verdacht der Mehrfachverwendung eines Einmalschlüssels (siehe auch: Koinzidenzindex). Basierend auf weiteren statistischen Abhängigkeiten und mithilfe von speziellen Differenztextbasen sowie passender Trigramm-, Tetragramm- und Pentagramm-Tabellen und rechnergestützter Untersuchung der verschiedenen Fälle können die Geheimtexte nun erfolgversprechend angegriffen werden.

Auf diese Weise kann das theoretisch unknackbare Einmalschlüssel-Verfahren plötzlich dennoch gebrochen werden, falls bei der praktischen Anwendung Fehler passieren.

Zusammenfassung

Vorteil

Das One-Time-Pad ist informationstheoretisch sicher und kann nicht entziffert werden, wenn der Schlüssel genauso lang ist wie die Nachricht und aus Zeichen besteht, die zufällig und unabhängig sind, und wenn er nur einmal zur Verschlüsselung verwendet wird. Unter diesen Voraussetzungen kann der Geheimtext nur mit Kenntnis des Schlüssels entschlüsselt werden.

Andere Verschlüsselungsverfahren (wie AES) erreichen ihre Sicherheit durch den immensen Berechnungsaufwand der theoretisch denkbaren Entzifferung, der praktisch nicht realisierbar ist. Mit anderen Worten, einem potenziellen Angreifer fehlt es an notwendigen Ressourcen (z. B. Rechenleistung oder Zeit), um seinen Entzifferungsversuch erfolgreich durchführen zu können. Die Sicherheit des One-Time-Pad dagegen beruht auf der einmaligen Verwendung des Schlüssels sowie der zufälligen Wahl des verwendeten Schlüssels. Es kann auch mit beliebig hoher Rechenleistung nicht gebrochen werden.

Nachteile

Das One-Time-Pad benötigt einen Schlüssel, der genauso lang ist wie die Nachricht selbst. Um beispielsweise die gesamten Daten eines Festplattenlaufwerks zu verschlüsseln, ist ein zweites Festplattenlaufwerk (mit mindestens gleicher Größe) zur Speicherung des Schlüssels nötig. Will man Nachrichten mit dem OTP-Verfahren verschlüsselt übertragen, muss der Schlüssel über einen anderen (sicheren) Kanal übertragen werden als die Nachricht. Beispielsweise kann eine CD mit Schlüsseln durch einen Boten überbracht werden, während die damit verschlüsselten Nachrichten über das Internet übertragen werden. Eine moderne Alternative, um auf einen Boten zu verzichten, stellt die Quantenkryptografie dar. Dabei handelt es sich um ein Verfahren, mit dem räumlich getrennte Sender und Empfänger einen gemeinsamen Schlüssel geheim erzeugen können.

Ein eher theoretisches Problem besteht darin, dass die einzelnen Zeichen des Schlüssels vollkommen zufällig und unabhängig erzeugt werden müssen. In idealer Weise kann das nur durch einen nichtdeterministischen

physikalischen Zufallszahlengenerator erreicht werden. In der Praxis verzichtet man häufig darauf und gibt sich mit mehr oder weniger guten Zufallsgeneratoren zufrieden. Handelsübliche Rechner sind nicht in der Lage, wirklich zufällige Schlüssel zu erzeugen.

Aufgrund des hohen logistischen Aufwands kann sich das One-Time-Pad für die Verschlüsselung in größeren Kommunikationsnetzen nicht durchsetzen. Für die zweiseitige geheime Kommunikation ist es in puncto Sicherheit jedoch nach wie vor die erste Wahl.

Literatur

- Friedrich L. Bauer: *Entzifferte Geheimnisse: Methoden und Maximen der Kryptographie*. Springer, Berlin 2000 (3. Aufl.). ISBN 3-540-67931-6
- Robert Louis Benson: *The VENONA Story*. Center for Cryptologic History, NSA, Fort Meade, USA. Abgerufen: 5. Januar 2011. PDF; 0,8 MB [4]
- Rudolf Kippenhahn: *Verschlüsselte Botschaften: Geheimschrift, Enigma und Chipkarte.* Rowohlt, Hamburg 1999. ISBN 3-499-60807-3
- Dirk Rijmenants: *Is One-time Pad History?* Cipher Machines & Cryptology, 2010. Abgerufen: 5. Januar 2011. PDF; 0,1 MB [5]
- Dirk Rijmenants: *The Complete Guide to Secure Communications with the One Time Pad Cipher.* Cipher Machines & Cryptology, 2010. Abgerufen: 5. Januar 2011. PDF; 0,2 MB [6]

Weblinks

- Häufige Fragen zum OTP (Englisch) mit Foto eines authentischen russischen Einmalblocks [7]
- Erläuterungen zum OTP (Englisch) mit Bildern, Übungen und Programmen [8]
- Graphische Variante des OTP als Webdemo [9]
- Graphische Variante des OTP als Windows-Programm [10]
- Zufallszahlengenerator zur Erzeugung von OTPs (Englisch) [11]

Belege

[1] Simon Singh: *Geheime Botschaften*. Carl Hanser Verlag, München 2000, S. 178. ISBN 3-446-19873-3

[2] Johannes A. Buchmann: *Introduction to Cryptography*. Springer, 2001, 4.

[3] Claude Shannon: *Communication Theory of Secrecy Systems.* (http://netlab.cs.ucla.edu/wiki/files/shannon1949.pdf) Bell System Technical Journal, Vol 28, 1949 (Oktober), pp. 656–715

[4] http://www.nsa.gov/about/_files/cryptologic_heritage/publications/coldwar/venona_story.pdf

[5] http://users.telenet.be/d.rijmenants/papers/is_one_time_pad_history.pdf

[6] http://users.telenet.be/d.rijmenants/papers/one_time_pad.pdf

[7] http://www.ranum.com/security/computer_security/papers/otp-faq/

[8] http://users.telenet.be/d.rijmenants/en/onetimepad.htm

[9] http://blackbox.ethz.ch/onetimepad

[10] https://sites.google.com/site/martinmeise/home/one-time-pad

[11] http://users.telenet.be/d.rijmenants/en/numbersgen.htm

RC4

RC4, *ARC4* oder *Arcfour* ist eine Stromverschlüsselung, die mit Standards wie HTTPS, SSH 1 und WEP bzw. WPA weite Verbreitung gefunden hat.

RC4 (*Ron's Code 4*) wurde 1987 von Ronald L. Rivest entwickelt, ist eine Marke von RSA Security und offiziell geheim (Security by Obscurity). *ARC4* (*Alleged* RC4) oder *Arcfour* geht auf eine anonyme Veröffentlichung von Quelltext im Jahr 1994 zurück und ist Open Source.

Beschreibung

Eine Zufallsfolge wird aus einem nur einmalig zu verwendenden Schlüssel erzeugt. Der Klartext wird Bit für Bit per XOR mit der Zufallsfolge verknüpft, um die Daten zu verschlüsseln.

Der Zufallsgenerator verwendet eine so genannte S-Box, eine zufällig gewählte Permutation oder Substitution der Zahlen 0 bis 255. Die S-Box wird in einem ersten Schritt aus dem geheimen Schlüssel berechnet und anschließend zur Berechnung der Zufallsfolge verwendet. Nach jedem Berechnungsschritt werden zwei Werte der S-Box vertauscht.

Die Sicherheit eines solchen Verfahrens ist nur gewährleistet, wenn sich die Zufallsfolge nicht wiederholt. Daher darf der Schlüssel bzw. das Passwort nur einmalig verwendet werden. Für die Besetzung der S-Box und die Werte zweier weiterer Variablen gibt es ca. 2^{1684} Möglichkeiten. Was einer Schlüssellänge von 210 (1684/8) Zeichen entsprechen würde. Nach dem Geburtstagsparadoxon ist zu erwarten, dass es Schlüssel mit einer Schlüssellänge von ((1684/2)/8) 105 Zeichen gibt, die identische Permutationen der S-Box erzeugen. Bekannt sind mittlerweile mindestens zwei 24 Zeichen (192 Bit) lange Schlüssel, die zur gleichen Permutation der S-Box führen. Damit gibt es zwei verschiedene Schlüssel, die zur gleichen Verschlüsselungsfolge führen[1].

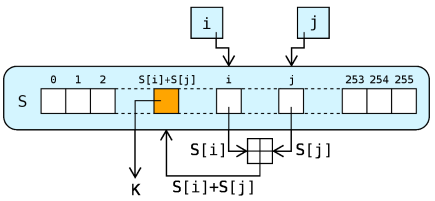
Der Algorithmus ist sehr einfach mit praktisch jeder Hard- und Software zu implementieren und sehr effizient berechenbar. Die Verarbeitungsgeschwindigkeit des DES ist selbst in kleinsten Mikrocontrollern meist ausreichend. Der Schlüssel beim DES und anderen Blockchiffren wie dem AES (CBC-Mode) kann im Gegensatz zum RC4 auch mehrfach verwendet werden. Vergleichbare Stromchiffren wie SEAL sind vergleichbar schnell und gelten als sicher.

Im WEP wurde der einmalige Schlüssel durch einfaches Zusammensetzen eines festen geheimen Schlüssels und eines Session Key bestimmt. In diesem Fall ist es jedoch möglich, den festen geheimen Schlüssel abzuleiten. Falls der Schlüssel mit einer Hash-Funktion quasi zufällig gewählt wird, kann der RC4 aber weiterhin als sicher betrachtet werden.

Bei Microsoft-Windows-Systemen, welche an eine NT-Domäne angebunden sind, wird das Anmeldepasswort, welches der Benutzer in der GINA-Oberfläche eingibt, nach vorangegangener Aushandlung eines Schlüssels per RC4-HMAC verschlüsselt und durch einen Kerberos-Frame an den Server übertragen. Die Aushandlung des Schlüssels findet während der Meldung „Netzwerkverbindungen werden vorbereitet“ statt.

Algorithmus

Kern des Verfahrens ist die so genannte S-Box, eine zufällige Vertauschung oder Permutation des Standard-Alphabets (Byte-Werte 0-255). Mittels der S-Box wird eine Zufallsfolge erzeugt, die Bit für Bit durch Addition modulo 2, auch XOR-Verknüpfung genannt, mit dem Nachrichtenstrom verknüpft wird. Die S-Box wird zunächst als identische Abbildung vorbesetzt, so dass $S[i] = i$ für i=0 bis 255 gilt.



Die Nachschlagephase von RC4 bei einem Alphabet der Größe eines Bytes. Das Ausgabebyte wird von den Werten $S[i]$ und $S[j]$ ausgewählt, diese werden addiert modulo 256 gerechnet.

Die initiale Belegung der S-Box kann mit dem folgenden Pseudo-Code beschrieben werden. Die S-Box wird dabei aus dem Schlüssel k der Länge L Byte berechnet:

```
k[]: gegebene Schlüssel-Zeichenfolge der Länge 5 bis 256 Byte
L := Länge des Schlüssels in Byte
s[]: Byte-Vektor der Länge 256
Für i = 0 bis 255
  s[i] := i
j := 0
Für i = 0 bis 255
  j := (j + s[i] + k[i mod L]) mod 256
  vertausche s[i] mit s[j]
```

Die anschließende Berechnung der Zufallsfolge erfolgt analog:

```
klar[]: gegebene Klartext-Zeichenfolge der Länge X
schl[]: Vektor zum Abspeichern des Schlüsseltextes
i := 0
j := 0
Für n = 0 bis X-1
  i := (i + 1) mod 256
  j := (j + s[i]) mod 256
  vertausche s[i] mit s[j]
  zufallszahl := s[(s[i] + s[j]) mod 256]
  schl[n] := zufallszahl XOR klar[n]
```

Zum Entschlüsseln verwendet man den gleichen Algorithmus, wobei der Schlüsseltext anstelle des Klartextes eingegeben wird. Zwei XOR-Verknüpfungen mit derselben Zufallszahl heben sich gegenseitig auf, und als Ausgabe entsteht wieder der Klartext.

Nennenswerte Schwächen des Verfahrens liegen allein in der Initialisierung der S-Box begründet. Bei einer zufällig gewählten S-Box als Ausgangspunkt der Berechnung ist das Verfahren trotz seiner Einfachheit gegenwärtig (Jan. 2011) praktisch nicht zu knacken.

Sicherheit

RC4 liefert sehr effizient eine Pseudozufallszahlenfolge, die mit dem Nachrichtenstrom durch XOR verknüpft wird (bei der zweiten XOR-Verknüpfung erscheint wieder das Original). Die Folge kann im Prinzip beliebig lang gemacht werden, in der Praxis der Nachrichtenübertragung treten jedoch Synchronisationsverluste auf, so dass neu aufgesetzt werden muss. Der dabei verwendete neue Schlüssel besteht aus einem offen übertragenen Teil und einem geheimen Teil. Der offene Teil sorgt für eine andere Zahlenfolge, der geheime Teil garantiert die Sicherheit.

In der WEP-Anwendung wird die Folge ab dem ersten Byte verwendet. Genau für dieses Byte besteht aber eine Simulationsmöglichkeit: die Initialisierung des Algorithmus wird nachvollzogen und dabei das nächste Schlüsselbyte des geheimen Schlüssels „geraten“. Die Wahrscheinlichkeit, dass das nun auszugebende erste Byte der Zahlenfolge in der weiteren Initialisierung nicht mehr verändert wird, ist relativ groß. Hat man genügend viele Neusynchronisationen mit neuen offenen Schlüsselteilen aufgezeichnet, so fällt ein richtig geratenes Schlüsselbyte statistisch auf. Man fährt so fort, bis alle geheimen Schlüsselbytes ermittelt sind. Das erste Byte der Zahlenfolge erhält man aus dem Datenstrom, da die Klartexte der ersten Bytes der Datagramme aus den Datenübertragungsprotokollen bekannt sind.

Der Angriff gelingt nur, wenn das erste Byte zur Verfügung steht. Das Verwerfen der ersten fünf bis zehn Bytes ist für die Wiederherstellung der Sicherheit ausreichend. Bei der Neuauflage der Protokolle (WPA) wurde aber auch direkt ein anderes Verschlüsselungsverfahren eingeführt, da die XOR-Stromverschlüsselung weitere Angriffsmöglichkeiten bietet: ist der Aufbau eines Datensatzes bekannt, besteht nämlich die Möglichkeit der Fälschung. Beispielsweise werden bei Banküberweisungen Kontonummer, Betrag usw. immer an der gleichen Position stehen, Kontonummern und Beträge sind relativ systematisch aufgebaut. Durch ein einfaches XOR mit entsprechend berechneten Masken lassen sich Beträge und Kontonummern verändern (z. B. 1000 addieren), wobei der Betrug nur mit einer gewissen Wahrscheinlichkeit auffällt. Allerdings besteht für den Angreifer lediglich die Möglichkeit der ungezielten Fälschung. Er kann kein bestimmtes Zielkonto angeben und auch den verschlüsselten Text nicht lesen. Diese Art des Angriffs existiert bei Blockchiffren wie DES oder AES in ähnlicher Form, denn je nachdem welcher Modus gewählt wurde lassen sich die Datenblöcke löschen, kopieren oder austauschen. Beim einen Modus sind die Blöcke voneinander abhängig und bei dem anderen nicht.

Weblinks

- Implementierung in Python [2]
- Überblick über Arbeiten bezüglich der Sicherheit von RC4 [3], englisch
- Original-Post im Usenet des damals noch geheimen RC4-Algorithmus [4], englisch
- Weaknesses in the Key Scheduling Algorithm of RC4 [5] von Scott Fluhrer, Itsik Mantin, und Adi Shamir, englisch (PDF-Datei; 297 kB)
- RSA Security Response to Weaknesses in Key Scheduling Algorithm of RC4 [6]
- *A Stream Cipher Encryption Algorithm "Arcfour"* [7] (Internet Engineering Task Force Network Working Group 1997)
- RFC 4345 *Improved Arcfour Modes for the Secure Shell (SSH) Transport Layer Protocol* [8]

Quellen

[1] Mitsuru Matsui. *Key Collisions of the RC4 Stream Cipher*. Fast Software Encryption 2009, in Lecture Notes in Computer Science Nummer 5665. Seiten 38–50. 2009. Springer Verlag. (Präsentation (engl.) (http://www.iacr.org/workshops/fse2009/slides/2302_1015_Matsui.pdf))
[2] http://code.activestate.com/recipes/576736-rc4-arc4-arcfour-algorithm
[3] http://www.wisdom.weizmann.ac.il/~itsik/RC4/rc4.html
[4] http://groups.google.com/groups?selm=sternCvKL4B.Hyy%40netcom.com
[5] http://www.drizzle.com/~aboba/IEEE/rc4_ksaproc.pdf
[6] http://www.rsa.com/rsalabs/node.asp?id=2009
[7] http://tools.ietf.org/html/draft-kaukonen-cipher-arcfour-01
[8] http://tools.ietf.org/html/rfc4345

Diskussion:RC4

Hallo zusammen,

warum gilt RC4 als gebrochen? Ist eine Schwachstelle im Algoritmus entdeckt worden? Meines Wissens nach, ist der Algorithus immer noch als sicher einzustufen. Oder liege ich damit falsch?

MFG Alex

RC4 ist gebrochen. Als erstes wurde der Key-Schedule von RC4 gebrochen. Fängt man die ersten 255 Byte oder so ab, erlauben diese Rückschlüsse auf die S-Boxen der Chiffre und damit auf den Zustand von RC4. Damit kann ein zweiter RC4 initialisiert werden und dieser liefert dann fortlaufend die gleichen Ergebnisse wie der erste. Desweiteren ist RC4 biased. Bei einer sicheren Chiffre sollte die Wahrscheinlichkeit, dass ein Byte x and stelle n steht gleich 1/255 sein. Bei RC4 weichen diese Wahrscheinlichkeiten für einige signifikante Bytes ab. Dazu kommt auch noch, dass die ersten Bytes sehr stark vom verwendeten Schlüssel abhängen, sprich, kennt man die ersten paar Bytes des Schlüsselstromes, kann man schon sehr gute Rückschlüsse auf den Schlüssel ziehen. Ich verweise dich hier mal auf Klein's Attack, Fluhrer, Mantin and Shamir attack, Biased Outputs of the RC4.
--Psygate 10:20, 30. Okt. 2009 (CET)

RC4 ist nicht wirklich gebrochen

Hallo! in RC4 wurde eine Schwachstelle gefunden, die den Autor wohl dazu bewegte zu schreiben, RC4 sei gebrochen.

Und zwar liefern die ersten paar Ausgaben Rückschlüsse auf den Zustand der S-Box zurück, was u.U. zur Kompromitierung des Algorithmus führen könnte. Das lässt sich allerdings ganz einfach umgehen indem man die ersten 256 Ausgaben verwirft.

Weiterhin ist es natürlich empfehlenswert dem Schlüssel eine zufällige beigabe ("Salt") hinzuzufügen, wodurch auch bei gleichem Schlüssel eine unterschiedliche Zufallsfolge austritt.

RC4 ist quasi gebrochen. Selbst wenn mand ie ersten 256 Bytes des Schlüsselstroms verwirft, ist es immer noch möglich durch statistische Attacken den Schlüssel zu erahnen.

-- Psygate 10:20, 30. Okt. 2009 (CET)

Was heißt denn gebrochen ?

Wenn der Entwickler oder Verkäufer behauptet der Algorithmus hätte eine bestimmte Eigenschaft, könnte man sagen er sei gebrochen, falls der Algorithmus diese Eigenschaft nachweislich nicht hat. Es gibt aber wenig klare Aussagen zu den Eigenschaften des RC4.

In jedem Fall darf der Schlüssel bei RC4 nur einmalig verwendet werden und wie bei WEP gezeigt, dürfen die einmaligen Schlüssel nicht durch einfaches Zusammensetzen mit einem festen geheimen Schlüssel gebildet werden. Unter diesen Voraussetzung dürfte der RC4 weitgehend sicher sein. Dies ist allerdings auch kein Kunststück.

Gebrochen bedeutet im kryptographischen Sinne eine Attacke, die schneller ist als die Exhaustion-Search (Brute Force).

-- Psygate 10:20, 30. Okt. 2009 (CET)

SSL und RC4

Wird nicht bei SSL sehr häufig RC4-128 eingesetzt? Das wäre dann ja sicherheitsmäßig äußerst "suboptimal". --Captain Crunch 14:45, 9. Okt 2005 (CEST)

Das SSL Protokoll ist als sicher anzusehen, obwohl RC4 suboptimal ist. [6]

--vPsygate 10:20, 30. Okt. 2009 (CET)

Bedeutung von RC

Laut FAQ Nr. 3 auf der Homepage von Ron Rivest [[1]] bedeutet RC "Ron's Code" und nicht "Ron's Cipher".

Einleitung: DES?!

Im 5. Absatz der Einleitung ist plötzlich – und für mich ziemlich unmotiviert / deplaziert – von DES die Rede bzw. die Schreibe; ich finde, das sollte jemand ausbügeln (d.h. *entweder* Erwähnung von DES ganz vermeiden, *oder* Zusammenhang / Bezug zu RC4 erklären), ich fühle mich aber nicht dazu berufen. — Nol Aders 14:29, 30. Mär. 2007 (CEST)

Korrelationsangriff von Klein

Sobald das Papier [[2]] von Andreas Klein in der Zeitschrift "Designs, Codes and Cryptography" veröffentlicht wird, sollten wir einen Abschnitt und eine Literaturangabe dazu einfügen. Der neue Angriff ist allgemeiner als der von FMS und deshalb erwähnenswert. --HeikoStamer 21:15, 5. Apr. 2007 (CEST)

> Der Angriff von Klein ist mittlerweile veröffentlicht [[3]] und hat alle Prüfungen überstanden. Hiermit ist es bei beliebiger Wahl des Schlüssels möglich, RC4 rein statistisch anzugreifen, die Anzahl der benötigten beobachteten Sitzungen liegt je nach Angriffsversion unter 100000. Auch ein verwerfen der ersten 256 Ausgabewerte reicht nicht mehr, um den Algorithmus zu schützen. Dies sollte in dem Artikel unbedingt erwähnt werden. --81.210.194.83 14:13, 13. Jun. 2009 (CEST)

"Sicherheit"

Zitat: Durch ein einfaches XOR mit entsprechend berechneten Masken lassen sich Beträge und Kontonummern verändern (z.B. 1000 addieren), wobei der Betrug nur mit einer gewissen Wahrscheinlichkeit auffällt. Allerdings besteht für den Angreifer lediglich die Möglichkeit der ungezielten Fälschung. Er kann kein bestimmtes Zielkonto angeben und auch den verschlüsselten Text nicht lesen.

Zitat Ende

1. **1000 zu addieren** ist eine gezielte Fälschung, da sie sich nur unter Kenntnis des tatsächlich veranlassten Betrags zu einer Bitmaske umformen lässt, die dann ge-XOR-t den Betrag um 1000 erhöht. Als Beispiel ist es also falsch Kennt man den Klartext, so kann man WEP-Pakete an beliebiger Stelle gezielt verändern.

2. Da CRC auch auf XOR basiert, kann ein MIM-Angreifer diesen Teil des Paketes in jedem Fall konsistent zu seinen anderen Modifikationen ändern. Es besteht also auch keine "**gewisse Wahrscheinlichkeit**" im WEP-Verfahren, mit der eine Veränderung bemerkt werden kann, sondern höchstens ein Plausibilitätscheck auf derAnwendungsebene.

Außerdem möchte ich die Frage aufwerfen, was die WEP-Schwachstelle in diesem Detailgrad in einem Artikel zu RC4 zu suchen hat. Thematisch sollte das m.E. hier stehen: Wired_Equivalent_Privacy#Schwachstellen

Veralgemeinerung des Pseudocodes

Ich wollte anfragen, ob der Pseudocode nicht veralgemeinert werden muss um allgemein gültig zu werden!

Die Zahl 256 erscheint hier als 'Magic-Number'

Der Algorithmus funktioniert aber für jeden beliebigen Zeichensatz (nicht nur für Zeichen (Bytes) der Grösse eines Oktets)

Folgender Vorschlag:

Initialisierung:

```
Setze j = 0
Setze k[] = (Schlüssel-Zeichenfolge beliebiger Länge > 0)
Setze s[] = (sBox-Zeichenfolge in der sämtliche möglichen Zeichen Platz finden)
Für i= 0 bis LängeVon(s)
  s[i] = i
j = 0
Für i = 0 bis LängeVon(s)
  s[i] = i
  j = (j + s[i] + k[i mod LängeVon(k)]) mod LängeVon(s)
  vertausche(s[i],s[j])
```

und

Code:

```
Setze i = 0 und j = 0
Wiederhole für alle Zeichen des Nachrichtenstroms
  i = (i + 1) mod LängeVon(s)
  j = (j + s[i]) mod LängeVon(s)
  vertausche(s[i],s[j])
  Zufallszeichen=s[(s[i]+s[j]) mod LängeVon(s)]
  ChiffreZeichen=Zufallszeichen XOR (nächstes Zeichen des Nachrichtenstroms)
```

--Reto.koenig 11:01, 22. Jul. 2008 (CEST)

Beim Durchforsten der anderen Artikel ist mir der Russische ins Auge gesprungen, welche als mod stets 2^n wählte!

Somit würde folgende Anmerkung in den Hauptartikel einfliessen:

```
Die verwendeten Zeichen basieren auf einem Zeichensystem der Grösse 2^n (so z.B. 2^8 für ein Byte)
```

Und der Pseudocode würde folgendermassen lauten:

Initialisierung:

```
Setze j = 0
Setze k[] = (Schlüssel-Zeichenfolge beliebiger Länge > 0 )
Setze s[] = (sBox-Zeichenfolge der Länge 2^n)
Für i= 0 bis LängeVon(s)
  s[i] = i
j = 0
Für i = 0 bis LängeVon(s)
  s[i] = i
  j = (j + s[i] + k[i mod LängeVon(k)]) mod LängeVon(s)
  vertausche(s[i],s[j])
```

und

Code:

```
Setze i = 0 und j = 0
Wiederhole für alle Zeichen des Nachrichtenstroms
  i = (i + 1) mod LängeVon(s)
  j = (j + s[i]) mod LängeVon(s)
  vertausche(s[i],s[j])
  Zufallszeichen=s[(s[i]+s[j]) mod LängeVon(s)]
  ChiffreZeichen=Zufallszeichen XOR (nächstes Zeichen des Nachrichtenstroms)
```

Falls also niemand einen Einwand erhebt, werde ich ab dem 30.07.2008 den Hauptartikel dahingehend ändern.

--Reto.koenig 11:34, 22. Jul. 2008 (CEST)

> Rein mathematisch gesehen benötigt RC4 kein Alphabet mit 2^n Zeichen. Dies ist lediglich daher sinnvoll, dass im Binärsystem Alphabete dieser Art besonders einfach darstellbar sind und somit optimal für effiziente Algorithmen geeignet. In der allgemeinen Beschreibung des Algorithmus kann denke ich allerdings trotzdem ein allgemeines n stehen, vielleicht mit dem Hinweis, dass in den gängigen Anwendungen n=256 gewählt wird. --81.210.194.83 14:18, 13. Jun. 2009 (CEST)

RC4 die fast perfekte Verschlüsselung

Um so länger ich darüber nachdenke, um so mehr muss ich mit dem Kopf schütteln. Dieser RC4-Algorithmus ist ja nun tatsächlich extrem simpel, aber trotzdem lässt sich damit eine nahezu perfekte Verschlüsselung realisieren. Was sind die Anforderungen an eine perfekte Verschlüsselung?

- Die Geheimhaltung ist immer gewährleistet, sofern der Schlüssel oder das Passwort nicht vollständig bekannt ist.
- Selbst wenn das Verfahren öffentlich bekannt ist und sogar wenn größere Mengen an Klartext und zugehörenden Chiffren vorliegen, bleibt die Geheimhaltung gesichert.
- Das Verfahren ist öffentlich von vielen Experten geprüft.
- Die Implementierung ist mit praktisch jeder Soft- und Hardware möglich.
- Der Resourcenverbrauch, Arbeitsspeicher, der Umfang des Quelltexts und die Laufzeiten minmal sind und der Quellcode so überschaubar ist, dass die Implementierung leicht geprüft werden kann.

RC4 erfüllt all diese Anforderungen, falls das Passwort einmalig verwendet wird. Zum Beispiel als:

```
Passwort_Neu = h(Passwort + laufende Nummer) oder
```

```
Passwort_Eeu = h(Passwort + Tagesdatum) mit der Hashfunktion h gleich SHA-1 oder SHA-256
```

gewählt wird.

Stromchiffre mit Hashfunktion

Eine Stromchiffre auf Basis von SHA mit

```
Zufall( counter ) = SHA-X ( Passwort + counter )
```

mit X größer oder gleich 1 unterscheidet sich praktisch nicht von einem One-Time-Pad, wenn das Passwort nur zur Berechnung einer einzigen Zufallsfolge benutzt wird. Dies folgt aus den Eigenschaften der SHA-Algorithmen. Das Argument der Hashfunktion SHA-X wiederholt sich unter diesen Voraussetzung niemals. Da sogenannte Kollisionen für SHA-X praktisch nicht berechnet werden können wiederholt sich auch die Zufallsfolge nicht und es ist ausgeschlossen das Passwort aus der Zufallsfolge zu bestimmen. Diese Verfahren können praktisch nicht verbessert werden. Sie sind schneller als die Datenübertragung, so sicher wie das Passwort, leicht zu implementieren, auf jeder Hard- und Software. Was will man mehr?

Auf Details der Implementierung etwa bei der Berechnung des Zählers

```
counter = start + step
```

können die Werte für Startwert und Schrittweite nahezu beliebig gewählt werden. Diese Parameter könnten auch abhängig vom Passwort gewählt werden. Die Zufallsfolge könnte auch in Abhängigkeit der bereits berechneten Chiffre berechnet werden.

```
Zufall( counter ) = SHA-X ( Passwort + counter + chipher (counter - 1) )
```

Jeder kann auch seine persönliche Chiffre mit einem persönlichen Schlüssel key benutzen.

```
Zufall( counter ) = SHA-X ( key + Passwort + counter + chipher (counter - 1) )
```

Was will man also mehr? --84.59.236.35 13:05, 23. Nov. 2008 (CET)

Stimmt - das Verfahren ist nach menschlichem Ermessen sicher!

Zitat aus Stromchiffre:

> Bei hohen Sicherheitsanforderungen ist eine Grundforderung, dass sich der Schlüsselstrom nicht wiederholen darf. Dies kann beispielsweise dadurch erreicht werden, dass eine Einwegfunktion zur Erzeugung des Schlüsselstroms implementiert wird und kein zyklischer Pseudorandomgenerator. Stromchiffrierungen können dann nahezu die Sicherheit vom One-Time-Pad Verfahren erreichen.

Durch die Verwendung des Zählers (counter) ist der „Pseudorandomgenerator“ definitiv nicht zyklisch. Es kann definitiv ausgeschlossenen werden, dass sich Werte des Zählers und damit der Argumente der Hashfunktion (Einwegfunktion) wiederholen. Der Zähler kann als Zustandsgröße des „Pseudorandomgenerator“ betrachtet werden, die sich niemals wiederholen kann. Die Sicherheit erreicht nahezu die Sicherheit des One-Time-Pad also nahezu beweisbare absolute Sicherheit. In der Praxis bedeutet dies, das Verfahren ist so sicher wie die Geheimhaltung der Passworte. Dies gilt auch für RC4 falls das Passwort als

```
password ( counter/date ) = SHA-X ( master password + counter/date )
```

berechnet wird. --88.68.100.230 20:48, 24. Nov. 2008 (CET)

Was ist daran so merkwürdig?

Die Tatsache dass es längst ein Verfahren, eigentlich eine ganze Menge ähnlicher Verfahren zur Verschlüsselung gibt, die allen sinnvollen Anforderungen genügen. Wobei die Performance faktisch nur durch die Datenübertragung oder das Schreiben bzw. Lesen von einem Speichermedium begrenzt ist. Die Verfahren also faktisch nicht wesentlich verbessert werden können. Aber noch Anfang dieses Jahrtausends wurden Software-Implementierungen bestimmter Algorithmen unter Exportverbot gestellt, weil diese wundervollen Entwicklungen ungeheuer wertvoll und sogar kriegsentscheidend seien. Um die Jahrtausendwende erschien jede Menge Literatur zu dem Thema, eine merkwürdige Diskussion über „starke Kryptographie“ wurde angestoßen. Weltverbesserer führten einen heldenhaften Kampf gegen die Geheimdienste, um die Bürger dieser Welt mit Software für starke Kryptographie zu versorgen. Dann werden auch noch ganze Firmen gegründet, die mit dem Public Key Sicherheit ohne Vertrauen in irgend einen Menschen versprachen, die an der Börse für Milliarden gehandelt werden. Die wundersamen „Produkte“, die diese grandiosen Firmen vertreiben und die Programme der Weltverbesserer erweisen sich aber im Nachhinein als eher unsicherer im Vergleich mit den verschmähten zuverlässigen Algorithmen, die vermeintlich vom Geheimdienst manipuliert waren. Ja, da kann ich nur den Kopf schütteln. --84.59.133.33 14:15, 21. Nov. 2008 (CET)

Ist die Zufallsfolge von einer echten zu unterscheiden?

Würde RC4 eine *echte* Zufallsfolge benutzen, wäre RC4 ein One-Time-Pad und somit exakt mathematisch beweisbar absolut sicher. RC4 kann folglich nur geknackt werden, wenn die Pseudo-Zufallsfolge von einer echten Zufallsfolge zu unterscheiden ist. Um die Frage zu klären wie schwer dies ist habe ich einmal ein Statistik-Programm geschrieben:

#include<stdio.h>

main(){

```
unsigned char s[256];
char key[] = "Key";
char txt[] = "Plaintext";
int h[256];
```

```
int i,j;
int k=0;
```

```
for (i=0;i<256;i++) s[i] = i;
for(i=0,j=0;i<256;i++){
  unsigned char tmp;
  j = (j + s[i] + key[i % strlen(key)]) % 256;
  tmp = s[i];
  s[i] = s[j];
  s[j] = tmp;
}
i = j = 0;
do {
   unsigned char tmp;
   i = (i + 1) % 256;
   j = (j + s[i]) % 256;
   tmp = s[i];
   s[i] = s[j];
   s[j] = tmp;
```

```
   printf ("%02x",
          (unsigned char) txt[k] ^
          (unsigned char) ( s[(s[i] + s[j]) % 256] )
          );
}  while (txt[++k] != 0);
printf("\n");
```

```
for (k = 0; k < 256; k++) h[k] = 0;
```

```
for (k = 0; k < 1000000; k++){
   unsigned char tmp;
   i = (i + 1) % 256;
   j = (j + s[i]) % 256;
   tmp = s[i];
   s[i] = s[j];
   s[j] = tmp;
   h[ s[(s[i] + s[j]) % 256] ]++;
}
```

```
for (k = 0; k < 256; k++)
     printf("%02x %d\n", k, h[k]);
```

}

Erst wird für einen Testfall (siehe) die RC4-Verschlüsselung berechnet, um zu zeigen, dass der Algorithmus korrekt implementiert ist. Dann werden von der Pseudozufallsfolge noch eine Million weitere Werte (Bytes) berechnet, wofür ein üblicher Rechner weniger als eine Sekunde benötigt. Die Häufigkeit der einzelnen Byte-Werte wird berechnet und zum Schluss ausgegeben. Im Mittel liegt die Häufigkeit bei einer Million geteilt durch 256 und weicht nur in etwa einem Drittel der Fälle mehr als die Wurzel aus dieser Zahl von diesem Erwartungswert ab. Die Verteilung ist also nahezu perfekt statistisch verteilt. Zumindest wenn der Zufallsgenerator wirklich zufällig initialisiert wird, gibt es nicht den geringsten Anhaltspunkt für die Entschlüsselung.

Mein Fazit: Stromchiffren sind praktisch nicht praktisch nicht zu knacken, es sei denn ein Schlüssel würde mehrfach verwendet. Die Sicherheit entspricht der Geheimhaltung der Schlüssel oder Passworte. --84.59.226.31 12:44, 27.

Nov. 2008 (CET)

Welchen Bezug haben Deine Aussagen zum Inhalt des Artikels? Die Diskussionsseiten dienen AFAIK der **Diskussion des Artikelinhalts** und nicht der Theoriefindung. --HeikoStamer 22:37, 27. Nov. 2008 (CET)

Unter welchen Bedingungen der RC4 sicher ist, also nicht geknackt werden kann ist doch durchaus Gegenstand des Artikels. Weit überraschender als die Tatsache, dass RC4 bei nicht zufälliger Wahl der Passworte unsicher sein kann, ist dass dieser eher simple Algorithmus bei einer vollständig zufälligen Wahl des Passworts keine wirklich entscheidenden Schwächen besitzt, obgleich er über Jahrzehnte so genau studiert wurde, wie kaum sonst ein Verfahren. Auffallend ist dabei, dass es beim RC4 nicht mehrere Runden, also mit etwas geänderten Parametern mehrfach durchgeführte Berechnungen, gibt. Denkbar wäre es zum Beispiel nur jedes zehnte berechnete Byte zur Verschlüsselung zu nutzen oder die Summe modulo 256 von je zehn Bytes zu benutzen. Der Algorithmus wäre dann immer noch schnell genug. Auffallend ist auch, dass der Schlüssel nur in die einmalige Initialisierung der S-Box eingeht und danach nicht mehr gebraucht wird. Es wäre aber möglich die Addition des Schlüssels auch in der weiteren Berechnung fortzuführen. Ich könnte mir tausende Variationen des RC4 vorstellen, die höchst wahrscheinlich keinen negativen Effekt auf die Sicherheit hätten. Tatsächlich gibt es ja auch eine Reihe anderer Stromchiffren, die als mindestens so sicher wie der RC4 gelten. Wenn sich jetzt irgend jemand, zum Beispiel Herr Müller, irgend einen simplen Algorithmus ausdenkt und auch noch selbst programmiert, ist dies wahrscheinlich mindestens so sicher wie der RC4. Ja, selbst wenn er einen groben Schnitzer einbauen würde, solange der Algorithmus geheim bleibt, würde trotzdem niemand den Algorithmus knacken, weil es sich kaum lohnt den privaten Algorithmus von Herrn Müller zu knacken. Eine einfache Stromchiffre ist für den Hausgebrauch praktisch nicht zu knacken. Warum wird da ständig geforscht, wenn es die (fast) perfekte Verschlüsselung längst gibt. --88.68.102.151 19:10, 2. Dez. 2008 (CET)

Du bist herzlich eingeladen Deine Gedanken in der wissenschaftlichen Gemeinde zu diskutieren, z.B. indem Du Deine Argumente zur (Un-)Sicherheit von RC4 in schriftlicher Form bei einer Fachkonferenz einreichst. Wikipedia ist doch kein allgemeines Diskussionsforum. Es sollten nur wissenschaftlich gesicherte Aussagen im Artikel stehen. Ich sehe nicht, wo Deine Ausführungen zur Qualität des Artikels beitragen. --HeikoStamer 19:40, 6. Dez. 2008 (CET)

Es gibt doch schon unzählige Veröffentlichungen zur (Un-)Sicherheit von RC4. Ich denke es reicht wirklich! Der Witz bei der Sache ist doch, dass trotz all dieser eingehenden Untersuchung des RC4 nichts bekannt ist, wie er geknackt werden kann, wenn die Passwörter einmalig und auch nicht sehr ähnlich, wie etwa geheimnis1, geheimnis2, geheimnis3, … oder so ähnlich gewählt werden. Sollte etwa dieser RC4 wirklich extrem raffiniert sein, so dass gerade diese Stromchiffre nicht zu knacken ist? Unsinn – es gibt etliche gut untersuchte Stromchiffren, die mindestens ebenso sicher sind, wahrscheinlich aber noch viel schwerer zu knacken sind. Tatsächlich lassen sich quasi unendlich viele Zufallsreihen konstruieren. Zum einen könnte statt mit der Identität, s[i] = i, auch mit einer anderen Permutation begonnen werden, der RC4 also in einem kleinen Details variiert werden, etwa s[i] = (i + 7) mod 256, statt s[i] = i. Wobei statt der Sieben natürlich auch eine andere Zahl genommen werden könnte. Jetzt könnte aus zwei oder mehreren Zufallsreihen eine neue Reihe gewonnen werden, in dem die Zufallszahlen etwa per XOR verknüpft werden oder sonst irgendwie „addiert“ werden. Damit werden die Eigenschaften bezüglich Sicherheit, etwa die statistische Verteilung, keineswegs schlechter. Es gibt also unendlich, mal unendlich, mal unendlich viele Möglichkeiten, alle mindestens so sicher wie RC4. Noch besser, wenn niemand weiß, wie die Zufallsreihe genau berechnet wird, ist es noch viel schwerer das Verfahren zu knacken. Bleibt das Passwort geheim und kann nicht erraten

werden, ist eine solche Chiffre praktisch nicht zu knacken. --84.59.253.114 22:13, 6. Dez. 2008 (CET)

Stimmt, eine (geheime) Arcfour-Variante ist total einfach, aber quasi unknackbar. (*nicht signierter Beitrag von* 88.152.231.121 (Diskussion) 17:25, 25. Jan. 2012 (CET))

S-Box

Wieso von 0 bis 255? Hängt die Länge der S-Box nicht von der Länge des Schlüssels ab? Oder sind alle Schlüssel 8 Bit lang? (*nicht signierter Beitrag von* 89.53.202.139 (Diskussion | Beiträge) 14:35, 20. Jun. 2009 (CEST))

Die S-Box enthält 256 Bytes also 2048 Bit. Die Schlüssellänge ist beliebig bis zu 256 Bytes wählbar. Hmmm - stimmt, im Artikel steht es etwas anders. Eigentlich ist die Darstellung des Algorithmus Artikel nicht korrekt (vergleiche). Der dargestellte Algorithmus ist eine Verallgemeinerung. --OlbersD 16:32, 24. Jul. 2010 (CEST)

Genauer untersucht wurde der Algorithmus nur mit der Länge 256. Zumindest bei bei kleinerer Länge erscheint die Sicherheit eher fraglich. Mit größerer Länge wird der Speicherbedarf unnötig erhöht. Für die Länge 256 werden je 8 Bit, also ein Byte, gleichzeitig berechnet. Da die Daten üblicher Weise byteweise verarbeitet werden, ist damit die Implementierung besonders einfach. Die Länge zu variieren erscheint daher nicht unbedingt sinnvoll. --OlbersD 16:43, 24. Jul. 2010 (CEST)

Die Länge ist 256. [[4]] --88.152.231.121 23:58, 10. Mär. 2012 (CET)

Da ist er - der Python-Code RC4B

RC4B ist eine Arcfour-Variante hier implementiert mit Python. Mit *rc4E* kann verschlüsselt werden, mit *rc4D* entschlüsselt oder auch umgekehrt. Das Pass- oder Kennwort wird als dritter Parameter *key* (englisch Schlüssel) übergeben. Gelesen wird die Datei mit dem Dateinamen *fnam* und geschrieben wird eine Datei mit dem Namen *fnamo* ("o" wie output). Die Programme (Funktionsdefinitionen *def*) können einfach per Cut and Paste von hier kopiert werden.

Eine Installation der erforderlichen Python-Software erfolgt in Sekunden aus dem Internet. Auf diese Weise ist die Nutzung der Software auch im Internetcafé möglich. --88.152.231.121 17:44, 11. Mär. 2012 (CET)

def rc4E(fnam, fnamo, key):

```
  j = 0
  s = range(256)
  for r in range(3):
    for i in range(256):
        j = (j + s[i] + ord(key[i % len(key)])) % 256
        s[i], s[j] = s[j], s[i]
```

```
  i = j = 0

  f = open(fnam, 'rb')
  f.seek(0,2)
  flen = f.tell()

  cnt = 0
  while cnt < flen % 777:
     cnt = cnt + 1
     i = (i + 1) % 256
     j = (j + s[i]) % 256
```

```
      s[i], s[j] = s[j], s[i]
```

```
   f.seek(0)
   fo = open(fnamo, 'wb')
```

```
   cnt = 0
   while cnt < flen:
      buf = f.read(1000)
      out = []
      for char in buf:
         cnt = cnt + 1
         i = (i + 1) % 256
         j = (j + s[i]) % 256
         s[i], s[j] = s[j], s[i]
         outchar = ord(char) ^ s[(s[i] + s[j]) % 256]
         out.append(chr(outchar))
         j = j + outchar
      fo.write(str().join(out))
   fo.close()
   f.close()
   return 'ALLES IN BUTTER'
```

def rc4D(fnam, fnamo, key):

```
   j = 0
   s = range(256)
   for r in range(3):
     for i in range(256):
         j = (j + s[i] + ord(key[i % len(key)])) % 256
         s[i], s[j] = s[j], s[i]
```

```
   i = j = 0

   f = open(fnam, 'rb')
   f.seek(0,2)
   flen = f.tell()

   cnt = 0
   while cnt < flen % 777:
      cnt = cnt + 1
      i = (i + 1) % 256
      j = (j + s[i]) % 256
      s[i], s[j] = s[j], s[i]
```

```
   f.seek(0)
   fo = open(fnamo, 'wb')
```

```
cnt = 0
   while cnt < flen:
      buf = f.read(1000)
```

```
      out = []
      for char in buf:
         cnt = cnt + 1
         i = (i + 1) % 256
         j = (j + s[i]) % 256
         s[i], s[j] = s[j], s[i]
         outchar = ord(char) ^ s[(s[i] + s[j]) % 256]
         out.append(chr(outchar))
         j = j + ord(char)
      fo.write(str().join(out))
   fo.close()
   f.close()
   return 'ALLES IN BUTTER'
```

--88.152.231.121 11:40, 9. Mär. 2012 (CET)

Quellennachweise

[1] http://theory.lcs.mit.edu/%7Erivest/faq.html
[2] http://cage.ugent.be/~klein/RC4
[3] http://www.springerlink.com/content/6086867367826646/?p=b4452c6819f941a09a5092f1cc6f552a&pi=1
[4] http://www.rsa.com/rsalabs/node.asp?id=2250

Quelle(n) und Bearbeiter des/der Artikel(s)

One-Time-Pad *Quelle*: http://de.wikipedia.org/w/index.php?oldid=100288545 *Bearbeiter*: 217.7.105.xxx, 7645, A.Savin, Aka, Alnilam, An.ha, Andim, Anton, Badenserbub, Ben-Zin, Boguslaw Sylla, Bohem, Bücherwürmlein, ChristophDemmer, Ckler, Conversion script, Cubefox, DaB., Der Burgstädter, Dominiklenne, Ephraim33, FRR, Fab, Fabian Bieker, Fedi, FelixReimann, FloSch, Fsswsb, Fulbrich, Gamgee, Geiserich77, Gorgo, Graphikus, HaeB, Hephaion, Hozro, Jailbird, Jesi, KRYPTOCHEF Detlef Granzow (persönlich)!, KaiMeier, Karl-Henner, Kenansulayman, Kurt Jansson, LKD, Lars Regensburger, Le-max, Leberwurscht, Leithian, Leyo, Lukas Niemeyer, Mac, MarioS, Martin Bahmann, Martin Meise, Mathias Schindler, Maxus96, Mellebga, Mosmas, Murkel, Neg, Nevfennas, Nicole Brandt, OS, P. Birken, PaterMcFly, PeeCee, Pendulin, Peter Hassel, Philipendula, Pinoccio, Priwo, ProloSozz, R. Möws, Rdoering, Revolus, RokerHRO, Roo1812, Seewolf, Sg050, Shaun72, Siehe-auch-Löscher, Soulhack, Srittau, StG1990, Stargamer, Steevie, Stefan Birkner, Stern, T34, Tafkas, Tdlr, TheVi, Tönjes, Unix, Video2005, Vux, Wdwd, Wieser d, ZDragon, Zuse, 105 anonyme Bearbeitungen

RC4 *Quelle*: http://de.wikipedia.org/w/index.php?oldid=100772595 *Bearbeiter*: Achim Raschka, Aka, Aths, BenediktWildenhain, Brubacker, Cepheiden, Chrizz, Der Lange, Der fahrer, ErikDunsing, Etamatic123, Fabian Bieker, Fgb, Fomafix, Franc, Fsswsb, Gbrands, HardDisk, He3nry, Joachim T., JuTa, Karsten11, Lemmie, Megatherium, Michael Strunck, MichaelDiederich, Mkleine, Nol Aders, OS, Pinoccio, Priwo, Purodha, Reto.koenig, Roterraecher, Seb35, Skriptor, Steven Malkovich, SvenP, Testwikiii, ThePacker, Thomas Springer, Vsc, Wdwd, WikiDiskussion, Wurstwolf, 55 anonyme Bearbeitungen

Diskussion:RC4 *Quelle*: http://de.wikipedia.org/w/index.php?oldid=100751569 *Bearbeiter*: Captain Crunch, Fsswsb, HeikoStamer, Lipilly, Nol Aders, Psygate, Reto.koenig, WikiDiskussion, 31 anonyme Bearbeitungen

Quelle(n), Lizenz(en) und Autor(en) des Bildes

Datei:OneTimePadExcerpt.agr.jpg *Quelle*: http://de.wikipedia.org/w/index.php?title=Datei:OneTimePadExcerpt.agr.jpg *Lizenz*: GNU Free Documentation License *Bearbeiter*: User:ArnoldReinhold

Datei:Joseph Mauborgne.jpg *Quelle*: http://de.wikipedia.org/w/index.php?title=Datei:Joseph_Mauborgne.jpg *Lizenz*: Public Domain *Bearbeiter*: Original uploader was Nobunaga24 at en.wikipedia

Datei:PersonalStorageDevices.agr.jpg *Quelle*: http://de.wikipedia.org/w/index.php?title=Datei:PersonalStorageDevices.agr.jpg *Lizenz*: GNU Free Documentation License *Bearbeiter*: User:ArnoldReinhold

Datei:RC4.svg *Quelle*: http://de.wikipedia.org/w/index.php?title=Datei:RC4.svg *Lizenz*: Public Domain *Bearbeiter*: Traced by User:Stannered, original by User:Matt Crypto

Lizenz

Wichtiger Hinweis zu den Lizenzen

Die nachfolgenden Lizenzen bezieht sich auf den Artikeltext. Im Artikel gezeigte Bilder und Grafiken können unter einer anderen Lizenz stehen sowie von Autoren erstellt worden sein, die nicht in der Autorenliste erscheinen. Durch eine noch vorhandene technische Einschränkung werden die Lizenzinformationen für Bilder und Grafiken daher nicht angezeigt. An der Behebung dieser Einschränkung wird gearbeitet.
Das PDF ist daher nur für den privaten Gebrauch bestimmt. Eine Weiterverbreitung kann eine Urheberrechtsverletzung bedeuten.

Creative Commons Attribution-ShareAlike 3.0 Unported - Deed

Diese "Commons Deed" ist lediglich eine vereinfachte Zusammenfassung des rechtsverbindlichen Lizenzvertrages (http://de.wikipedia.org/wiki/Wikipedia:Lizenzbestimmungen_Commons_Attribution-ShareAlike_3.0_Unported) in allgemeinverständlicher Sprache.
Sie dürfen:

- das Werk bzw. den Inhalt **vervielfältigen, verbreiten und öffentlich zugänglich machen**
- **Abwandlungen und Bearbeitungen** des Werkes bzw. Inhaltes anfertigen

Zu den folgenden Bedingungen:

- **Namensnennung —** Sie müssen den Namen des Autors/Rechteinhabers in der von ihm festgelegten Weise nennen.
- **Weitergabe unter gleichen Bedingungen —** Wenn Sie das lizenzierte Werk bzw. den lizenzierten Inhalt bearbeiten, abwandeln oder in anderer Weise erkennbar als Grundlage für eigenes Schaffen verwenden, dürfen Sie die daraufhin neu entstandenen Werke bzw. Inhalte nur unter Verwendung von Lizenzbedingungen weitergeben, die mit denen dieses Lizenzvertrages identisch, vergleichbar oder kompatibel sind.

Wobei gilt:

- **Verzichtserklärung —** Jede der vorgenannten Bedingungen kann aufgehoben werden, sofern Sie die ausdrückliche Einwilligung des Rechteinhabers dazu erhalten.
- **Sonstige Rechte —** Die Lizenz hat keinerlei Einfluss auf die folgenden Rechte:
  - Die gesetzlichen Schranken des Urheberrechts und sonstigen Befugnisse zur privaten Nutzung;
  - Das Urheberpersönlichkeitsrecht des Rechteinhabers;
  - Rechte anderer Personen, entweder am Lizenzgegenstand selber oder bezüglich seiner Verwendung, zum Beispiel Persönlichkeitsrechte abgebildeter Personen.
- **Hinweis —** Im Falle einer Verbreitung müssen Sie anderen alle Lizenzbedingungen mitteilen, die für dieses Werk gelten. Am einfachsten ist es, an entsprechender Stelle einen Link auf http://creativecommons.org/licenses/by-sa/3.0/deed.de einzubinden.

Haftungsbeschränkung
Die „Commons Deed“ ist kein Lizenzvertrag. Sie ist lediglich ein Referenztext, der den zugrundeliegenden Lizenzvertrag übersichtlich und in allgemeinverständlicher Sprache, aber auch stark vereinfacht wiedergibt. Die Deed selbst entfaltet keine juristische Wirkung und erscheint im eigentlichen Lizenzvertrag nicht.

GNU Free Documentation License

Version 1.2, November 2002

Copyright (C) 2000,2001,2002 Free Software Foundation, Inc.

51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA

Everyone is permitted to copy and distribute verbatim copies

of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.
This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.
We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.
A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.
A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.
The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.
The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.
A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".
Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.
The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.
A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.
The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.
You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.
If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.
If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.
It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- **A.** Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- **B.** List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- **C.** State on the Title page the name of the publisher of the Modified Version, as the publisher.
- **D.** Preserve all the copyright notices of the Document.
- **E.** Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- **F.** Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- **G.** Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- **H.** Include an unaltered copy of this License.
- **I.** Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- **J.** Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- **K.** For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- **L.** Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- **M.** Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
- **N.** Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
- **O.** Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.
You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.
You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.
The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.
The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.
You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.
If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.
If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.
Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright (c) YEAR YOUR NAME.
>
> Permission is granted to copy, distribute and/or modify this document
>
> under the terms of the GNU Free Documentation License, Version 1.2
>
> or any later version published by the Free Software Foundation;
>
> with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.
>
> A copy of the license is included in the section entitled
>
> "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

> with the Invariant Sections being LIST THEIR TITLES, with the
>
> Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.
If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.