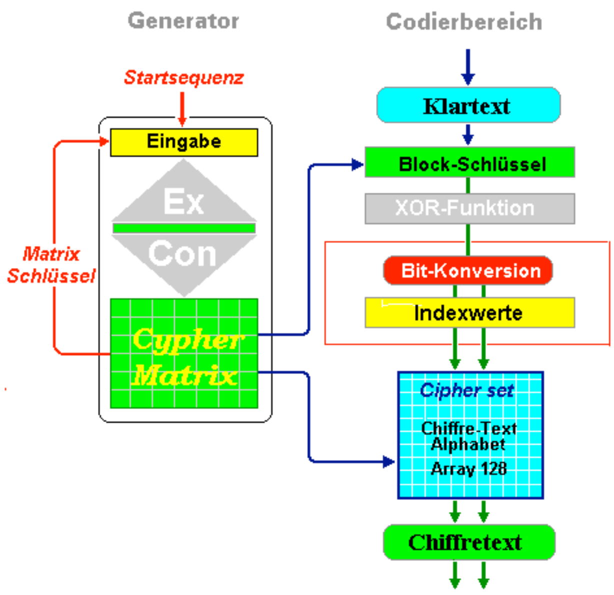
Generator

(Ernst Erich Schnoor)

>Generator.py< ist ein Python-Modul als Bestandteil der Basis-Funktion

>CypherMatrix®<

Die Funktion arbeitet in zwei Bereichen: Der **Generator** erzeugt die Steuerungsparameter und im **Codierbereich** wird der Chiffretext geschrieben:



Jeder Bereich kann auch einzeln verwendet werden. Das Programm benutzt eigene Zeichensätze mit jeweils 256 Zeichen wie folgt:

弓Ϊ㐱Ж닯斗T嘁ǟ疒б㗊낱哩큐廴_䝤XB팫냉ȡ彡勴苂目ϐ沦퀵ヨЂЍ矛兝㕗Q心手日냩퀅呿냻Ȇ꿓ӰȄå\欠厸M냒ȭ
ȯCO䍃J닱虫㔓냑넉넒弋IЏ匽T냇㖟攴呪R剛D谷JVSŘŸ꿀ㄡD넥HɆ꿍И匜歺文ПȮ耆넫EcZP殳CKÛ厚끲提Ьв
УШP皿BWⅡЁ꿈ɃΞ訛@ǘмGλ幺皮ǔ勸H制Ч暑戸广냄ҺOЫ叉ЌΛ叓日ưN劃Ц生吴月木YÂФFaЉƹ끮剃Oᆻ냤E?A
ØI넂ЭΣ냚C疋斤无б用ɛë끫哌ơΓĀДX龙止ЛБ敐æ矢ƳГ詑巾下ЁЯЋU냙支?勑彳A叓꾒[LЪ己Ƒ韧냅Є냊創ǖƳԁ干
êPĠ냂Ʈ냈냥田ǫ石ЊЮK方M냉ʈЙ퀴3T白Ф毋넆넿녀녁

Alternativ können auch andere Zeichensätze verwendet werden (z.B. indische/ bengalische, japanische Zeichen u.a.):

উ৪ॅ◌ौ৫ডাঋৄ৮ঐফ০ঠী◌ৈ৬গ্রআতঘলূজপছ২ঔড়ঘ্লৃঅঁমকভ১ধদঁয়র৯ষতণঅুঢ়ংক্রডূূূংস৯৲৬যু৯ছলঃ
ঞৗ⁄লডঙ্গযঁঙ্গচটৈ৬।বব্রঋজঈ॥দঔঘ৴অুবকটচ৭৫নগা◌◌◌াশঞ্জখ্বটণওখ৭ষছঔগঔফজ্রটাঅতথগাসরঠ
খঅি◌লর্ম◌ংফঢ়৭এঢ৩ভওড়অঁঘহহঁওঁঢথজহ২সঙ্ঘয়ফ৬রশঝঐঠঃনৃইভঞধ৩ঔৢষউদধম২নত৪রগব
ঢলৃঢচখযজঃঢরঅ৹০৭৫১ন।ৃলএঁঠবষ০◌ৈএঠমগোঅঃই৯৭বড৺ডপুকহআঐঋঢঙ্গ০টৗঠীঋৠলO_

Mit Eingabe einer Startsequenz von mind. 7 bis 80 Zeichen wird das Programm initialisiert. Zum Beispiel:

Steinbeisser im Steinhuder Meer

Die Startsequenz ist eine Folge von definierten Bytes (**ai**) der Länge (**n**). Jedes Zeichen a(i) der Eingabe (Startsequenz, Klartext) wird positionsgewichtet, indem sein Wert mit seiner Position **p**(i) innerhalb der Eingabe und der Zeit **t** multipliziert wird. Die Zeit ist nicht relevant, daher t = 1. Die Produkte addieren sich zum Zwischenwert (P):

$$P = \sum^{n} (a(i) + 1) * p(i) * t(i) \qquad t(i) = 1$$

Mit der Positionsgewichtung unterscheiden sich zwar die Zeichen a(i), aber Kollisionen als Folge des Austauschs von Zeichen innerhalb der Eingabe sind noch nicht ausgeschlossen. Zur Vermeidung von Kollisionen wird eine Trennkonstante **C**(k) eingeführt. Sie bestimmt sich allein aus der Länge (**n**) der Eingabe (Kollisionsfreiheit):

$$C(k) = n * (n-2) + code \qquad code = 1$$
$$C(k) = 2704$$

Unter Einbindung der Trennkonstante C(k) wird ein gewichteter Positionswert **H**(p) wie folgt errechnet:

$$H(p) = \sum_{i=1}^{n} (a(i)+1) * (C(k) + p(i))$$
$$H(p) = 14826868$$

Zur Erweiterung der Bestimmungsbasis wird eine Hashfunktionsfolge eingeführt, die die Eingangsdaten zu einer umfangreichen Folge in einem höherwertigen Zahlensystem (Basis **77**) expandiert. Der Vorgang ist die erste **Einweg-Funktion** des Verfahrens. Gleichzeitig ermittelt das Verfahren die Summe aller Ergebnisse als zusätzlichen Expositionswert **H**(k)

$$H(k) = \sum_{i=1}^{n} ((a(i)+1) * H(p) * p(i)) + p(i) + r(j)$$
$$H(k) = 2217447072145$$

Aus den Ergebnissen generiert das Programm die folgenden Steuerungsparameter:

Alpha	= (H(k) + H(p)) MOD 127)+1	Offset Runden-Alphabet:	61
Beta	= (H(k) MOD 169)+1	Offset Runden-Schlüssel:	127
Gamma	= (H(p)+code) MOD 196)+1	Offset Matrix-Schlüssel:	133

Um die Bestimmungsbasis auf dezimale Größen zurückzuführen wird eine Kontraktfunktion eingeführt. Für die Ziffern der Hashfunktionsfolge wird das Zahlensystem zur Basis **78** unterstellt. Jeweils drei Ziffern der Hashfunktionsfolge werden seriell durch Modulo 256 in dezimale Zahlen **0** bis **255** (ohne Wiederholung) zurückgerechnet. Die Ergebnisse werden in der BasisVariation gespeichert, einem Array von 16x16 Elementen. Die Elemente in ihrer Struktur ergeben die **CypherMatrix**. Nach der Wahrscheinlichkeits - rechnung entsteht die Wiederholung einer gleichen CypherMatrix erst in **256!**

(Fakultät) **= 8E+506** Fällen. Der Vorgang stellt die zweite **Einweg-Funktion** dar. Die Funktion ist nicht umkehrbar und eine Bestimmung der vorhergehenden Daten retrograd ist nicht möglich.

CypherMatrix

0	A 日 O 噘 Џ Б 疋 Σ 堤 呪 皮 ǔ ʔ O 斤 П 吴 Ȅ Q 支 月 觧 虫 Ʀ 냗 方 毋 Ɣ H M 3 彡	31
32	无 Ï 呸 냄 녖 歹 刬 넉 c Ʈ 剢 C 剻 干 ǭ 目 暑 叏 Y C C ɛ́ È 戶 G N 厶 넑 Ë 下 M B	63
64	꾁 龙 Й 叒 Ц ë 庈 P 꿓 ϐ æ T 辶 냉 叡 û Є T Э 幺 W 꿀 넾 Ƒ ê ȡ J 勴 넓 勅 넒 [	95
96	剈 Ф ơ ƛ в ţ 꿝 弋 攴 녀 Ъ 彳 Â 谷 꿞 疒 垊 R ǘ б 己 啕 ƌ Ш 曰 夛 I 文 劚 Њ 넓 A	127
128	I 矛 \ T 岔 E ʔ Ж 꾌 O 欠 U @ V 止 꿻 녻 꿓 넋 Г H 넛 꿨 Я У K Ǵ Ѝ 廾 D P ǻ	159
160	낛 Ǿ 广 木 Ў Ф 넓 꿈 넬 K ΄Y Ю 戳 劶 嘥 嗞 F ь Ч 녂 X 用 殳 녁 厚 田 訑 넉 J 꾌 沦 녈	191
192	匵 Ā м Л ȝ 냍 Ґ Ў 耆 矢 녁 Ђ 剮 白 Ɗ 生 넜 꿉 냅 Z И ᄏ 꾁 P 手 ћ Ќ E ǟ S 巾 犱	223
224	匹 Ƀ B Ơ i Ꞩ ʚ Ꞙ Ꮍ A J E ơ Ꮑ Ꭿ Ꞙ Γ & W ɗ ᑭ G M Ꮖ ᏽ Ơ H Ꮥ Ꮍ ϴ ţ G	255

Ab Positione Alpha = 61 werden 128 Elemente aus der CypherMatrix entnommen und als **Runden-Alphabet** definiert:

下 MB 꾁龙 Й 叒 Цë 庈 P 꿓 ϐæT辶냉叡 ûЄTЭ幺W 꿀넾ƑêȡJ 勴
넓勅넒[剈 ФơƛвţǍ 꿝弋攴녀Ъ彳Â谷꿞疒垊 Rǘб己啕 ƌШ曰夛 I文劚
Њ 넓 AI矛\T 岔 EʔЖ 꾌 O欠U@V止꿻녻꿓넋 ГH 넛꿨 ЯУKǴЍ廾
DPǻ 낛 Ǿ广木ЎФ 넓꿈넬 K΄YЮ 戳劶嘥嗞 FьЧ 녂 X用殳녁厚田訑넉 J

Der **Runden-Schlüssel** wird ab Position 127 aus der Cypher-Matrix geholt. In jeder Runde wird eine andere Position generiert.

넓 AI矛\T 岔 EʔЖ 꾌 O欠U@V止꿻녻꿓넋 ГH 넛꿨 ЯУKǴЍ廾DPǻ 낛 Ǿ广木ЎФ 넓꿈넬 K΄YЮ 戳劶嘥嗞 FьЧ 녂 X用殳녁厚田訑넉 J

Zur Initialisierung der nächsten Runde wird der **Matrix-Schlüssel** ab der Position 133 der aktuellen Cypher-Matrix entnommen und

岔 EʔЖ 꾌 O欠U@V止꿻녻꿓넋 ГH 넛꿨 ЯУKǴЍ廾DPǻ 낛 Ǿ广木ЎФ 넓꿈넬 K΄YЮ 戳劶嘥嗞 FьЧ 녂 X用殳녁厚田訑넉 J 꾌沦녈匵 Āм

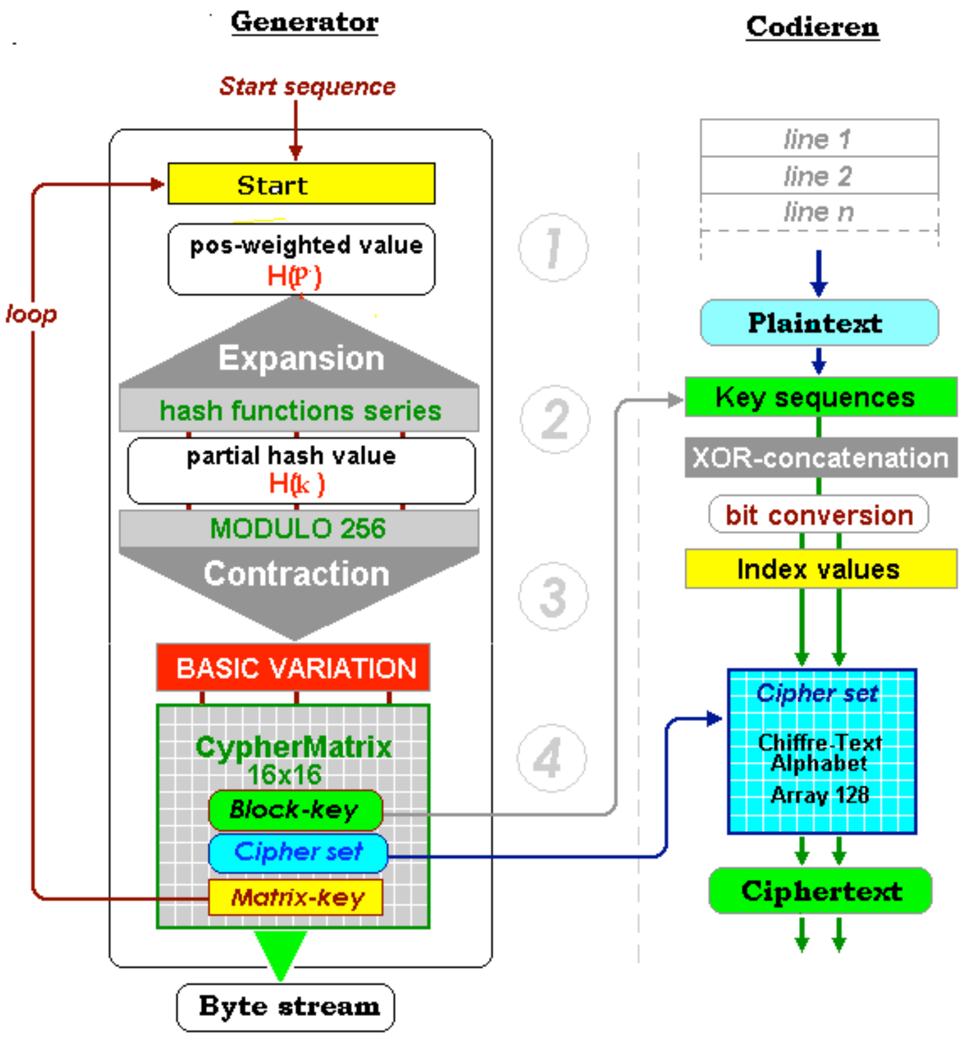
auf den Eingang zurückgeführt (loop) und der Durchlauf beginnt von Neuem.

Verschlüsseln

Der Generator generiert die für blockweise Verschlüsselung notwendigen Steuerungsparameter: Matrix-Schlüssel, Runden-Schlüssel und Runden-Alphabete.

Die Verschlüsselung wird allein durch die Operationen „one-time-chain" und „Bit-Konversion" durchgeführt. Die Startsequenz wird zur Verschlüsselung nicht benötigt und auch nicht gespeichert. Der Matrix-Schlüssel wird in jeder Runde neu generiert und zur Initialisierung der nächsten Runde auf den Eingang zurückgeführt (loop). Die Schleife beginnt von Neuem und das Programm verschlüsselt die folgende Klartextzeile.

Die folgende Übersicht zeigt die Zusammenhänge:



Ein Beispiel für die Verschlüsselung:

Klartext: Auf der Elbe fahren drei große Überseedampfer

Rundenkey: YLH 厎下弓냳叓 ЎJ 沦 БИ\叉 ﻹ 倇넥暑 3QĠPǔ 낦 R广ASÈŸ 勅냄石生TUϐȯK 냐止ДKOƕ癶殳б 넟 ƊRσOX 勫@Ʈ 劶劉剷 ȡʍΰʙ

XOR-Verkn: σЂ 吴 ЂЉJÈĠ 녗 T 劶 È 넚 ŸX 剃 H谷辶幺å₣ 呴 ŭ3ưϼ弋넒勫剃 C 倇歹3CI 龙手OS 넜꿍 O 녕 ΣB 厎 PЎ支Û6TBФ 끸 ȍË 넉止σЄƕ生

Ciphertext: 疋幺넜石Hƺϐϐ

„one-time-chain"

Der zu verschlüsselnde Text wird in gleicher Länge mit einem aus der CypherMatrix entnommenen Schlüssel **XOR**-verknüpft. Das Ergebnis als Bit-folge holt mit den dezimalen Werten der Elemente aus dem internen Zeichen-satz das zugeordnete Zeichen und verbindet es zur weiteren Arbeitsfolge.

Da Klartext und Schlüssel immer die gleiche Länge haben, entsteht auf diese Weise ein „partielles **one-time.pad**". Der Schlüssel wird auch nicht wiederholt. In jeder Runde wird ein anderer Schlüssel aus der jeweiligen CypherMatrix entnommen. Das ergibt für den gesamten Vorgang eine Kette zusammen-hängender „one-time-pad" Funktionen, gewissermaßen als „**one-time-chain**". Nach derzeitigem Stand wird absolute Sicherheit erreicht.

Bit-Konversion

Bit-Konversion ist die Umwandlung einer Bitfolge von einem Bitsystem in ein anderes Bitsystem, im vorliegenden Verfahren von 8-bit in 7-bit. Dabei bleiben die Anzahl der Bits und ihre Reihenfolge gleich. Kein Bit wird hinzugefügt und kein Bit wird weggelassen. Nur die Anzahl der Bits in einer Einheit ändert sich. Die dezimalen Werte der neuen Einheiten sind Indexwerte für das zugeordnete Alphabet. Die Bit-Konversion von Basis 8 zur Basis 7 geschieht im Einzelnen wie folgt:

Bitfolge im Original:

011000100110100101110100011001100110111101101100

98 105 116 102 111 108

Bitfolge nach Konversion:

011000100110100101110100011001100110111101101100

49 26 46 70 51 61 88

Entschlüsseln

Die Entschlüsselung geschieht in umgekehrter Reihenfolge. Alle Steuerungs-Parameter werden mit derselben Startsequenz erzeugt. Der Chiffretext nach der Bit-Konversion wird mit dem Runden-Schlüssel XOR-verknüpft, so dass auch hier ein **one-time-pad** entsteht. Als Ergebnis ergibt sich der ursprüngliche Klartex:

Der Ciphertext wird entschlüsselt

Ciphertext: 疋幺쌨石Hƺϐ 깒냉 OBC斤B 嘸 OPţR 訇曰吳 MŏKţ 녞 ŭ支ƕ 耆 A 吼ƺ 下歹@弓ɛ 勑 O 喢넉 ŭB 녳 J干N 心ИЦP 넓귅 Ġ 녞 T彡吳 R 鉋녑 û曰ϐ 낱냑彡訇下 ϐBSƕ

XOR-Verkn: ơЂ 吴 ЂЉJĚĠ 녳 T 峏 Ȅ 넚 ŸX 剓 H谷廴幺ǻɆ 咰 ŭ3ưp弋냚勫剓 C 鋌歹3CI 龙手OS 넜꿓 O 냉 ΣB 哌 PЎ支Û6TɃФ 핃 ȍЁ 넉止ơЄƕ生

Rundenkey: YLH 哌下弓냉叓 Ў心J 沦 бИ\叉ƺ 鋌넉暑 3QĠPŭ 넓 R广ASȄŸ 勑냄石生TUϐȯK 냑止ДKOƕ癶殳6 녳 ƊÐ歹ơOX 勫@Ţ 峏劚創 ȡмúɃ

Klartext: Auf der Elbe fahren drei große Überseedampfer nach Hamburg

Zur Unterscheidung werden verschlüsselte Dateien mit Präfix: „**Cy**-“ (Cyphertextdatei) und entschlüsselte Dateien mit Präfix: „**Kt**-“ (Klartextdatei) gekennzeichnet.

Anwendungen

Im Folgenden sind einige Beispiele aufgeführt, die mit Python 3.6.2 unter IDLE /Run / Run Module aufgerufen und ausgeführt werden können:

- Analyse.py (Zeichen analysieren)
- Buchstaben.py (eigener Zeichen-Satz)
- teleCypher.py (Dateien verschlüsseln)
- Pykurier.py (Kommunikation verschlüsseln)

Pychange.py	(sicherer Schlüsselaustausch)
Hashwert.py	(Hashwerte berechnen)
Userpass.py	(sicheres Passwortverfahren)

Weitere Informationen zu Python-Modulen stehen unter teleCypher.net zur Verfügung. Für Fragen und Erläuterungen kann der Autor per e-mail unter eschnoor@multi-matrix jederzeit angesprochen werden.

München, im Januar 2018

[#1] wikipedia.org/One-Time-Pad
[#2] wikipedia.org/wiki/Base64