CypherMatrix Datenverschlüsselung

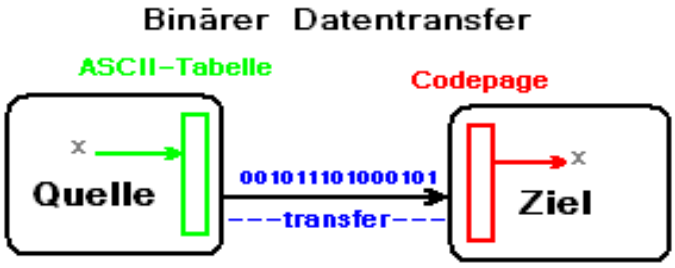
(Ernst Erich Schnoor)

CypherMatrix® als neues Verschlüsselungsverfahren verwendet auch eine neue Technik für den Datentransfer, die im Folgenden mit dem binären Datentransfer im Computer verglichen werden soll [#1]. Es stellt sich die Frage, ob die Verschlüsselung vielleicht nur ein Sonderfall des binären Datentransfers ist, oder -'vice versa' – ob der binäre Datentransfer nur einen besonderen Fall der Verschlüsselung darstellt?

Datentransfer im Computer

Wie beispielsweise kommen die Zeichen von der Tastatur zum Drucker? Texte, Wörter und Zeichen werden in Computern durch Zahlen repräsentiert, daher ist es notwendig, eine Zuordnung von Zahlen und Zeichen herzustellen. Diese Zuordnung wird durch eine Zeichensatztabelle definiert, die den darstellbaren Zeichen und Steuerzeichen Zahlenwerte zuordnet. Alternative Begriffe für Zeichensatztabelle sind **Codepage** oder Character Map [#2].

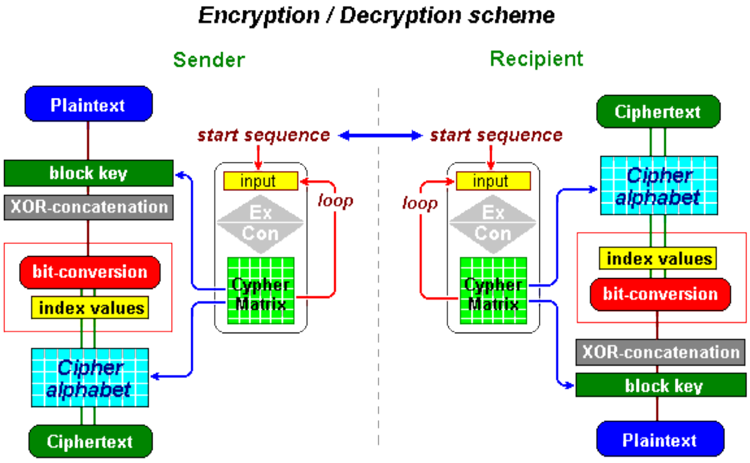
Die oberste Abstraktionsebene des binären Datentransfers zeigt sich somit als Übertragung von Zahlenwerten von der 'Quelle' zum 'Ziel', d.h. von der sendenden Codepage zur empfangenden Codepage. Das setzt natürlich voraus, dass beide Codepages auch gleich sind.



Da Computer nur die binären Zeichen „0" und „1" verarbeiten [#3], müssen die in der 'Quelle' (z.B: Tastatur) eingegebenen Zeichen einer binären Zahl zugeordnet werden ('00000000' bis '11111111' dh. dezimal 0 bis 255).
Die Zuordnung der Bitfolge erfolgt über Zeichen der definierten Codepage.
Die binären Zeichen werden zum 'Ziel' transferiert ('Schicht 1 OSI-Modell') und dort mit der gleichen Codepage in die ursprünglichen Zeichen zurückverwandelt.
Die Bitfolgen während des 'Transfers' können natürlich von Dritten gelesen werden. Insoweit ist von „Verschlüsselungen" noch keine Spur. Dazu müssen sowohl beim Sender ('Quell-Zeichensatz') als auch beim Empfänger ('Ziel-Zeichensatz') zusätzliche Schritte eingebaut werden.

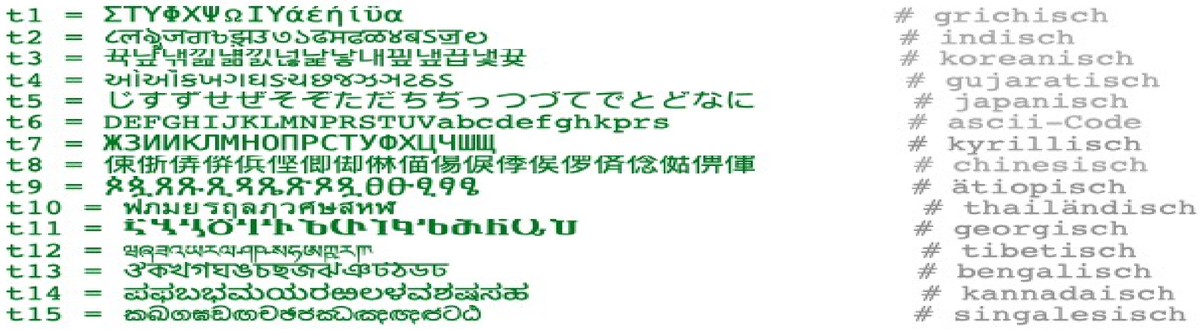
Datentransfer bei der Verschlüsselung

Der Transfer der Daten bei der Verschlüsselung hat im Prinzip die gleiche Aufgabe wie der digitale Datentransfer im Computer: Die Information von der 'Quelle' zum 'Ziel' zu transportieren.



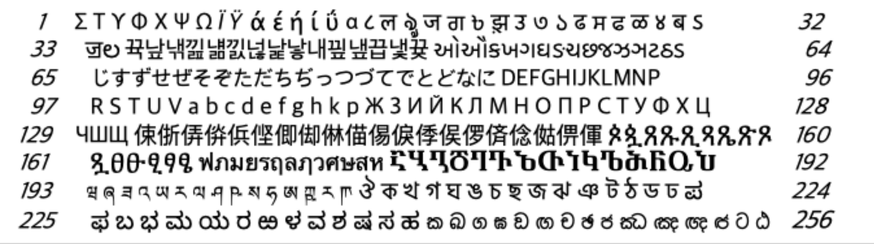
Sowohl beim Sender als auch beim Empfänger muss der gleiche Zeichensatz (codepage) arbeiten. Beim digitalen Datentransfer ist das bekanntlich der ASCII-Zeichensatz [#4]. Da die Zeichen jedoch fest definiert sind (statisch), kommen sie für dynamische Abläufe bei der Verschlüsselung kaum infrage. Hier sind 'Unicode-Zeichen' besser geeignet [#5]. Sie lassen sich digital darstellen und mit den binären Ziffern '0' bis '255' zu einem geeigneten Zeichensatz (Codpage) ordnen: **„Quell-Zeichensatz"**.

Vom Programm werden beispielsweise folgende 256 Unicode-Zeichen vorgegeben:



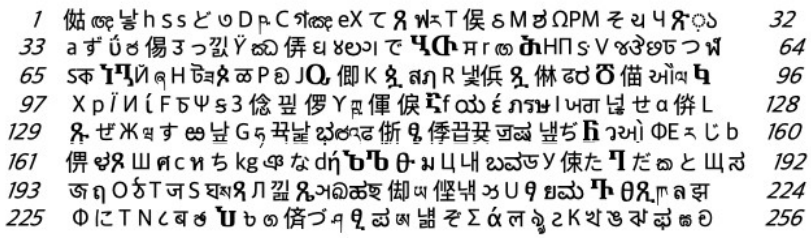
Für die Erzeugung eines dynamischen Zeichensatzes kommt hier der 'Matrix-Generator' ins Spiel [#6]. Eine Eingangsphrase mit 30 bis 96 Zeichen startet den Generator, der einen dynamischen 'Quell-Zeichensatz' erzeugt und daraus die weiteren Bestimmungsfaktoren für die Verschlüsselung ableitet.

'Quell-Zeichensatz'



Aus dem Zeichensatz generiert der Generator die jeweilige Runden-Matrix (256 Zeichen), sowie Runden-Schlüssel und Matrix-Schlüssel für die aktuelle Runde.

'Cypher-Matrix'



Die CypherMatrix-Verschlüsselung arbeitet mit folgenden Variationen:

Linearer Verlauf (stream-cypher)

Nur am Anfang des Verfahrens wird eine CypherMatrix (codepage) generiert. Da alle Chiffrezeichen aus der einzigen Matrix abgeleitet werden, ensteht ein linearer Verlauf des Verfahrens (statisch).

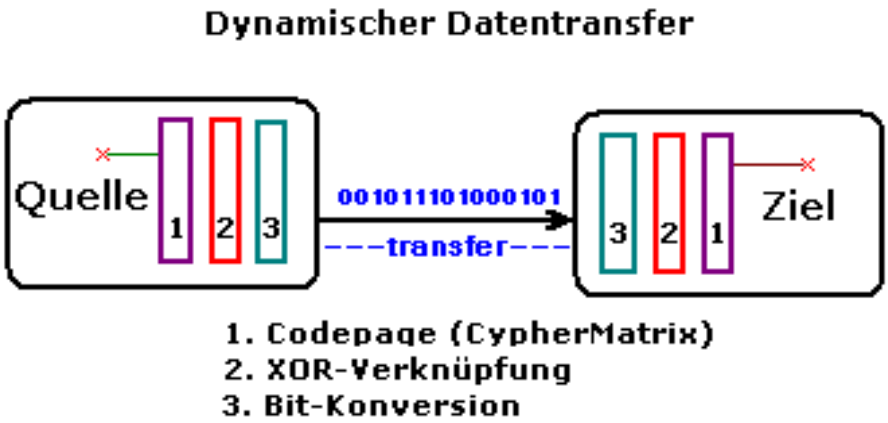
Klartext → CODEPAGE → ***Chiffretext*** (Datentransfer)

teleCode -1.py

Wird dagegen für jede Runde eine neue Runden-Matrix mit unterschiedlicher Zeichenstruktur generiert, entsteht ein dynamischer Verlauf des Verfahrens. Folgende Auszug aus dem Quellcode des Programms: '*teleCode-1.py*'.

```
zeile = ""
lesen = open(datei,"r")
zeile = lesen.read()                                          # Klartext
i = 0
a = ""
matrkey,matrix = generator(matrkey,breite,cd)          # CODEPAGE (statisch)
umfang = len(zeile)
for a in zeile[0:umfang]:                                    # 'stream'-cypher
    digit = ord(a)
    zeichen = matrix[digit]
    print(zeichen)                                       # Bildschirmdarstellung
chiffretext = chiffretext + zeichen                           # Chiffretext
```

Dynamischer Verlauf (block-cypher)



Die einzelnen Programme enthalten folgende Schritte:

1. ***Klartext*** → CODEPAGE → ***Chiffretext*** (Datentransfer)

   telecode -2.py

2. ***Klartext*** → CODEPAGE → ***7-bit-Konversion*** → ***Chiffretext*** (Datentransfer)

   teleCode -.py*

3. ***Klartext*** → CODEPAGE → ***7-bit-Konversion*** → ***8-bit XOR-Verknüpfung*** → **Chiffretext (T**ransfer)

   paraCypher -.py*

4. ***Klartext*** → CODEPAGE → ***8-bit XOR-Verknüpfung*** → ***Chiffretext*** (Datentransfer)

   teleCypher-extra.py

5. ***Klartext*** → CODEPAGE → ***8-bit XOR-Verknüpfung*** → ***7-bit-Konversion*** → ***Chiffretext*** (Transfer)

   teleCypher -.py*

Die Operationen können in den beigefügten Skript-Programmen getestet werden. Im Folgenden ein Auszug aus dem Quellcode des dynamischen Block-Cypher Programms: *teleCypher.py*

```
zeile = ""
lesen = open(datei,"r")
zeile = lesen.read()                                          # Klartext
breite = 56                                                  # Blockbreite
i = 0
chiffretext = ""
abschn = ":"
while zeile:                                                  # Rundenlauf
    abschn = zeile[i:i+breite]                               # Blockbildung
    i += breite
    if len(abschn) == 0:
        break
    rundkey,matrkey,alphabet,matrix = generator(matrkey,breite,cd) # CODEPAGE
    eins = abschn
    zwei = ""
    drei = rundkey
    vier = ""
    resultat = XORgen(1,eins,zwei,drei,vier)                 # XOR-Verknüpfung
    cypher = bitsdown(resultat,matrix)                        # Bit-Konversion
    zeichen = bytes(cypher.encode("utf8"))
    print(cypher)                                        # Bildschirmdarstellung
chiffretext = chiffretext + cypher                                # Chiffretext
```

Entschlüsselung

Die Chiffretext-Datei erhält dieselbe Bezeichnung wie die Klartext-Datei mit der zussätzlichen Erweiterung: „.ctx“ (Geheim.txt.ctx). Der Ablauf geschieht in gleicher Weise wie bei der Verschlüsselung, nur in der umgekehrten Reihen-folge. Der Arbeitstext 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 Klartext.

Chiffretext → ***8-bit XOR-Verknüpfung*** → ***7-bit-Konversion*** → CODEPAGE → ***Klartext***

paraCypher -.py*

Chiffretext → ***7-bit-Konversion*** → ***8-bit XOR-Verknüpfung*** → CODEPAGE → ***Klartext***

teleCypher -.py*

Für die praktische Anwendung sollten die dynamischen Programme mit der integrierten XOR-Verknüpfung (Gruppen 3 bis 5) bevorzugt werden.

Sicherheit

Programme der 1. Gruppe der Operatoren (linearer Verlauf) bieten nur eine geringe Sicherheit. Infolge des statischen Zeichensatzes wird für jedes Klartextzeichen dasselbe Chiffrezeichen gesetzt. Damit können vor allem Häufigkeitsanalysen, Wiederholungsmuster und Wortkombinationen zum Brechen der Chiffre führen.

Programme der Gruppe 2 erreichen zusätzliche Sicherheit dadurch, dass Klartext und Chiffretext in verschiedenen Bit-Systemen arbeiten (Bit-Konversion). Zum Brechen der Chiffre müsste die jeweils erzeugte Runden-Matrix (Codepage) gefunden werden, um die Ordnungsziffern der verwendeten Zeichen feststellen zu können. Diese Barriere ist nicht zu überwinden.

Die Sicherheit der Gruppen 3 bis 5 wird insbesondere durch folgende Eigenschaften verstärkt „one-time-chain" und „bit-konversion" (Umwandlung von 8-bit in andere Bitfolgen (5,6,7,9,10 oder) 11-bit) [#7].

'**One-time-chain**'

Jeder Klartext-Block wird mit dem aus der jeweiligen Runden-Matrix entnommenen und gleich langen Runden-Schlüssel **XOR**-verknüpft (one-time-pad). Das Ergebnis als 8-Bit Folge holt das zugeordnete Zeichen aus der CypherMatrix und verbindet es zur weiteren Arbeitsfolge. Für das gesamte Verfahren ergibt sich somit eine Kette zusammenhängender „one-time-pad" Funktionen, gewissermaßen als „**one-time-chain**". Nach derzeitigem Stand wird eine absolute Sicherheit erreicht [#8].

Bit-Konversion

Bit-Konversion ist die Umwandlung einer Bitfolge von einem Bitsystem in ein anderes Bitsystem. Kein Bit wird hinzugefügt und kein Bit wird weggelassen. Es wird nur eine andere Struktur gebildet.

Bitfolge im Original in 8-bit:

011000100110100101110100011001100110111101101100

98 105 116 102 111 108

Bitfolge nach Konversion in 7-bit:

011000100110100101110100011001100110111101101100

49 26 46 70 51 61 88

Quintessenz

Als Schlussfolgerung aus den dargestellten Zusammenhängen zeigt sich ein korrelatives Verhältnis zwischen binärem Datentransfer und Verschlüsselung. Da der binäre Datentransfer schon von Anfang an verwendet wird, kann die CypherMatrix-Verschlüsselung nur aus dem binären Datentransfer abgeleitet sein. Die Verschlüsselung ist insoweit ein **Sonderfall** des allgemeinen binären Datentransfers.

München, im Juli 2020

[#1] Algorithmus-CypherMatrix,telecypher.net/Algorithmus-CypherMatrix.pdf

[#2] Wikipedia.org/wiki/Zeichensatztabelle

[#3] Wikipedia.org/Dualsystem

[#4] Wikipedia.org/Codepage 850

[#5] Wikipedia.org/wiki/Schriftsysteme in Unicode

[#6] Matrix-Generator, http://www.telecypher.net/Matrix-Generator.pdf

[#7] CypherMatrix Verfahren, http://www.telecypher.net/CYPHKERN.HTM

[#8] Wikipedia.org/XOR-Gatter