Vorläufige Beschreibung der Chiffrieralgorithmusklasse

D E L T A

1. Einleitung

Die im Punkt 2 definierte CA Klasse kann in den standardisierten Arbeitsmoden für 64-Bit Blockchiffrieralgorithmen gemäß ISO 8372 angewandt werden.
Die Strukturierung soll zu einem schnellen Algorithmus für 16-Bit Prozessoren führen und gleichzeitig bei richtiger Anwendung die notwendige Sicherheit bieten.
Bezeichnungen sind für alle kryptanalytischen Arbeiten im Referat verbindlich.
Die Beschreibung ist nicht vollständig, sie will aber zur Standardisierung und somit zur Effektivität der Arbeit beitragen. Im Zuge der Entwicklungs- und Analysearbeiten erfolgen weitere Präzisierungen. Änderungen sind möglich.
Die Beschreibung der Arbeitsweisen im CBC und CFB Modus erfolgen später.

2. Definitionen

2.1. Bezeichnungen

i,j,k,t Indizes, $T \in N$

$\forall m \in N$: $Vm =: \{0,1\}m$

R1,R2,R3,R4 Register zu je 16 Bit

$\forall(i,t) \in 1,4 \times 0,T$

$ai(t) \in V_{16}$ Inhalt des Registers Ri

zum Zeitpunkt (Takt) t

$a(t) = (a_j^i(t))_{j=1}^{16} V$ wobei $a_j^i(t) \in \{0,1\}$

Im weitern gilt $k \in \{1,2,3\}$ beliebig, aber fest.

Sei $X(t) \in Vk*16$

$X(t) = (x_j(t))_{j=4}^{k*16}$, wobei $x_j(t) \in \{0,1\}$

2.2 Die Abbildung ΦK

$\Phi K: V_{16} \times V_{16} \times V_{16} \times VK*16 \dashrightarrow V_{16}$

(ΦK wird ggf präzisiert. Die Abbildung sollte sich aus
Operationen zusammengesetzt, die schnell EDV mäßig verarbeitbar
sind. z. B.

- ⊠ Addition mod 2^{16}
- ⊗ Addition mod $2^{16} - 1$

$\oplus$ bitweise Addition mod 2 (blockweise)

$\wedge$ bitweise and (blockweise)

rot zyklische Verschiebung links oder rechts

P Permutation V_{16} --> V_{16}

Negation

2.3 Beschreibung einer Runde der Arbeitsweise des Blockchiffrieralgorithmus

Abbildung D: V_{64} x VK*16 --> V_{64}

$\forall t \in 0,T-1$:

$(a^1(t+1),a^2(t+1),a^3(t+2),a^4(t+1)) = D((a^1(t),a^2(t),a^3(t),a^4(t)), X(t))$

wobei gilt $a^1(t+1) = a^2(t)$

$a^2(t+1) = a^3(t)$

$a^3(t+1) = a^4(t)$

$a^4(t+1) = a^1(t) \oplus \Phi K((a^2(t),a^3(t),a^4(t), X(t))$

Abbildung D^{-1} : V_{64} x VK^{*16} --> .V_{64}

$\forall\ t \in 0,T\text{-}1$:

$(a^1(t),a^2(t), a^3(t),a^4(t)) = D^{-1}(a^1(t+1),a^2(t+1),a^3(t+1),a^4(t+1)), X(t))$

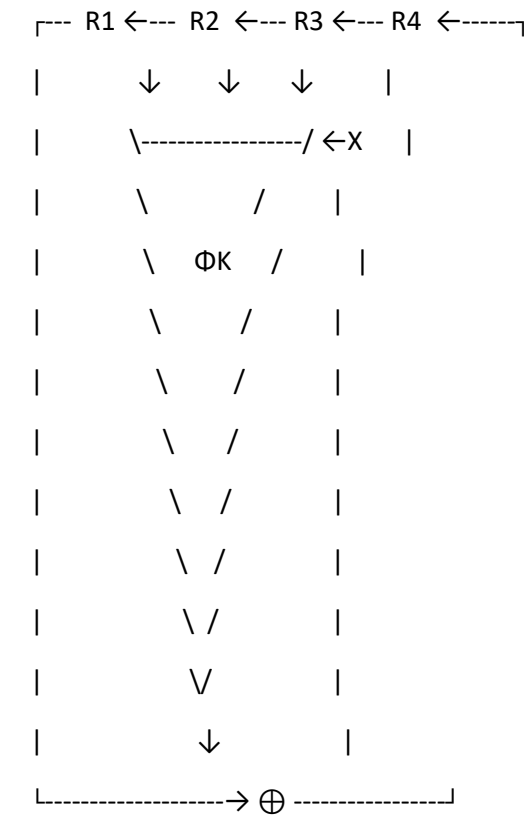
wobei gilt: $a^1(t) = a^4(t+1) \oplus \Phi K(a^1(t+1),a^2(t+1),a^3(t+1),X(t))$

$a^2(t) = a^1(t+1)$

$a^3(t) = a^2(t+1)$

$a^4(t) = a^3(t+1)$

Skizze:



2.4 Anzahl der Schlüsselelemente und Anzahl der Runden

Es wird vorerst auf eine Schlüssellänge von 256 Bit auf 32 Runden orientiert.

Sollten Algorithmen dieser Klasse auf der Ebene unterhalb von Staatsgeheimnissen eingesetzt werden, dann sind z. B. auch 128 Bit Schlüssellänge und 16 Runden denkbar.

In der Abb. Φ sollte ein hinreichend großer LZS wirken.

3. Bezeichnungen für die Analyse

3.1. MEDWEDJEV-Automat A = Ak(Par.,n)

Par.-Bezeichnung für den variablen Parameter, der durch die Wahl eines LZS fixiert wird.

Eingabenmenge V^{n}_{K*16}

Zustandsmenge V_{64}

Überführungsfunktion: $\delta: V^{n}_{k*16} \times V_{64} \dashrightarrow V_{64}$

$$\forall (X(t))_{t=1}^{n} \in V^{n}_{k*16} \quad \forall a \in V_{64} \quad \delta((X(t)_{t=1}^{n}) \; Dn \; (X(1),X(2),.....X(n))a$$

partielle Überführungsfunktion:

$$\forall\,(X(t))_{t=1}^{n} \in V \qquad \forall\, a \in V_{64}\ \delta x(1),x(2),...X(n)\ \ a =: Dn(X(1),X(2),.....X(n))a$$

wobei $Dn(X(1),X(2),.....,X(n))\ a =: D(X(n),D(X(n-1),.....D(X(1),a)...))$

Gruppe des Automaten

$$GA =: < \{\ \delta(X(1),X(2),.....X(n) \mid \{\ X(1),X(2),.....X(n) \in V^{n}_{k*16}\ \} >$$

Anlage 1

Varianten für den DELTA-Algorithmus

(gleichzeitig Beschreibung des ECB Modus)

Sei a der Klartextblock mit 64 bit und c der Geheimtextblock

mit 64 bit d.h. $a \in V_{64}$, $c \in V_{64}$

Variante DELTA 1\1

$n = 64$

$S \in V_{256}$

$S = (s_1 , ..., s_{256}),$ wobei$\forall$ i $\in$ 1,256 : si $\in$ (0,1)

t $\in$ 1,16 : X(t) = (s(t-1)16+1 ,.....,s(t-1)16+16)

t $\in$ 17,64 : X(t) = rot(7R) X(t-16)

$\Phi 1\ (a^2(t), a^3(t), a^4(t), X(t) =: rot\ (3L)\ P\{[a^2(t) \boxtimes a^3(t)\ ] \oplus [a^4(t) \boxtimes X(t)]\}$

Die Chiffrierabbildung:

$c =: D^{64}\ [X(1),X(2),....,X(64)]\ a$

Die Dechiffrierabbildung:

$a =: D^{-64}\ [X(64),X(63,....;X(2)]\ c$

Variante DELTA 1\2

n = 64

$S \in V^{256}$

$S = (s_1 , ...,s_{256})$, wobei $\forall$ i $\in$ 1,256 : si $\in$ {0,1}

t $\in$ 1,16 : X(t) = (s(t-1)16+1 ,....,s(t-1)16+16)

t $\in$ 17,64 : X(t) = rot(7R) X(t-16)

Π - Permutation der 4 16-bit Blöcke:

$\Pi\ (a^1(t),a^2(t),\ a^3(t),a^4(t)) =: a^4(t),a^3(t),a^2(t),a^1(t)$

$\Phi 1\ (a^2(t),\ a^3(t),a^4(t),\ X(t)) =: rot(3L)\ P\{[a^2(t) \boxtimes a^4(t)] \oplus [a^3(t) \boxtimes X(t)]\}$

Die Chiffrierabbildung:

$c =: \Pi\ \{D^{64}[X(1),X(2),....,X(64)]a\}$

Die Dechiffrierabbildung:

$a =: \Pi\ \{D^{64}[X(64),X(63),...,X(1))]c\}$

Varianten für den DELTA-Algorithmus

(gleichzeitig Beschreibung des ECB-Modus)

Sei a der Klartextblock mit 64 Bit und c der Geheimtextblock

mit 64 Bit, d. h. $a \in V_{64}$, $c \in V_{64}$

Variante DELTA 2\1

$n = 32,$

$s \in V_{256}$

$S = (s_1 , ... , s_{256}\ )$, wobei $\forall\ i \in 1,256 : si \in \{0,1\}$

$t \in 1,8 : X(t) = (s\ (t-1)*32+1\ \ \ \ \ \ , ... , s(t-1)*32+32\ )$

$t \in 9,32 : X(t) = ROR_7\ X(t-8)$

$t \in 1,32 : x^1(t) = ((X_1(t)) , ... , (X_{16}(t)))$

$x^2(t) = ((X_{17}(t)) , ... , (X_{32}(t)))$

$\Phi_2\ (a^2(t), a^3(t), a^4(t), X(t)) =: P\ \{[a^2(t) \boxtimes a^4(t) \boxtimes X^1(t)\ ] \oplus (ROR_3\ [a^3(t) \boxtimes x^2(t)\ ]\ )\ \}$

Die Chiffrierabbildung:

$c =: D^{32}\ [X(1), X(2), ... , X(32)]\ a$

Die Dechiffrierabbildung:

$a =: D^{-23}\ [X(32), X(31), ... , X(1)]\ c$

Variante DELTA 2\2

$n = 32,$

$S \in V_{256}$

$S = (s_1 , ..., s_{256}\ ),$ wobei $\forall\ i \in 1,256 : si \in \{0,1\}$

$t \in 1,8 : X(t) = (s(t-1)*32+1 , ... , s(t-1)*32+32)$

$t \in 9,32 : X(t) = ROR_7\ X(t-8)$

$t \in 1,32 : X^1(t) = ((X(t))_1 , ... , (X(t))_{16})$

$X^2(t) = ((X(t))_{17} , ... , (X(t))_{32})$

Π - Permutation der 4 16-Bit Blöcke

$\Pi (a^1(t), a^2(t), a^3(t), a^4(t)) =: (a^4(t), a^3(t), a^2(t), x^1(t)]\}$

$\Phi_2 (a^2(t), a^3(t), a^4(t), X(t)) =: P \{[a^2(t) \boxtimes a^4(t) \boxtimes X^1(t)] \oplus (ROR_3 [a^3(t) \boxtimes X^2(t)]) \}$

Die Chiffrierabbildung:

$c =: \Pi (D^{32} [X(1), X(2), ... , X(32)] a)$

Die Dechiffrierabbildung:

$a =: \Pi (D^{32} [X(32, X(31), ... , X(1)] c)$

Anlage 2

Die Arbeitswiese im CBC/CFB Modus und Imitationsschutz.
(Anmerkung: Durch die Beschreibung der Varianten des DELTA Algorithmus in Anlage 1 ist die Arbeitsweise im ECB Modus vollständig beschrieben.)

1. Die Arbeitsweise im CBC Modus ist mit der Beschreibung in ISO 8372 vollständig identisch.

2. Die Arbeitsweise im CFB Modus ist mit der Beschreibung in ISO 8372 vollständig identisch.

3. Für den OFD Modus wird folgendes festgelegt:

$A(\emptyset), B(\emptyset), E, F, \in V_{32}$; $\boxtimes_{32}$ Add. mod. 2^{32}

$\otimes_{32}$ Add. mod. $2^{32} - 1$

$A(\emptyset), B(\emptyset)$ werden aus einem Startvektor SY (SyF) von 64 Bit gebildet

E, F seien Konstanten (ungerade)

Erzeugt werden die Folgen $A(j), B(j), j \in 1,m , m \in N;$

$A(j) =: A(j-1) \boxtimes_{32} E$

$B(j) =: B(j-1) \otimes_{32} F$

Weiterhin sei:

$\forall j \in 1,m: a(j) = (A(j), B(j))$

Die Additionsreihe $C(j)$ $j \in 1,m$ ergibt sich durch Anwendung des Chiffrieralgorithmus (im ECB Modus auf die Folge

$a(j)$ $j \in 1,m.$

(Die Konstanten E, F sind noch zu präzisieren.)

4. - Durch Anwendung des CBC Modus ist ein gewisser Imitationsschutz gegeben.

- Falls eine offene Übertragung oder Speicherung der Daten vorgesehen ist, bietet die Speicherung bzw. Übertragung nur des letzten 64-Bit Vektors des im CBC Modus chiffrierten gesamten Textes einen Imitationsschutz. (Ob für diese Version eine Reduzierung der Rundenzahl in der Arbeit des Chiffrieralgorithmus vorgenommen werden kann, bleibt zu Untersuchen.)

- Die Einbindung einer Imitationsschutzvariante für den Algorithmus muß weiter untersucht und präzisiert werden.

E := 000 1 000 000 1 000 000 1 000 000 1

P := 000 000 01 0001 0001 0001 0001 0001 0001

Referat 21 Berlin, 29.5.90

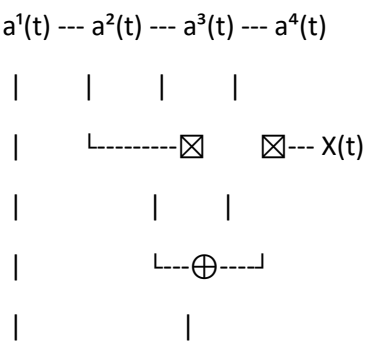
Ergänzung zur Anlage 2, Punkt 3.

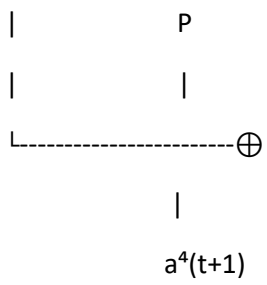
Vorläufige Variante für E und F

E =: (0001 0000 0010 0000 0100 0000 1000 0001)

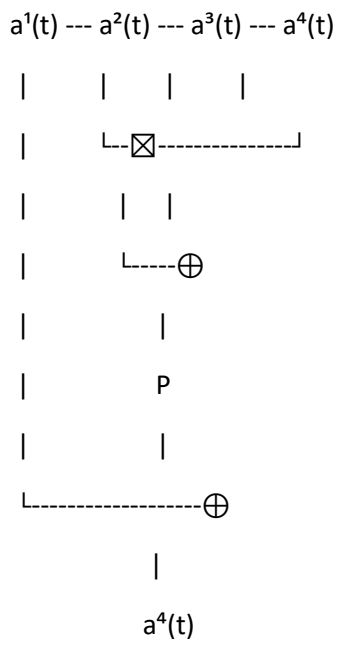
F =: (0000 0001 0001 0001 0001 0001 0001 0001)

Version DELTA 1\1

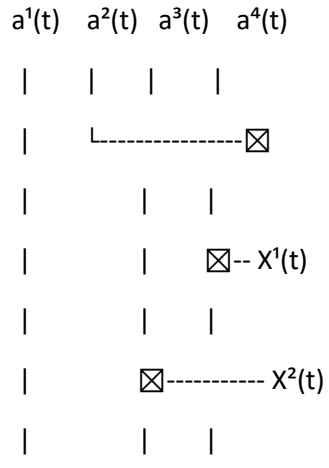


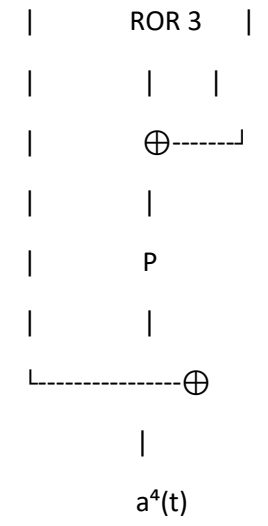


Version DELATA 1\2

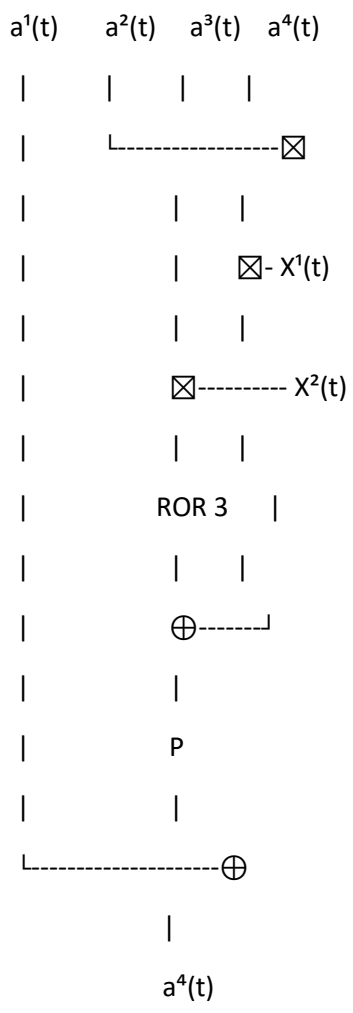


Version DELATA 2\1





Version DELTA 2\2



Abschätzen Laufzeiten 12.7.1990

Anlage Programm Korn Δ 1\1 44 Befehle

Δ 1\2 44

Δ 2\1 60

Δ 2\2 60

Vergleich R (9,6 Kbit) 96

DES (9,6 Kbit) 85

R (reine ~200
Software)

DES (reine ~200
Software)

gesicherte Aussage für alle Variante DELTA!

Doppelt so schnell wie die reine Softwarelösung DES

(z.Z. 12 ms ohne xxx)

2 <= 6 ms (f = 2,5 MHz, Z80)

32 Umläufe

Δ 1 schneller Δ 2

Δ 1 <= 5 - 5,2 ms

16 Umläufe für Δ 2

~ 3ms 4 x so schnell wie DES, reine Softwarelösung

Programmversion Korn Δ1\1 Entwurf, ungetestet)

6L. SYF - Synchronfolge

6L. KEY - Schlüssel

6L. Perm1 - Permutation1

Perm2

Ausgangspunkt:

A - UMLZ

BC - $a^4(t)$

DE - $a^3(t)$

HL - $a^2(t)$

IX - $a^1(t)$

Stack:

a2 (t)

a1 (t)

a3 (t)

a4 (t)

UMLZ

```
MØ:  PUSH AF      ; UMLZ (Umlaufzähler)
     PUSH BC      ; a4 (t)
     PUSH DE      ; a3 (t)
     PUSH IX      ; a1 (t)
     PUSH HL      ; a2 (t)
                          ┐
     LD A,D       ; a2 (t)          |          (-)
     CPL          ; (-)             |
     ADD H        ; ⊠               > a²(t) ⊠ a³(t)
     LD D,A                         |
     LD A,E                         |
     CPL                            |
     ADC L                          |
     LD E,A                         ┘
```

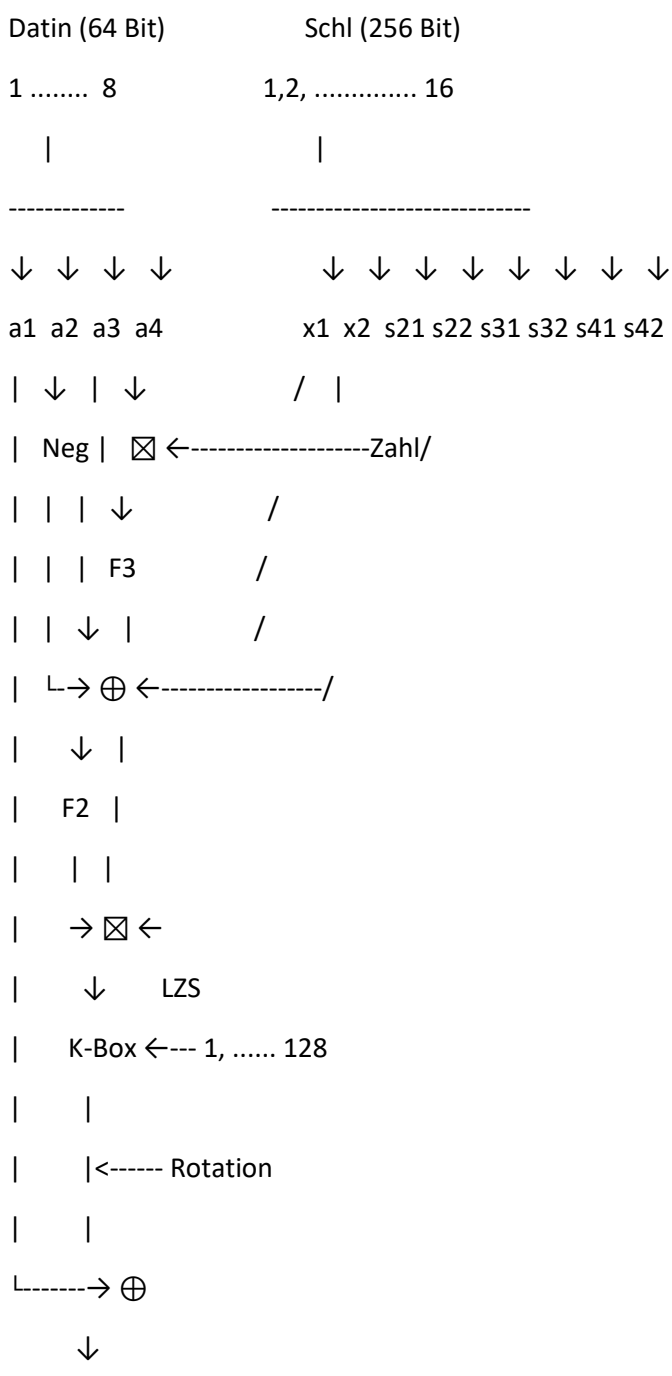
```
LD HL, 6L.KEY              ┐
LD A,B                     |
ADD H                      |
LD B,A                     > a⁴(t) ⊠ X(t)
INC HL                     |
LD A,C                     |
ADC H                      ┘
XOR E       ; ⊕ a³(t)
LD C,A
LD A,B
XOR D
LD B,A      ; ⊕ a³(t) Ergebnis in BC

LD H,6L.PERM1  ; Permutation 4 Bit Adresse
LD L, B     ;         -> 4 Bit Deth.
LD B, H
LD H,6L.PERM2
LD L,C
LD C,H

POP IX
POP DE      ; a²(t)          a⁴(t) -> a³(t) -> a²(t) -> a¹(t)

LD A,B
XOR D       ; a¹(t) ⊕ Ergebnis PERM
LD B,A
LD A,C
XOR E
LD C,A      ; in BC neuer Wert a⁴(t+1)
POP HL      ; alt a⁴(t)   a²(t+1)
POP DE      ; alt a⁴(t)   a³(t+1)
```

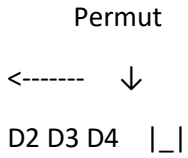
POP AF

DEC A

JRNZ MØ-#

Versuchsalgorithmus V2 (32 Runden)




Permut
<-------
D2 D3 D4
|_|