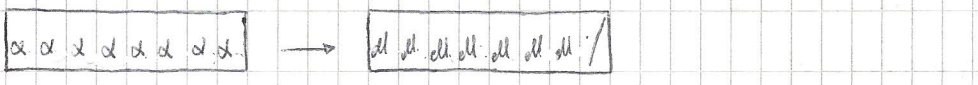
(ITEMS 1→6)

2013-01-30

ITEM 1: compute ENC_E, given CLEAR and λ

```
; read LAMBDA value
; read CLEAR array and compute it's length (LEN)
enc_E_wrap:
    mov LEN, cl
    xor SI, SI      ; used as index for both CLEAR and ENC_E arrays
    mov DL, LAMBDA
enc_E_loop:                 ; μ0 = LAMBDA
    XOR AH, AH
    MOV AL, CLEAR[SI]       ; αi
    ADD AL, DL              ; αi + μi-1
    PUSH CX
    MOV CL, 7
    ROR AL, CL
    SHR AL, 1       ; <AL> = 0 μ7 μ6 μ5 μ4 μ3 μ2 μ1
    POP CX
    MOV ENC_E[SI], AL
    MOV DL, AL      ; used for the encryption of the next char
    INC SI
    LOOP enc_E_loop
; print encrypted message
```



TO IMPLEMENT MODULO 128 = SHR 7 AND STORE THE RESULT IN THE SAME REGISTER

encrypted value

ITEM 2: compute DEC_E, given ENC_E and λ

```
; read value of LAMBDA
; read encrypted message (numbers)
dec_E_wrap:
    mov LEN, CL
    Xor SI, SI    ; used as index for DEC_E and ENC_E arrays
    MOV DL, LAMBDA
dec_E_loop:
    XOR AH, AH
    MOV AL, ENC_E [SI]
    SUB AL, DL
    CMP AL, 0
    JGE update_Dec_E
    ADD AL, 128
update_dec_E:
    MOV DEC_E [SI], AL
    MOV DL, ENC_E [SI]
    INC SI
    loop dec_E_loop
    ; print DEC_E array as characters
```

	λ			
ENC_E	M_1	M_2	M_3	M_4
DEC_E	α_1	α_2	α_3	α_4

$\alpha_1 = M_1 - \lambda$

IF $\alpha_1 \geq 0 \Rightarrow$ STORE α_1

IF $\alpha_1 < 0 \Rightarrow$ STORE $\alpha_1 + 128$

$\alpha_i = M_i - M_{i-1}$

IF $\alpha_i \geq 0 \Rightarrow$ STORE α_i

IF $\alpha_i < 0 \Rightarrow$ STORE $\alpha_i + 128$

ITEM 3: (decrypt only one char at position I)
compute DX_I, given λ, I, ENC_E

```
; read λ, I and ENC_E

... ; <AL> ← I
MOV I, AL
CMP AL, 0
JE decryptWithLawbdaE
XOR DI, DI
XOR AH, AH
ADD DI, AX          ; position I
SUB DI, 1           ; position I-1
MOV DL, ENC_E[DI]   ; M_{i-1}
JMP decryptOneCharE
decryptWithLawbdaE:
MOV DL, LAMBDA
decryptOneCharE:
XOR SI, SI
ADD SI, AX          ; position I
SUB AL, DL          ; M_i - M_{i-1}  OR  M_1 - λ
CMP AL, 0
JGE update_DX_I
ADD AL, 128         ; IF M_i - M_{i-1} OR M_1 - λ < 0 => ADD 128
update_DX_I:
MOV DX_I, AL
; print decrypted char:
    MOV AH, 2
    MOV DL, DX_I
    INT 21H
```

ITEM 4: compute ENC_D, given CLEAR and λ

```
; read CLEAR and LAMBDA
enc_D_wrap:
    MOV LEN, CL
    XOR SI, SI
    MOV DL, LAMBDA
enc_D_loop:
    XOR AH, AH
    MOV AL, CLEAR [SI]      ; char to be encrypted : d_i
    ADD AL, DL              ; sum λ + previous encrypted chars  modulo 128
    PUSH CX
    MOV CL, 7
    ROR AL, CL              ; compute modulo 128 (as in ITEM 1)
    SHR AL, 1
    POP CX
    MOV ENC_D [SI], AL
    ADD DL, AL              ; add to λ and previous enc chars, the current enc char
    INC SI
    loop enc_D_loop
; print ENC_D array (numbers)
```

ITEM 5: compute DEC_D, given ENC_D and λ

```
; read λ and ENC_D (numbers)
dec_D_wrap:
    MOV LEN, CL
    XOR SI, SI
    MOV DL, LAMBDA
dec_D_loop:
    XOR AH, AH
    MOV AL, ENC_D[SI] ; char to be decrypted (M_i)
    SUB AL, DL
    ADD DL, ENC_D[SI] ; to be used in the next decryption
    CMP AL, 0
    JGE update_dec_D
    ADD AL, 128
update_dec_D:
    MOV DEC_D[SI], AL
    ADD SI, 1
    loop dec_D_loop
```

$M_0 = \lambda$

M_1	M_2	M_3	M_4

α_1	α_2	α_3	α_4

DECRYPTION WITH ALGO D:

$$\alpha_i = M_i - M_{i-1} - \dots - M_0$$

IF $\alpha_i \geq 0 \Rightarrow$ STORE α_i

IF $\alpha_i < 0 \Rightarrow$ STORE $\alpha_i + 128$

ITEM 6 : COMPUTE DX_I = char at position I decrypted with algo D

```
; read I, λ, ENC_D
    MOV DL, LAMBDA
    CMP AL, 0
    JE decryptOneCharD
    XOR SI, SI
    MOV CL, I
    XOR CH, CH
addLoop:
    ADD DL, ENC_D [SI]
    ADD SI, 1
    loop addLoop          ; at the end of the loop, <DL> = λ + M1 + ... + Mi-1
decryptOneCharD:
    XOR AH, AH
    XOR DI, DI
    ADD DI, AX  ; position I
    MOV AL, ENC_D [DI]        ; char at position I to be decrypted
    SUB AL, DL
    CMP AL, 0
    JGE saveDecryptedCharD
    ADD AL, 128
saveDecryptedCharD:
    MOV DX_I, AL
; print DX_I
```