Sümmeetrilised krüptoalgoritmid

Erika Matsak

1

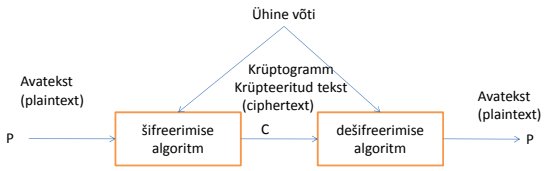
Sisukord

- Vaadeldakse sümmeetriliste krüptoalgoritmidega seotud põhialuseid, krüpteerimisvõti, avateksti (plaintext), krüptogrammi (ciphertext).
- Esitatakse algoritmi kindluse määratlus
- Vaadeldakse operatsioonide tüüpe, mis on seotud sümmeetriliste krüptoalgoritmidega
- Vaadeldakse Feisteli võrku, mis on aluseks paljudele kaasaegsetele sümmeetrilistele krüptoalgoritmidele
- Esitatakse DES ja kolmekordse DES algoritmi kirjeldus
- Vaadeldakse lineaarset ja diferentsiaal-krüptoanalüüsi

2

Sümmeetrilise krüptosüsteemi põhialused 1

- Vaatleme sümmeetrilise ehk traditsioonilise krüptograafia põhiskeemi



Võti on sõltumatu algsõnumist. Võtme muutmine muudab krüpteeritud sõnumit

3

Sümmeetrilise krüptosüsteemi põhialused 2

- Sümmeetrilise krüptosüsteemi klassikalised näited:
  - Lihtne transponeerimine (lepitakse kokku tabeli suurus, avatekst kirjutakse tulpadesse, siis loetakse ridade kaupa – ehk transponeeritud kujul)
  - Valikuline ümberpaigutus võtme abil (erineb eelmisest selle poolest, et tulbad paigutatakse ümber võtme alusel)
  - Topelt ümberpaigutamine (näiteks kasutades esimest varianti tehakse kaks järjestikust kodeerimist erinevate ruutudega)
  - "Maagilise ruudu" ümberpaigutus

16	3	2	13
5	10	11	8
9	6	7	12
4	15	14	1

16.	3 A	2 A	13 U
5 U	10 M	11 I	8 O
9 M	6 N	7 H	12 K
4 B	15.	14 L	1 S

Sümmeetrilise krüptosüsteemi põhialused 3

- Turvalisus, mida tagatakse krüptograafia abil sõltub mitmest asjaolust:
  - Algoritm, millega teostatakse krüpteerimist, peab olema "kindel", st – et olukorras, kus krüpteerimisvõtit ei teata, ei oleks võimalik sõnumit dešifreerida erinevate statistiliste meetodite või mingi teise analüüsi abil
  - Edastava sõnumi turvalisus peab olema sõltuv krüpteerimise võtme salastamisest, mitte algoritmi salastamisest. Algoritm tuleb analüüsida spetsialistide poolt, et selgitada välja selle nõrgad kohad
  - Algoritm peab olema selline, et ei oleks võimalik kätte saada krüpteerimisvõtit, isegi siis, kui on teada palju šifreeritud ja dešifreeritud sõnumid paare, mis on saadud sama krüpteerimisvõtmega

5

Sümmeetrilise krüptosüsteemi põhialused 4

- Claude Elwood Shannon – ameerika insener ja matemaatik, informatsiooniteooria looja, andis samuti suure panuse automaatide teooriasse, ning süsteemide juhtimise teooriasse (küberneetika)
- Tõi sisse mõisted hajutamine (diffusion) ja segamine (confusion) krüpto-algoritmite kindluse kirjeldamiseks
  - Hajutamine: Kui me ei taha, et oleks võimalik teades krüptogrammi "tuletada" võtit, siis on oluline, et kui avatekstis on teatavad statistilised omadused, siis samad omadused poleks märgatavad krüpteeritud tekstis
  - Segamine: statistilise seose likvideerimine krüpteeritud teksti ja krüpteerimisvõtme vahel. Selleks peaks see seos olema kindlasti mittelineaarne ning võimalikult keeruline. Et oleks raske teha järeldusi võtme kohta, kui on olemas krüptogramm ja võtme fragment (keerulised algoritmid tähtede asendamiseks). Lihtsaim viis on ümberpaigutus.

30. Aprill 1916 –
24. Veebruar 2001

6

Sümmeetrilise krüptosüsteemi põhialused 4.1.

- Kuidas saavutada hajutatust?
  - Iga element (täht) avatekstis peab mõjutama palju elemente (tähte) krüptogrammis ⇔ Iga element krüptogrammis on seotud paljude elementidega avatekstis
  - Seos avateksti ja krüpteeritud teksti vahel peab olema maksimaalselt keeruline, et oleks maksimaalselt keeruline tuletada võtit
  - Näide: kui tahame saada krüpteeritud tähe y_n, siis peame liitma *k* tähte avatud tekstis.

$$y_n = \sum_{i=1}^{k} m_{n+i} \ (mod\ 26)$$

  - Lihtsaim viis on transpositsioon

7

Sümmeetrilise krüptosüsteemi põhialused 5

- Kui X on avatekst, Y on krüptogramm ning K on krüpteerimisvõti, siis $Y = E_K[X]$ ning $X = D_K[Y]$ (kus E_K on võtit K kasutav krüpteerimisfunktisoon ja D_K on võtit K kasutav dekrüpteerimisfunktisoon)

  Vastane, kes ei tea X ega K, peab teada saama kas X või K või mõlemad
- Sümmeetrilised krüptoalgoritme saab eristada meetodite kaupa , mis töötlevad avateksti
  - Plokkide kaupa (block cipher). Teksti jaotakse plokkideks (tavaliselt 64 või 128 baiti) ning siis šifreeritakse iga plokk ükshaaval krüpteerimisvõtme abil
  - Jadana (stream cipher). Iga sümbol šifreeritakse lähtudes mitte ainult võtmest, vaid ka lähtudes selle sümboli asukohast avatekstis

8

Plokkšiffer

- Plokkšiffer koosneb kahest algoritmist: šifreerimise algoritm E ja dešifreerimise algoritm E^{-1}.
- Sisendsignaalideks on blokk suurusega *n* bitti ning *b*-biti pikkune võti K.
- Iga fikseeritud võtme juures, on dešifreerimise funktsiooniks on šifreerimise funktsiooni E pöördfunktsioon E^{-1}

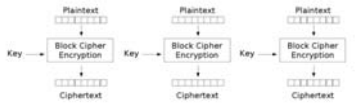
$$E_k^{-1}\left(E_k(M)\right) = M$$

- E_K on iga võtme jaoks on *n*-biti asendus (substitutsioon)
- Ploki suurus on tavaliselt 64 või 128
- Plokkšifrid rakendavad nii segamist kui hajutamist

9

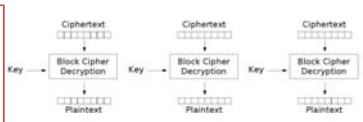
Plokkšifrite töörežiimid 1

- Koodiraamat (Electronic Codebook Mode, ECM) – lihtsaim režiim. Iga plokk avatekstis šifreeritakse eraldi, sõltumata.



Electronic Codebook (ECB) mode encryption

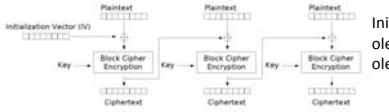
pilt ECM Kuidas peaks



Electronic Codebook (ECB) mode decryption[10]

Plokkšifrite töörežiimid 2

- Ahelrežiim (Cipher Block Chaining Mode, CBC) – iga plokk liidetakse eelmise šifreeritud plokiga loogilise XOR operaatoriga



Initsieerimise vektor ei pea olema salastatud, kuid peab olema unikaalne

Cipher Block Chaining (CBC) mode encryption

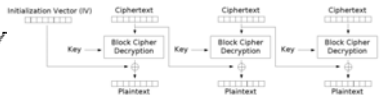
$C_0 = IV$

Krüpteerimine:

$$C_i = E_K(P_i \oplus C_{i-1}), C_0 = IV$$

Dekrüpteerimine:

$$P_i = D_K(C_i) \oplus C_{i-1}, C_0 = IV$$

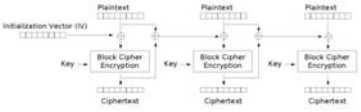


Cipher Block Chaining (CBC) mode decryption

11

Plokkšifrite töörežiimid 3

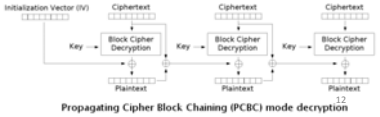
- PCBC (Propagating cipher-block chaining)



Propagating Cipher Block Chaining (PCBC) mode encryption

$$C_i = E_K(P_i \oplus P_{i-1} \oplus C_{i-1}), P_0 \oplus C_0 = IV$$

$$P_i = D_K(C_i) \oplus P_{i-1} \oplus C_{i-1}, P_0 \oplus C_0 = IV$$

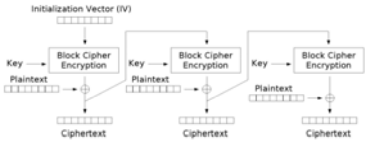


Propagating Cipher Block Chaining (PCBC) mode decryption

12

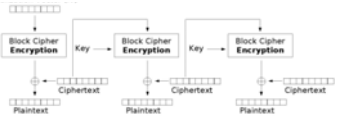
Plokkšifrite töörežiimid 4

- CFB (k-bit Cipher Feedback Mode)



Cipher Feedback (CFB) mode encryption

$$C_i = E_K(C_{i-1}) \oplus P_i$$
$$P_i = E_K(C_{i-1}) \oplus C_i$$
$$C_0 = \text{IV}$$

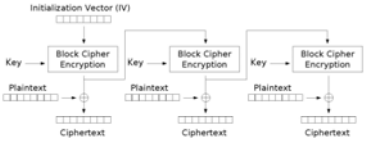


Cipher Feedback (CFB) mode decryption

13

Plokkšifrite töörežiimid 5

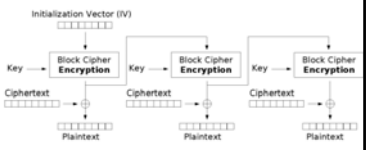
- OFB (Output Feedback)



Output Feedback (OFB) mode encryption

$$C_i = P_i \oplus O_i$$
$$P_i = C_i \oplus O_i$$

$O_i = E_K(O_{i-1})$
$O_0 = IV$



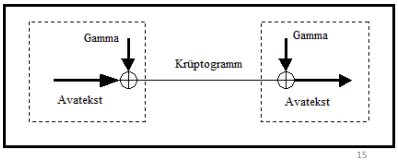
Output Feedback (OFB) mode decryption

14

Jadašiffer

- Algoritmid on kiired
- Kasutatakse ainult lihtsaid operatsioone, nagu liitmine ja korrutamine
- Saab kergesti analüüsida, kasutades algebralisi meetodeid
- Kasutatakse näiteks kõne, video, võrguliikluse krüpteerimiseks või muu sellise info korral, mille pikkus pole eelnevalt teada

Jadašifrites kasutatakse peamiselt segamist, kuigi mõned tagasisidega lahendused võimaldavad ka hajutamist

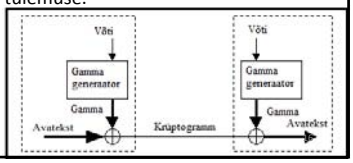


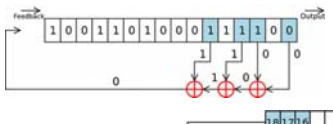
15

Sünkroonsed jadašifrid

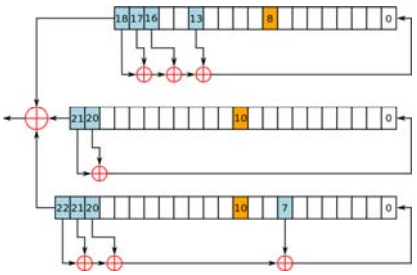
- Sünkroonsed jadašifrid. Krüpteerimisvõtmed genereeritakse sõltumata avatekstist ning krüptogrammist.
  - Sünkroniseerimise nõue, ehk võtmete jada peab šifreerijal ning dešifreerijal olema ühesugune
  - Üks viga ülekandmisel, et mõjuta järgmiste sümbolite dešifreerimist
  - Aktiivne rünne on kergesti tuvastatav, kuna sümboli lisamine või eemaldamine sõnumist mõjutab sünkroniseerimist. Teksti dešifreerimine annab vale tulemuse.

Gamma on genereeritud pseudo-juhuarvu generaatoriga, sõltub võtmest, 0 ja 1 peavad olema võrdse tõenäosusega, ehk 0.5





Linear feedback shift register

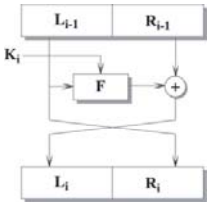


The operation of the keystream generator in A5/1, a LFSR-based stream cipher used to encrypt mobile phone conversations.

17

Feistel'i võrgud

- Sisend jagatakse teatud (2^n) pikkusega plokkideks
- Iga plokk jagatakse paremaks (R_i) ja vasakuks (L_i) osaks.
- L_0 muudetakse/krüpteeritakse funktsiooniga $f(L_i,K_i)$ võtme K_i abil, ning seejärel liidetakse (XOR) selle plokiga (R_i)
- Liitmise tulemus paigutatakse plokki L_{i+1} ning L_i paigutatakse tervikuna, muutmata kujul plokki R_{i+1}



Dešifreerimiseks kasutatakse sama algoritmi, krüpteeritud tekst sisestatakse ning võtmeid kasutatakse tagurpidises järjekorras

18

Data Encryption Standard

- 15 mai 1973 esitas USA Federal Register NBS nõudmised võimalikule krüpterimis-algoritmile, mille saaks võta standardiks:
- Algoritm peab võimaldama kõrge taseme turvalisust
- Algoritm peab olema täielikult määratud ja kergesti arusaadav
- Algoritmi turvalisus peab baseeruma võtmele ning ei pea salastama algoritmi ennast
- Algoritm peab olema kättesaadav igale kasutajale
- Algoritm peab olema paindlik ning peab võimaldama kohandamist erinevatele eesmärkidele
- Algoritm peab võimaldama ökonoomset riistvaralist realisatsiooni
- Algoritm peab olema efektiivne kasutamisel
- Algoritm peab võimaldama kontrollimist
- Algoritm peab olema lubatud ekspordiks

19

Data Encryption Standard

- **Lucifer**

Roy Adler
Don Coppersmith
Horst Feistel
Edna Crossman
Alan Hoffman
Bryant Tuckerman
Carl Meyer
Bill Notz
Lynn Smith
Walt Tuchman

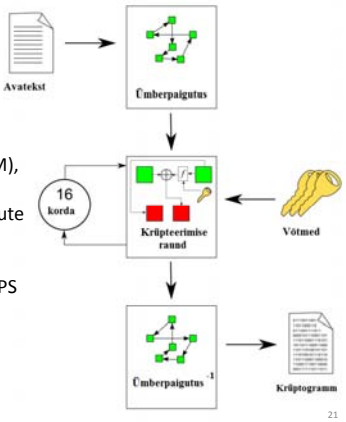
Horst Feistel, (January 30, 1915–November 14, 1990)

20

Data Encryption Standard

Töötatud välja 1977a. (IBM), on võetud
1980 NIST (National Institute of Standards and Technology) poolt standardiks Ameerikas (FIPS PUB 46)

Ploki pikkus 64 bitti
Võtme pikkus 56 bitti
16 raundi



21

DES

- Kõigepealt transformeeritakse tekst HEX (16-nd süsteemi) koodiks.
- Kasutatakse ANSI X3.4 (1968), millest arendati välja US ASCII kodeerimise standart
- Seejärel teisendatakse saadud kood kahendkoodiks
- Näiteks sõna TERE on 16-süsteemis 54 45 52 45; kahendkoodis aga: 01010100 01000101 01010010 01000101

ASCII Code Chart

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

DES

Iga ploki pikkus peab olema 64 bitti, ehk iga sõna pikkus on 8 tähte (= 8 Bait = 64 bitti)

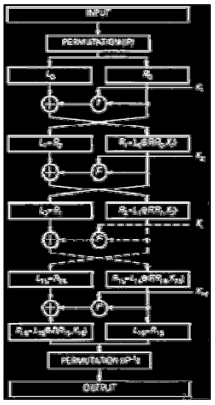
Permutatsiooni jaoks kasutatakse standardset tabelit (see bitt, mis oli kohal 58 läheb esimeseks, see mis oli kohal 50 teiseks jne:

IP

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

$L_i = R_{i-1}$

$R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$



DES

Võtme ettevalmistamine:

- Esialgne võti on HEX kujul: **K** = 133457799BBCDFF1
- Teisendame kahendkoodiks:

K = 00010011 00110100 01010111 01111001 10011011 10111100 11011111 11110001

Kasutame ümberpaigutust (koos teatud bittide väljajätmisega) standardse tabeli alusel:

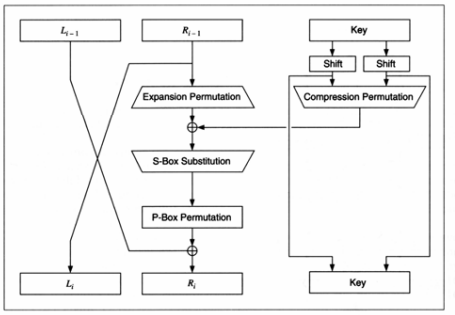
PC-1

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

Tulemuseks on 56 bitine permutatsioon:
K+ = 1111000 0110011 0010101 0101111 0101010 1011001 1001111 0001111

24

DES



25

DES

Võtme ettevalmistamine (jätk)

- 56 bitine võti jagatakse kaheks 28 bitiseks osaks:

C_0 = 1111000 0110011 0010101 0101111

D_0 = 0101010 1011001 1001111 0001111

- Moodustatakse 16 alamvõtit, kasutades registri nihutamist vastava tabeli abil

- Näiteks saame esimesel sammul:

C_1 = 1110000110011001010101011111

D_1 = 1010101011001100111100011110

Iteration Number	Number of Left Shifts
1	1
2	1
3	2
4	2
5	2
6	2
7	2
8	2
9	1
10	2
11	2
12	2
13	2
14	2
15	2
16	1

26

DES

- **Võtme ettevalmistamine (jätk)**
- Igat alamvõtit C_nD_n lühendatakse 48 bitini vastava permutatsioonitabeli abil:

PC-2

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

Tulemuseks on 16 alamvõtit $K_1 \ldots K_{16}$

27

DES

- Meil on olemas esialgne tekst, pikkusega 64 bitti, mis on läbinud permutatsiooni **IP**. See jagatakse vasakuks ja paremaks võrdse pikkusega osaks

Näiteks kui

IP = 1100 1100 0000 0000 1100 1100 1111 1111 1111 0000 1010 1010 1111 0000 1010 1010

Siis saame

L_0 = 1100 1100 0000 0000 1100 1100 1111 1111

R_0 = 1111 0000 1010 1010 1111 0000 1010 1010

Edasi kasutame valemeid

$L_n = R_{n-1}$

$R_n = L_{n-1} \oplus f(R_{n-1}, K_n)$

28

DES

Selleks, et arvutada *f* on vaja 32 bitise R "laiendada" 48 bitini

E BIT-SELECTION TABLE

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

$K_n \oplus \mathbf{E}(R_{n-1})$

K_1 = 000110 110000 001011 101111 111111 000111 000001 110010

$\mathbf{E}(R_0)$ = 011110 100001 010101 010101 011110 100001 010101 010101

$K_1 \oplus \mathbf{E}(R_0)$ = 011000 010001 011110 111010 100001 100110 010100 100111

Tulemuse esitame kui:

$K_n \oplus \mathbf{E}(R_{n-1}) = B_1B_2B_3B_4B_5B_6B_7B_8$

Ning suuname S-boxi

$S_1(B_1)S_2(B_2)S_3(B_3)S_4(B_4)S_5(B_5)S_6(B_6)S_7(B_7)S_8(B_8)$

29

DES. S-boxes (asenduste kast)

- Saab sisendiks *m* bitti ning annab väljundis *n* bitti
- DES algoritmis on 8 plokki
- Näiteks olgu tegu arvuga : $\boldsymbol{B}$ = 011011. Esimene ja viimane bitt määravad rea numbri, neli keskmist bitti tulba numbri. Ehk tulp on 01 ja rida on 1101. Sisendiks oli 6 bitti, tulemuseks saame 4 bitti

S_5		Middle (inner) 4 bits of input															
		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
Outer bits	00	0010	1100	0100	0001	0111	1010	1011	0110	1000	0101	0011	1111	1101	0000	1110	1001
	01	1110	1011	0010	1100	0100	0111	1101	0001	0101	0000	1111	1010	0011	1001	1000	0110
	10	0100	0010	0001	1011	1010	1101	0111	1000	1111	1001	1100	0101	0110	0011	0000	1110
	11	1011	1000	1100	0111	0001	1110	0010	1101	0110	1111	0000	1001	1010	0100	0101	0011

$S_1(B_1)S_2(B_2)S_3(B_3)S_4(B_4)S_5(B_5)S_6(B_6)S_7(B_7)S_8(B_8)$ =

= 0101 1100 1000 0010 1011 0101 1001 0111

30

DES

Viimane samm *f* arvutamisel:
$f = P(S_1(B_1)S_2(B_2)...S_8(B_8))$
f = 0010 0011 0100 1010 1010 1001 1011 1011
$R_1 = L_0 \oplus f(R_0, K_1)$

= 1100 1100 0000 0000 1100 1100 1111 1111
⊕ 0010 0011 0100 1010 1010 1001 1011 1011
= 1110 1111 0100 1010 0110 0101 0100 0100

Permutation P

16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

Kui 16 raundi on läbitud $(R_{16}L_{16})$ siis läbitakse viimane permutatsioon

Sõnum on krüpteeritud

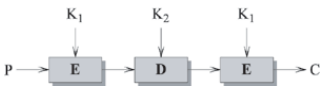
Final Permutation (IP**-1)

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

31

DES ja kolmekordne DES

- DES. Kokku on võimalik moodustada 2^{56} võtit. Tänapäeval ei ole selle murdmine probleem.
- Kolmekordne DES kahe võtmega. Sel juhul 2^{168}



32

DES

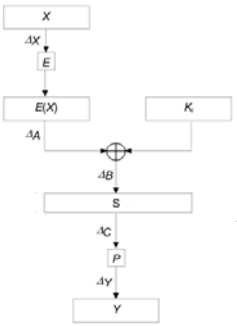
- DES on realiseeritud nii ristvaraliselt kui ka programmselt
- Nõrgad võtmed:
  - 00000000 00000000
  - 00000000 FFFFFFFF
  - FFFFFFFF 00000000
  - FFFFFFFF FFFFFFFF

Masin, mis suutis murda DES aastal 1993 maksis 1 million dollarit, murdis 3,5 tunniga.

33

Diferentsiaalne krüptoanalüüs

- Krüptoanalüüsi eesmärgiks on koodi murdmine
- 1990 a
- Analüüsi eemärgiks tuvastada raundi võti
- Vaatleb erinevusi avatekstides ΔX ning võrdleb erinevusega , mis on saadud peale raundi šifreerimist ΔY. Otsitakse võtmete tõenäosust



Eli Biham

Adi Shamir

$$\Delta X \xrightarrow{P} \Delta Y$$

$\Delta X = X_1 \oplus X_2 \quad \Delta Y = Y_1 \oplus Y_2$

34

Lineaarne krüptoanalüüs

- 1993 a Mitsuru Matsui
- Otsida lineaarseid võrrandeid avateksti, krüpteeritud teksti ja võtme vahel
- Nende seoste kasutamine koos paaridega (avatekst-krüptogramm) võtme bittide tuletamiseks.

Lineaarne aproksimeerimine

$$P_{i1} \oplus P_{i2} \oplus \ldots \oplus P_{ia} \oplus C_{j1} \oplus C_{j2} \oplus \ldots \oplus C_{jb} = K_{k1} \oplus K_{k2} \oplus \ldots \oplus K_{kc}, (1)$$

Tõenäosus, et selline võrrand on õige on ½, kui aga see erineb palju ½-st (mida rohkem, seda parem), siis seda võrrandit saab kasutada krüptoalgoritmi murdmiseks

Erijuhtum:

$$C_{j1} \oplus C_{j2} \oplus \ldots \oplus C_{jb} = K_{k1} \oplus K_{k2} \oplus \ldots \oplus K_{kc}.$$

35