Peatükk 3

Plokkšifrid

3.1 Plokkšifrite töörežiimid

3.1.1 Koodiraamatu režiim

Töörežiimid esitame pseudokeelsete algoritmidena, mis saavad ette teatud hulk sisendparameetreid. Sealjuures, $E_k(x)$ ($E_k^{-1}(x)$) on konkreetse plokkšifri (de)-krüpteerimisfunktsioon, p on avatekst, c on krüptogramm ning l on plokkide arv. p_i ning c_i-ga tähistame vastavalt avateksti ning krüptogrammi i-ndat plokki. Vektorit $(0, \ldots, 1, \ldots, 0)$, mille ainus ühene bitt on i-s, tähistame e_i.

Esimesena vaatame kõige lihtsamat, koodiraamatu režiimi (ingliskeelne lühend ECB).

Algoritm 3.1.1

<u>**funct**</u> *crypt_ecb*$(E, k, p, c, l) \equiv$
  <u>**for**</u> $i := 0$ <u>**to**</u> $l - 1$ <u>**do**</u>
    $c_i := E_k(p_i)$
  <u>**od**</u>
.

<u>**funct**</u> *decrypt_ecb*$(E, k, p, c, l) \equiv$
  <u>**for**</u> $i := 0$ <u>**to**</u> $l - 1$ <u>**do**</u>
    $p_i := E_k^{-1}(c_i)$
  <u>**od**</u>
.

Koodiraamatu režiimi omadused:

1. Identsed avatekstid: Avateksti identsed plokid krüptuvad sama võtme abil identseteks krüptogrammi plokkideks.

2. Ahelsõltuvused: plokid krüptitakse sõltumatult teistest plokkidest. Avateksti plokkide ümberjärjestamisel järjestuvad vastavalt ümber krüptogrammi plokid.

3. Vealevi: ühe või enama biti viga ühesainsas krüptogrammi plokis mõjutab ainult sellesama krüptogrammi dekrüptimist. Tüüpilise šifri E korral dekrüptub vigane plokk juhuslikult (50% saadud avateksti bittidest on valed).

Kuna krüptogrammi plokid on sõltumatud, siis krüptogrammide plokkide kuritahtlikud asendused (näiteks tihti esineva ploki lisamised) ei mõjuta kõrvalolevate plokkide dekrüptimist. Kuna avateksti identsed plokid krüptuvad identseteks krüptogrammi plokkideks, siis ei soovitata koodiraamatu režiimi kasutada pikemate kui üheplokiste teadete saatmiseks (samuti ei soovitata koodiraamatu režiimis ühe võtmega saata rohkem kui ühte üheplokist teadet). Turvalisust saab mõnevõrra tõsta lisades igasse plokki mõned juhuslikud bitid.

3.1.2 Ahelrežiim

Ahelrežiimi ingliskeelseks lühendiks on CBC (*cipher block chaining*). Lisasisendiks võrreldes koodiraamatu režiimiga on iv (**algväärtus**, *initial value*).

Algoritm 3.1.2

funct $crypt\_cbc(E, k, p, c, l, iv) \equiv$
  for $i := 0$ **to** $l - 1$ **do**
    $c_i := E_k(p_i \oplus iv)$
    $iv := c_i$
  od
.

funct $decrypt\_cbc(E, k, p, c, l, iv) \equiv$
  for $i := 0$ **to** $l - 1$ **do**
    $p_i := iv \oplus E_k^{-1}(c_i)$
    $iv := c_i$
  od
.

Koos krüptogrammiga c saadetakse teisele osapoolele edasi ka iv väärtus *enne* $crypt\_cbc$ väljakutsumist, iv väärtust *pärast* funktsioonist $crypt\_cbc$ naasmist kasutatakse paljudes rakendustes järgmise teate sama võtmega krüptimisel algväärtusena. Vastuvõtja rakendab saadud paarile (iv, c) alamprogrammi $decrypt\_cbc$ ega jäta iv väärtust meelde.

Ahelrežiimi omadusi:

1. Identsed avatekstid: krüptogrammi identsete plokkide dekrüptimisel saadakse identne tulemus kui krüptitakse sama võtme ja algväärtusega. Algväärtuse, võtme või avateksti esimese ploki muutmisel saadakse erinev krüptogramm.

2. Ahelsõltuvused: ahelrežiimis sõltub c_i p_i-st ning kõigist sellele eelnenud avateksti plokkidest (kõikidest eelnenud plokkidest sõltuvus on küll "peidetud" sõltuvusse krüptogrammi eelnevast plokist). Seega, krüptogrammi plokkide ümber järjestamine mõjutab oluliselt dekrüptimise tulemust; krüptogrammi korrektse ploki õige dekrüptimine eeldab krüptogrammi eelnenud ploki korrektsust.

3. Vealevi: ühebitine viga krüptogrammi plokis c_i mõjutab plokkide c_i ja c_{i+1} dekrüptimist (kuna p_i sõltub c_i-st ning c_{i-1}-st). Vigasest krüptogrammist $c_i^* := c_i \oplus e_s$ dekrüptimisel saadud avateksti plokk p_i^* on tüüpiliselt täiesti juhuslik (50% bittidest vigased), kuid samas $p_{i+1}^* = p_{i+1} \oplus e_s$. Seega muutes vajalikke bitte c_i-s, saavutab kurikael p_{i+j}-s ennustatavad bitimuutused.

4. Veaparandus: CBC režiim on isesünkroniseeruv. Vea korral, mis leiab aset plokis c_i (k.a. ploki kaotsiminek), kuid mitte plokis c_{i+1} dekrüptub c_{i+2} korrektselt x_{i+2}-ks. Samas, avateksti x_i bittide muutmine muudab krüptogrammi kõiki järgnevaid plokke (see mõjutab ahelrežiimi kasutatavust rakendustes, milles läheb vaja juhupöördust andmete poole).

Harjutus 3.1.3 *Üks hea plokkšifri kriteeriume on see, et šiffer "näeks välja" nagu tõeliselt juhuslik funktsioon. Näidata, et juhuslik funktsioon rahuldab tingimust "vigasest krüptogrammist* $c_i^* := c_i \oplus e_s$ *dekrüptimisel saadud avateksti plokk* p_i^* *on tüüpiliselt täiesti juhuslik (50% bittidest vigased)".*

Märkus 3.1.4 *Plokkide piire muutvat bittide kaotust ei võimalda sünkroniseerida ükski töörežiim.*

Algväärtus *iv* ei pea olema salajane, kuid tema autentsus peab olema tagatud: *iv* muutmine võimaldab kurikaelal teha dekrüptitud avateksti plokis ennustatavaid muudatusi. Autentsuse tagamiseks tuleb kasutada vastavaid krüptograafia primitiive.

3.2 Protokollide teooria algtõed

Demonstreerime järgnevalt erinevate töörežiimide mõju, tuues mitteformaalselt sisse **krüptograafilise protokolli** mõiste (formaalne käsitlus tuleb edaspidi). Samas demonstreerime esimest korda krüptograafiliste primitiivide koos kasutamise metoodikat.

Igal krüptograafilisel protokollil on fikseeritud arv osapooli P_i (vaata paragrahvi 1.2.2 terminoloogiat!), kes on volitatud informatsiooni saatma, vastu võtma ning manipuleerima. Peale nende on olemas ka volitamata osapool E (kurikael). Protokolli puhul on fikseeritud teatud tingimused ning turvaeesmärk, mille protokoll peab saavutama antud tingimustel. Tingimusteks on näiteks:

1. Millised osapooled teavad protokolli alguses mida.
2. Kurikaela E poolt võimalikud ründeliigid.
3. Nõuded, mida protokolli jooksul kasutatavad krüptograafilised primitiivid peavad rahuldama.

Suurust $e_k(p)$ (kus $e \in \mathcal{E}$) tähistame protokollide juures lühendatult $\{p\}_k$, suurust $d_\ell(c)$ (kus (k, ℓ) on võtmepaar ning $d \in \mathcal{D}$) aga $\{c\}_{k^{-1}}$. Protokollide formaalses turvaanalüüsis jäetakse tähelepanu alt välja pisidetailid, näiteks mõeldakse $\{f_1, \ldots, f_n\}_k$ all järjestatud hulgast $(f_1, \ldots, f_n)$ mingi kodeeringu võtmega k krüptimisel saadud krüptogrammi. Eeldatakse, et kodeering rahuldab turvaeesmärke.

Protokoll koosneb raundidest $1, 2, \ldots, r$; r on antud protokolli raundide arv. Protokolli raundi i jooksul saadab osapool S_i osapoolele R_i teate M_i. Kõige lihtsamaks protokolliks on $A \xrightarrow{1} B \colon X$ (osapool A saadab osapoolele B edasi teate X, sellega protokoll lõpeb). Tavaliselt aga ei piirdu protokollid ühe sammuga. Toome sisse mõned loogilised sümbolid oma arutelude lihtsustamiseks.

Definitsioon 3.2.1 *Ütleme, et $P \ni X \Leftrightarrow$ osapool P teab suurust X või suudab seda välja arvutada teistest talle teadaolevatest suurustest. Ütleme, et $P|\!\!\equiv\phi \Leftrightarrow$ osapool P usub, et valem ϕ on tõene. Tähistame lühidalt $\langle P_1, \ldots, P_j\rangle_K := \forall E(E \ni K \supset E = P_1 \vee \cdots \vee E = P_j)$. Ütleme, et $\sharp X$ kui X on värske, ehk ükski osapool pole X-i teisele saatnud enne käimasoleva käitustiiru algust.*

Kasutades neid sümboleid, saame kirja panna oma protokolli eeldused ning järeldused. Protokolli iga sammu $S_i \xrightarrow{i} R_i : M_i$ eel tuleb tõestada, et $S_i \ni M_i$. Täpsemalt räägime sellest kõigest hiljem, kuid esialgu paar näidet.

Shamiri protokoll

Shamiri protokollil on kaks volitatud osapoolt A ja B, A-l on salajane võti K_A ning B-l salajane võti K_B (ehk $A \ni K_A$, $\langle A\rangle_{K_A}$, $B \ni K_B$, $\langle B\rangle_{K_B}$). Ühisteadmisi algul ei ole. Protokolli alguses $A \ni p$, mida ta soovib B-le edastada. Kui krüptimisfunktsioon E rahuldab tingimust $\{\{p\}_{k_1}\}_{k_2} = \{\{p\}_{k_2}\}_{k_1}$ suvalise kahe võtme $k_1, k_2 \in \mathcal{K}$ ning suvalise avateksti $p \in \mathcal{P}$ korral (lisaks veel sümmeetriliste krüptosüsteemide puhul tavaline tingimus, et e ja tema pöördväärtus e^{-1} oleksid ökonoomselt arvutatavad: $\forall E \forall p \forall k(E \ni k \wedge E \ni p \supset E \ni \{p\}_k \wedge E \ni \{p\}_{k^{-1}})$), saavutame turvaeesmärgi järgmise protokolliga:

Protokoll 3.2.2
$$\begin{aligned} A &\xrightarrow{1} B : \{p\}_{K_A} \\ B &\xrightarrow{2} A : \{p\}_{K_A K_B} \\ A &\xrightarrow{3} B : \{p\}_{K_B} \end{aligned}$$

Märkus 3.2.3 *Siinkohal eeldame, et kurikaelal E on kanalile vaid passiivne ligipääs: ta saab andmeid lugeda, aga mitte nendega manipuleerida. Aktiivselt ründava vastase vastu Shamiri protokoll niivõrd üldisel kujul kaitset ei paku.*

Kuna $A \ni p$ ja $A \ni K_A$, siis ka $A \ni \{p\}_{K_A} (=: M_1)$, mille ta saadab esimesel sammul B-le. Seejärel $B \ni M_1$. Kuna ka $B \ni K_B$, siis $B \ni \{p\}_{K_A K_B} (=: M_2)$, mille ta edastab teisel sammul A-le. Seejärel $A \ni \{p\}_{K_A K_B}$. Kuna ka $A \ni K_A$, siis $A \ni \{p\}_{K_A K_B K_A^{-1}} (=: M_3)$. Funktsiooni e valiku järgi $M_3 = \{p\}_{K_A K_A^{-1} K_B} = \{p\}_{K_B}$ (NB! sealjuures *ei tea* A võtit K_B); A saadab kolmandal sammul B-le teate M_3. Seejärel $B \ni \{p\}_{K_B}$. Kuna ka $B \ni K_B$, siis $B \ni p$.

Harjutus 3.2.4 *Shamiri protokolli krüptimisfunktsioon E peab olema väga hästi valitud. Olgu $\mathcal{K} = \mathcal{P} = \mathcal{C}$ ning $E_e(p) := e \oplus p$, näidata, et sel juhul teab kurikael protokolli lõppedes väärtust p.*

Protokoll ELBE2

On kaks volitatud osapoolt, C (klient) ning S (server), ning $\langle C, S\rangle_K$ (pikaajaline võti). Krüptimiseks kasutatakse suvalist plokkšifrit. Klient soovib serveriga alustada sideseanssi, mille jooksul krüptitakse andmed nn seansivõtmega n, mille genereerib S ning saadab võtmevahetusprotokolli jooksul kliendile. Protokolli lõpus peavad mõlemad pooled olema veendunud, et teine pool oskab krüptida võtmega K (autentsus), samuti peab protokolli lõpus kehtima $\langle C,S\rangle_n \wedge \langle C,S\rangle_K$ (salastus). Protokoll ise on selline:

Protokoll 3.2.5
$$\begin{aligned} &C \xrightarrow{1} S\colon r \\ &S \xrightarrow{2} C\colon \{r,n\}_K \\ &C \xrightarrow{3} S\colon \{n\}_K \end{aligned}$$

Protokolli alguses genereerib C juhuarvu $M_1 = r$ ($\Rightarrow C \ni M_1$) ning saadab selle serverile. Seejärel $S \ni r$. Server genereerib seansivõtme n ($\Rightarrow S \ni n$). Kuna ka $S \ni K$, siis $S \ni \{r,n\}_K (=: M_2)$ ja saadab selle kliendile. Seejärel $C \ni \{r,n\}_K$. Kuna ka $C \ni K$, siis $C \ni (r,n)$. Nüüd kontrollib klient, kas saadud r on tõepoolest sama r, mille ta ise saatis. Kui on, siis $C {\models} S \ni K$ (st et ta räägib tõepoolest õige serveriga) ning edastab serverile teate $M_3 := \{n\}_K$. Seejärel $S \ni \{n\}_K$. Kuna ka $S \ni K$, siis $S \ni n$. Kui nii saadud n võrdub eelnevalt saadetud n-ga, siis $S {\models} C \ni K$ (st et n sai vahetatud õige kliendiga) ning alustab C-ga seanssi, krüptides edaspidi andmeid võtmega n.

See tähendab, et protokoll ise on korrektne ning saavutab oma eesmärgid (salastus saavutati tänu sellele, et vahepeal saadeti kõik andmed edasi krüptituna K-ga, mis oli protokolli alguses salajane ning mille väärtus protokolli käigus ei lekkinud). Samas, protokolli realisatsioon peab olema hoolega läbi mõeldud.

Olgu plokkšifri ploki pikkus b, võtme pikkus $2b$ ning $|r| = b$. Tähistame $n = (n_0, n_1)$ (konkatenatsioon), kus $|n_0| = |n_1| = b$, st et protokolli saab üles kirjutada järgmiselt:
$$\begin{aligned} &C \xrightarrow{1} S\colon r \\ &S \xrightarrow{2} C\colon \{r,n_0,n_1\}_K \\ &C \xrightarrow{3} S\colon \{n_0,n_1\}_K \end{aligned}$$
Siin on sulgudes iga üksik element täpselt ploki pikkune. Kui krüptimine käiks koodiraamatu režiimis, siis oleks protokoll lihtsalt järgmine:

Protokoll 3.2.6
$$\begin{aligned} &C \xrightarrow{1} S\colon r \\ &S \xrightarrow{2} C\colon (\{r\}_K, \{n_0\}_K, \{n_1\}_K) \\ &C \xrightarrow{3} S\colon (\{n_0\}_K, \{n_1\}_K) \end{aligned}$$

Protokolli 3.2.6 vastu leidub lihtne rünne. Kurikael E loeb kanalilt esimese ja teise sammu tulemused, $E \ni r$, $E \ni (\{r\}_K, \{n_0\}_K, \{n_1\}_K)$, seega ka $E \ni (\{n_0\}_K, \{n_1\}_K)$. Seejärel peale imiteerib E kolmandal sammul klienti (tähistame $E[C]$) ning saadab serverile tagasi paari $(\{n_0\}_K, \{n_1\}_K)$. Server krüptib saadud paari lahti, kontroll õnnestub. Seega $S {\models} E \ni K$ ning alustab C-ga seanssi, kasutades võtit n. Kuna aga $\neg(C \ni n)$, on protokolli turvaeesmärgid rikutud. Ründe saab üles kirjutada järgmiselt:

Rünne 3.2.7
$$\begin{aligned} &C \xrightarrow{1} S\colon r \\ &S \xrightarrow{2} E[C]\colon (\{r\}_K, \{n_0\}_K, \{n_1\}_K) \\ &E[C] \xrightarrow{3} S\colon (\{n_0\}_K, \{n_1\}_K) \end{aligned}$$

Leidub palju ohtlikum rünne, kus E saab kliendile peale sundida tema poolt valitud võtme, nii et too jääb uskuma, et $E = S$ (siin tähistame täpikesega neid andmeid, mida volitatud osapooled saadavad kurikaelale ning mida too edaspidi ei kasuta):

Protokoll 3.2.8

$$\begin{array}{ll}
1 & C \xrightarrow{1} E[S]\colon r \\
2 & E[C] \xrightarrow{1} S\colon x \\
3 & S \xrightarrow{2} E[C]\colon (\{x\}_K, \cdot, \cdot) \\
4 & E[C] \xrightarrow{1} S\colon y \\
5 & S \xrightarrow{2} E[C]\colon (\{y\}_K, \cdot, \cdot) \\
6 & E[C] \xrightarrow{2} S\colon r \\
7 & S \xrightarrow{2} E[C]\colon (\{r\}_K, \cdot, \cdot) \\
8 & E[S] \xrightarrow{2} C\colon (\{r\}_K, \{x\}_K, \{y\}_K) \\
9 & C \xrightarrow{3} E[S]\colon (\{x\}_K, \{y\}_K)
\end{array}$$

Seletus. Kui Edgar (E) näeb, et C saadab S-le juhuarvu r (samm 1), saadab ta serverile kaks juhuarvu x ning y, millele server vastab kolmikutega, mis sisaldavad neid juhuarve krüptitud kujul. Sammul 6 edastab Edgar serverile eelnevalt tallele pandud arvu r, saades tagasi väärtuse $\{r\}_K$. Sammul 8 saadab Edgar kliendile kolmiku $(\{r\}_K, \{x\}_K, \{y\}_K)$. Kliendi kontroll õnnestub: $C \models E[S] \ni r$ (ehk et $E = S$) ning pärast viimast sammu võib Edgar edukalt simuleerida serverit, kasutades võtit $n = (x, y)$.

Paneme tähele, et antud juhul on tegu *teadaoleva avatekstigaründega*: sammude $2 - 5$ järel teab Edgar kahte paari $(x, \{x\}_K)$ ning $(y, \{y\}_K)$. Ta saab küll arve x ja y ise valida, kuid antud ründe jaoks kõlbavad suvalised x ja y. Kuna protokolli iga käitustiiru jooksul saab Edgar teada paari $(r, \{r\}_K)$, ei pea ta samme $2 - 5$ ilmutatud kujul rakendama: piisab, kui ta on eelnevalt pealt kuulanud kahte erineva r-ga käitustiiru.

Definitsioon 3.2.9 *Kui teadaoleva avatekstiga ründe (vastavalt valitava avatekstiga ründe/valitava krüptogrammiga ründe) edukaks sooritamiseks on vaja n erinevat paari $(x_i, \{x_i\}_K)$, siis öeldakse, et selle ründe* ***andmekeerukus*** *on n.*

Seega ründe 3.2.8 andmekeerukuseks on 2.

Harjutus 3.2.10 *Teostada protokolli 3.2.6 vastu rünne andmekeerukusega 1. Kui palju on turvaeesmärgid rikutud sellise ründe puhul?*

Vaatleme järgmisena ahelrežiimi. Oletame lihtsuse mõttes, et kolmas teade saadetakse edasi algväärtusega IV=0 (mis jäetakse edasi saatmata), teise sammu jaoks genereerib S juhuarvu I.

Protokoll näeb välja selline:

Protokoll 3.2.11

$$\begin{array}{l}
C \xrightarrow{1} S\colon r \\
S \xrightarrow{2} C\colon (I, \{r \oplus I\}_K, \{\{r \oplus I\}_K \oplus n_0\}_K, \{\{\{r \oplus I\}_K \oplus n_0\}_K \oplus n_1\}_K) \\
C \xrightarrow{3} S\colon (\{n_0\}_K, \{\{n_0\}_K \oplus n_1\}_K)
\end{array}$$

Harjutus 3.2.12 *Leida rünne protokolli 3.2.11 vastu, kus C jääb uskuma, et $E = S$ ning hakkab E-ga andmeid vahetama E poolt pealesurutud võtmega.*

Analüüsida juhtu, kus protokolli 3.2.11 teisel sammul saadetakse $\{r, n_0, n_1\}_K$ asemel edasi

1. $\{n_0, r, n_1\}_K = (I, \{n_0 \oplus I\}_K, \{\{n_0 \oplus I\}_K \oplus r\}_K, \{\{\{n_0 \oplus I\}_K \oplus r\}_K \oplus n_1\}_K)$
2. $\{n_0, n_1, r\}_K = \ldots$

Mille poolest on need variandid turvalisemad? Millised ründed ikkagi võimalikud on? Pakkuda välja lihtne täiendus protokollile 3.2.11, nii et uus protokoll saavutaks oma eesmärgid.

Kuivõrd turvaline on protokoll 3.2.11 andmekeerukusega 1 ja 2 rünnete vastu?

3.3 DES

3.3.1 Feisteli šifrid

Olgu $x \in \mathbf{B}^n$, $1 \leq i \leq j \leq n$. Defineerime $\mathrm{substr}(x, i, j)$ kui arvu, mille moodustavad arvu x bitid $i \ldots j$ (näiteks $\mathrm{substr}(100111, 3, 5) = 011$). Olgu šifris r raundi ning olgu K_i, $1 \leq i \leq r$ raundi i **alamvõti**, mis on tuletatud võtmest K **võtmelaiendusalgoritmi** abil. Šifri plokipikkuseks on t.

Krüptosüsteem 3.3.1 (Feistel'i šiffer) *Sisend: avatekst $p \in \mathbf{B}^t$, kus $2 \mid t$.*

```
funct feistel(p) ≡
  L := substr(p, 1, t/2)
  R := substr(p, 1 + t/2, t)
  for i := 1 to r do
      L' := L
      L := R
      R := L' ⊕ F(R, K_i)
  od
  feistel := R||L.
```

```
funct defeistel(c) ≡
  L := substr(c, 1, t/2)
  R := substr(c, 1 + t/2, t)
  for i := 1 to r do
      L' := L
      L := R
      R := L' ⊕ F(R, K_{r−i})
  od
  defeistel := R||L.
```

Joonis 3.1: Feistel'i šiffer

Harjutus 3.3.2 *Tõestada, et suvalise Feistel'i šifri korral*

$$\mathit{defeistel}(\mathit{feistel}(p)) = p.$$

Feisteli šifrid on näiteks DES ning FEAL; Feisteli šifrid erinevad omavahel suurustes r, F ning K_i. Mõne šifri (näiteks DES) korral on r ja t fikseeritud, mõne šifri korral antakse r ja t väärtus ette šifri parameetrina.

On olemas mitmeid nn kvaasi-Feisteli šifreid, mille operatsioon erineb vaid ühes elemendis: $R := L' \oplus F(R, K_i)$ asemel on võetud mõni teine pööratav kujutus. Nii näiteks šifris RC5 on vastavaks sammuks $R := (\mathrm{rol}(L' \oplus R, R)) + K_i$, kus $\mathrm{rol}(\alpha, \beta)$ tähistab suuruse α $\beta \pmod{t/2}$ bitist rotatsiooni vasakule $\beta \pmod{t/2}$ biti võrra.

3.3.2 DES

Enim kasutatud plokkšiffer (ja enim kasutatud krüptosüsteem üldse) on IBM'i teadlaste välja töötatud DES (*Data Encryption Standard*), mis on USA's olnud klassifitseerimata andmete kaitsestandardiks aastast 1977, NBS (National Bureau of Standards) on DES-i siiamaani ca iga viie aasta tagant üle vaadanud ja standardina uuendanud; viimati 1994. aasta jaanuaris aastani 1998. Üldine arvamus on, et järgmine kord standardit enam ei pikendata.

DES-i korral $\mathcal{P} = \mathcal{C} = \mathbf{B}^{64}$, $\mathcal{K} = \mathbf{B}^{56}$ (tegelikult on K 64-bitine, millest 56 bitti moodustavad võtme ning 8 bitti on paarsuskontrolliks (bitid 8, 16, ..., 64 valitakse nii, et igas baidis oleks paaritu arv ühtesid), kuid viimaseid ignoreeritakse järgnevas), sealjuures kasutatakse teatuid fikseeritud permutatsioone. Nii näiteks algpermutatsioon IP ja tema pöördpermutatsioon on esitatud joonisel 3.2 (x-i 58-s bitt on IP(x)'i esimene bitt, x-i 50-s bitt on IP(x)'i teine bitt jne).

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

Joonis 3.2: **Algpermutatsioon IP**

DES-i lühikirjeldus oleks selline.

funct *des_crypt*(p, z) $\equiv$
  $L := \mathrm{substr}(\mathrm{IP}(p), 1, 32)$
  $R := \mathrm{substr}(\mathrm{IP}(p), 33, 64)$
  for $i := 1$ **to** 16 **do**
    $L' := L$
    $L := R$
    $R := L' \oplus F(R, z_i)$
  od
  des_crypt $:= \mathrm{IP}^{-1}(R \| L)$.

E					
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

Joonis 3.3: **Laienduskujutus** E

S_1															
14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S_2															
15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

Joonis 3.4: S-kastid

Olgu $E : \mathbf{B}^{32} \to \mathbf{B}^{48}$ joonisel 3.3 esitatud permutatsioon. DES-is defineeritakse kaheksa **asendusmoodulit** (S-*box*), kus iga S_i on fikseeritud 4×16 maatriks (kahe esimese asendusmooduli tabelid on toodud joonisel 3.4, ülejäänusid võib leida suvalisest heast krüptoraamatust ([Sti95])). Asendusmooduli S_i väljund $S_i(B_j) \in \mathbf{B}^4$ (kus $B_j \in \mathbf{B}^6$) leitakse järgmiselt. Bitistring $b_1 b_6$ annab mooduli S_i rea r, bitistring $b_2 b_3 b_4 b_5$ aga veeru c. Olgu $C_j := S_j(B_j) := (S_j)_{rc}$, $1 \leq j \leq 8$.

Vaatleme funktsiooni $f : \mathbf{B}^{32} \times \mathbf{B}^{48} \to \mathbf{B}^{32}$ ja defineerime järgnevas funktsiooni F ($P \in \mathcal{S}_{32}$ on fikseeritud permutatsioon).

<u>**funct**</u> $F(R, J) \;\equiv$
   $B = E(R) \oplus J$
   <u>**for**</u> $i := 1$ <u>**to**</u> 8 <u>**do**</u>
       $C_i := S_i(\mathrm{substr}(B, 6i-5, 6i))$ <u>**od**</u>
   $C := C_1 || C_2 || C_3 || C_4 || C_5 || C_6 || C_7 || C_8$
   $F := P(C)$**.**

Defineerime alamvõtmete z_i-de moodustamise algvõtmest K. Võtmelaiendusalgoritmis kasutatakse fikseeritud permutatsioone PC-1 ja PC-2 ning tsüklilisi nihkeid vasakule j biti võrra, kus $j = 1$ kui $i \in \{1, 2, 9, 16\}$ ning $j = 2$ vastasel juhul.

<u>**proc**</u> *des_key_exp*$(K, z) \;\equiv$
   $K' := \text{PC-1}(K)$
   $C := \mathrm{substr}(K', 1, 28)$
   $D := \mathrm{substr}(K', 29, 56)$
   <u>**for**</u> $i := 1$ <u>**to**</u> 16 <u>**do**</u>
       $C_i := \mathrm{rol}(C, i)$
       $D_i := \mathrm{rol}(D, i)$
       $K_i := \text{PC-2}(C || D)$
   <u>**od**</u>

3.3.3 DES praktikas

3.3.4 DES'i krüptoanalüüs

Suvalise Feistel'i šifri juures on oluliseks nõrkuseks see, et igas raundis jääb pool plokist muutmata, seega kipub neis šifrites raunde 2 korda rohkem olema, kui teistes šifrites.

Erinevuste krüptoanalüüs

Lineaarne krüptoanalüüs

3.4 IDEA

IDEA esimene versioon toodi avalikkuse ette aastal 1990, artiklis [LM90] nime all **PES** (*Proposed Encryption Standard*). Esimese aasta jooksul leiti pakutud šifris olulisi vigu ning 1991. aastaks valmis täiendatud šiffer **IPES** (*Improved PES*), mida hiljem hakati kutsuma IDEA'ks (*International Data Encryption Algorithm*).

IDEA korral $\mathcal{P} = \mathcal{C} = \mathbf{B}^{64}$, $\mathcal{K} = \mathbf{B}^{128}$, st et kõikvõimalikke erinevaid võtmeid on 2^{128}.

Harjutus 3.4.1 *Arvutada, kui kaua aega võtaks tänapäeval eksisteerivatel arvutitel IDEA võtmete täielik läbisorimine (eeldades, et ühe võtme peale kulub n takti).*

3.4.1 Kirjeldus

Šifreerimine koosneb kaheksast raundist, millele järgneb väljundteisendus. Esimene raund ja väljundteisendus on kujutatud joonisel 3.5. Avatekst X jaotatakse neljaks 16-bitiseks alamplokiks X_1, X_2, X_3, X_4, st. $X = X_1X_2X_3X_4$. Analoogiliselt ka $Y = Y_1Y_2Y_3Y_4$. Sealjuures kasutatakse 52 erinevat võtme 16-bitist alamplokki, mis saadakse 128-bitisest võtmest hiljem defineeritud viisil (raundi r jooksul, $1 \leq r \leq 8$, kasutatavaid võtme alamplokke tähistame siinjuures $Z_1^r, \cdots, Z_6^r$. Väljundteisenduse juures kasutatud nelja alamplokki tähistame $Z_1^9, \cdots, Z_4^9$).

Defineerime järjestuse võtme alamplokkidel: $Z_1^1, \ldots, Z_6^1, Z_1^2, \ldots, Z_6^2, \ldots, Z_1^8, \cdots, Z_6^8, Z_1^9, \ldots, Z_4^9$. 128-bitine võti K jaotatakse kaheksaks alamplokiks, mida kasutatakse otseselt kui esimest kaheksat võtme alamplokki. Võtit K nihutatakse seejärel 25 biti võrra tsükliliselt vasakule, saadav 128-bitine võti jaotatakse jälle kaheksaks alamplokiks, millest saadakse järgmised 8 võtme alamplokki. Seejärel nihutatakse saadud võtit jälle tsükliliselt 25 biti võrra vasakule ja korratakse sama protseduuri, kuni on kätte saadud kõik 52 võtme alamplokki.

Dešifreerimisel kasutatakse sama arvutusgraafi, mida šifreerimiselgi, ainsaks vaheks on kasutatavad võtmed — Z_j^i asemel kasutatakse võtme alamplokki K_j^i, kus

$$
\begin{aligned}
(K_1^r, K_2^r, K_3^r, K_4^r) &= \left(\left(Z_1^{10-r}\right)^{-1}, -Z_3^{10-r}, -Z_2^{10-r}, (Z_4^r)^{-1}\right) & r = 2, \ldots, 8 \\
(K_1^r, K_2^r, K_3^r, K_4^r) &= \left(\left(Z_1^{10-r}\right)^{-1}, -Z_2^{10-r}, -Z_3^{10-r}, (Z_4^r)^{-1}\right) & r = 1, 9 \\
(K_5^r, K_6^r) &= (Z_5^r, Z_6^r) & r = 1, \ldots, 8
\end{aligned}
$$

Šifreerimisprotsessis kasutatakse kolme erinevat rühmatehet 16-bitistel alamplokkide paaridel, nimelt: