UNIVERZA V LJUBLJANI
FAKULTETA ZA ELEKTROTEHNIKO

Tomaž Požrl

Bločni šifrirni postopki

Seminarska naloga na podiplomskem študiju

Ljubljana, 2006

Kazalo

Kazalo slik

Kazalo tabel

1. Uvod

Živimo v svetu, kjer je človeška dejavnost neločljivo povezana z možnostjo pridobivanja in izmenjave najrazličnejših informacij. Pri tem je izrednega pomena, da je prava informacija ob pravem času na pravem mestu. Nekoč so ti postopki potekali »analogno«, z ročnim prenosom sporočil, poštnimi sistemi, dandanes, ko živimo v informacijski družbi, pa sporočila nastajajo in se nato tudi izmenjujejo praktično le še v elektronski obliki.

Napredek in modernizacija z uvedbo raznovrstnih računalniških sistemov je prinesel bistveno olajšanje vsakdanjega življenja, še posebej pa je to očitno prav pri izmenjavi informacij v elektronski obliki. Sodobni informacijski in telekomunikacijski sistemi omogočajo, da je vedno več elektronskih informacij enostavno javno dostopnih vedno večjemu številu potencialnih uporabnikov. Posamezniki lahko brez večjih težav dostopajo do skorajda poljubne informacije, ne da bi za to morali biti fizično v stiku z njo. Prav tako lahko tovrstne informacije tudi spreminjajo, brišejo, dodajajo, skratka z njimi načeloma počnejo, kar se jim zljubi.

Tu pa naletimo na veliko težavo elektronskih dokumentov. Tako kot jih je enostavno kreirati in jih po elektronski poti pošiljati k želeni osebi, jih je zelo lahko tudi prestreči med pošiljanjem in spreminjati njihovo vsebino. Telekomunikacijska infrastruktura namreč v osnovi dovoljuje možnost precejšnjih zlorab pri izmenjavi informacij in temu problemu je potrebno posvetiti veliko pozornosti. V današnjem času je tako zelo pomemben vidik pri načrtovanju telekomunikacijskih sistemov njihova varnost.

Vidik varnosti je mogoče razumeti na različne načine. V telekomunikacijah je osnovni element komunikacije sporočilo, ki ga pošljemu od pošiljatelja k prejemniku. Sporočila so lahko namenjena nekemu končnemu uporabniku in imajo zato sama po sebi določeno vrednost. Druge vrste sporočil pa so sistemska sporočila, ki omogočajo delovanje telekomunikacijskega sistema in pri katerih lahko vrednost ocenjujemo le posredno. Pri obeh vrstah sporočil pa lahko nepooblaščen dostop do sporočil in nepooblaščena uporaba sporočil uporabniku telekomunikacijskega omrežja povzroči zelo veliko škodo. Naloga varnega telekomunikacijskega sistema je preprečiti tovrstne zlorabe. Pojem varnosti v telekomunikacijah zato opredelimo kot sposobnost telekomunikacijskega sistema, da zagotavlja kar se le da zanesljiv prenos sporočil k njihovim naslovnikom, pri tem pa mora biti zagotovljena tudi celovitost sporočil.

Problem zagotavljanja varnosti v telekomunikacijskih sistemih pa ni nekaj novega, s čimer bi se soočili šele v današnjih dneh. Že vse od začetkov elektronskih komunikacij na sredini prejšnjega stoletja se inženirji srečujejo z vedno večjimi zahtevami pri zagotavljanju varnih prenosov elektronskih sporočil. Podobne koncepte varnosti pri prenosih sporočil pa bi lahko sledili še mnogo bolj nazaj v človeško zgodovino. S podobnimi problemi so se prvi ukvarjali že Egipčani 4000 let pred našim štetjem, od tedaj pa so jih vse do današnjih dni srečevali na najrazličnejših področjih človeškega udejstvovanja, od začetka predvsem v vojski, diplomatskih službah in pri vodenju držav.

Z iznajdbo računalnikov in komunikacijski sistemov v 60. letih prejšnjega stoletja je problem varnosti postal precej bolj izrazit tudi v zasebnem sektorju v smislu varovanja elektronskih informacij in zagotavljanja varnih komunikacijskih sistemov. Prvi produkti raziskav na področju kriptografije, vede, ki se ukvarja z iskanjem

postopkov za zagotavljanje varnosti pri izmenjavi informacij, so se pričeli uveljavljati že na začetku 70. let prejšnjega stoletja, ko je ameriški zvezni urad začel z uporabo danes zelo znanega standarda DES. Vse odtlej se pojavljajo vedno nove in nove zamisli, kako komunikacije narediti varne, s tem pa se tudi v vsakdanjem življenju uveljavljajo praktične implementacije vedno bolj zanesljivih varnostnih sistemov.

2. Kriptografija

Kriptografija je študija matematičnih tehnik, ki so povezane z različni aspekti informacijske varnosti, kot npr. zasebnost, celovitost podatkov, avtentikacija entitet v komunikaciji in avtentikacija izvora podatkov.

Skozi stoletja so znanstveniki razvili celo vrsto različnih protokolov in mehanizmov, ki se ukvarjajo s specifičnimi problemi informacijske varnosti, ko govorimo o fizičnih dokumentih. Pogosto takšni problemi niso rešljivi le z matematičnimi postopki, temveč je potrebno določene elemente opredeliti tudi v zakonodaji. Tudi v današnjih časih, ko imamo že v veliki meri opravka z elektronskimi dokumenti, se stvari ne lotevamo bistveno drugače. Ena večjih razlik pa je vseeno ta, da je informacije dandanes veliko enostavneje kopirati in spreminjati. In ne samo to – naredimo lahko na tisoče enakih kopij, ki jih ni mogoče ločiti od originala. Zato je eden najpomembnejših zahtev pri uveljavljanju informacijske varnosti, zmožnost podpisovanja informacij oz. dokumentov. Podpis je osnovni element za številne aspekte informacijske varnosti.

Ne glede na to v kakšni obliki so informacije, je potrebno zagotoviti njihovo varnost. V tej seminarski nalogi bomo obravnavali varnost v današnji informacijski družbi, v tem kontekstu pa je potrebno v računalniški obliki implementirati različne matematične tehnike in mehanizme, ki poskrbijo za različne vidike informacijske varnosti.

Z v prejšnjih dveh odstavkih omenjenimi problemi se torej ukvarja kriptografija. Ta sicer ni edina možnost, kako zagotoviti varnost v komunikacijah, je pa na tem področju v zadnjem času izjemno veliko raziskav, ki ponujajo vedno nove in nove rešitve. Kakšni so skupni cilji vseh teh raziskav pa v naslednjem poglavju.

2.1 Cilji kriptografije

Kriptografija si pri zagotavljanju varnosti v informacijski družbi postavlja različne naloge in cilje. Med temi cilji bi lahko izpostavili štiri, ki so najbolj bistveni in iz katerih lahko izpeljemo vse ostale. Ti cilji so:

- *Zasebnost* – ta vidik zagotavlja, da je vsebina sporočila dostopna samo njegovemu naslovniku. Uporablja se tudi sinonimen pojem tajnost. Obstajajo številni pristopi k zagotavljanju zasebnosti, od fizične zaščite pa vse do matematičnih algoritmov, ki informacije preoblikujejo v potencialnemu napadalcu nerazumljivo obliko.
- *Celovitost* podatkov – ta vidik preprečuje nepooblaščeno spreminjanje podatkov. Da bi ta vidik lahko zagotoviti, moramo biti sposobni odkriti

manipulacijo s podatki. Pod manipulacijo s podatki pa razumemo tako dodajanje podatkov, brisanje podatkov in zamenjavo podatkov (simbolov, besed, besednih zvez …)

- *Avtentičnost* – ta vidik je povezan z identifikacijo. Slednja se nanaša tako na samo informacijo kot tudi na vpletene entitete (pošiljatelja, naslovnika …). Pri komunikaciji se morata obe »stranki« predhodno identificirati. Pri prenosu informacije preko komunikacijskega kanala mora biti tudi ta identificirana – v smislu izvora informacije, datuma nastanka informacije, vsebine informacije, časa ko je bila informacija poslana itd. Zaradi omenjenih razlogov je avtentičnost običajno razdeljena v dva velika podproblema: *avtentičnost entitet* in *avtentičnost izvora podatkov*. Slednji implicitno vsebuje tudi problem celovitosti, saj se je v primeru spremembe sporočila moral spremeniti tudi izvor.
- *Nezanikanje* – ta vidik preprečuje, da bi določena entiteta zanikala prej izvedeno akcijo. V primeru da npr. pošiljatelj zanika, da je poslal sporočilo, morajo obstajati mehanizmi, ki ta zaplet razrešijo. Običajno se v takšnih primerih v komunikacijo vključi zaupanja vredno tretjo osebo (oz. institucijo).

2.2 Kriptografski pristopi

Osnovni cilj kriptografije je ustrezno doseganje v prejšnjem poglavju naštetih ciljev tako v teoriji kot v praksi. Kriptografija se ukvarja z detekcijo in preprečevanjem zlorab v komunikaciji.

Za doseganje vseh teh nalog se v kriptografiji uporablja zelo raznovrstne pristope, orodja oz. primitive, ki so predstavljeni v tabeli 1. V grobem bi lahko te pristope razdelili v **pristope z uporabo ključev** in tiste, ki za zagotavljanje varnosti **ne uporabljajo ključev**. Prve bi lahko razdelili še nekoliko podrobneje v postopke s **simetričnim ključem** in postopke z **javnim ključem**.

<table>
<tr><th>Postopki s ključem</th><th>Simetrični ključ</th><th>Šifrirni postopki simetričnim ključem</th><th>Bločni šifrirni postopki</th></tr>
<tr><td rowspan="6"></td><td rowspan="4"></td><td rowspan="2"></td><td>Pretočni šifrirni postopki</td></tr>
<tr><td colspan="2">Podpisi</td></tr>
<tr><td colspan="2">Identifikacija</td></tr>
<tr><td colspan="2">Pseudo naključne sekvence</td></tr>
<tr><td colspan="2">Zgoščevalne funkcije</td></tr>
<tr><td rowspan="3">Javni ključ</td><td colspan="2">Podpis</td></tr>
<tr><td colspan="2">Identifikacija</td></tr>
<tr><td colspan="2">Šifrirni postopki z javnih ključem</td></tr>
</table>

Postopki brez ključa	Naključne sekvence
	Enosmerne permutacije
	Zgoščevalne funkcije

Tabela 1 : Kriptografski primitivi

Primitive, ki ne uporabljajo ključev, za nas v okviru te seminarske naloge ne bodo posebej zanimivi, vseeno pa omenimo, da jih lahko razdelimo v naključne sekvence, enosmerne permutacije in zgoščevalne funkcije poljubnih dolžin.

Postopki z javnim ključem temeljijo, kot že ime pove, na javnem ključu, razviti pa so bili na primer v obliki podpisov, identifikacijskih primitivov in šifrirnih postopkov z javnim ključem.

Za nas najbolj zanimivi pa so postopki s simetričnim ključem. Slednji prav tako omogočajo na primer implementacijo podpisov in identifikacije, ob tem pa je možno z njimi izvesti tudi pseudo naključne sekvence, zgoščevalne funkcije poljubnih dolžin in, kar bo nas v tej seminarski nalogi še najbolj zanimalo, šifrirne postopke s simetričnim ključem. Med slednje uvrščamo tako pretočne šifrirne postopke kot tudi bločne šifrirne postopke, ki jim bom posvetili največ pozornosti.

Vsi omenjeni pristopi morajo za uspešno implementacijo v praksi zadostiti določenim pogojem. Najpomembnejši pogoj je prav gotovo *varnost postopka*. Ta se običajno meri s številom operacij, ki so potrebne za razbitje algoritma in tako predstavlja ovrednotenje truda, ki ga je potrebno vložiti v razbijanje enkripcijskega postopka. *Funkcionalnost postopka* je pomembna, ko govorimo o namembnosti postopka, torej kateremu varnostnemu vidiku je namenjen. Vsak postopek lahko uporabimo na *različne načine*, temu ustrezno pa se lahko do določene mere spreminja tudi njegova funkcionalnost. Posledično se lahko spreminja tudi *učinkovitost* oz. *zmogljivost postopka* v določenem načinu uporabe. Konec koncev pa je iz praktičnih razlogov pomembno tudi to, kako *enostavno* je določen postopek možno *realizirati* v konkretni situaciji, pri čemer mislimo tako na programsko kot strojno implementacijo.

2.3 Osnovni pojmi in koncepti

Preden se posvetimo bločnim šifrirnim postopkom, se na kratko dotaknimo tudi osnovne terminologije in konceptov, ki so ključnega pomena za dobro razumevanje opisanih postopkov.

Osnovni element kriptografije je dejansko **sporočilo**, ki ga želimo prenesti po komunikacijskem kanalu in ga zato predelujemo.

Sporočilo je sestavljeno iz več **simbolov**, ki pripadajo t.i. **abecedi**. Ta je lahko binarna (0,1), sestavljena iz črk slovenske abecede, lahko pa vsebuje tudi katerikoli drug simbol, ki predstavlja zalogo vrednosti, ki jo simbol lahko zavzame. S simboli iz

abecede nato sestavimo sporočila, ki pripadajo t.i. **sporočilnemu prostoru**. Iz simbolov lahko ustvarimo določeno število sporočil, ki so spet lahko predstavljena v različnih oblikah: binarni, kot navaden tekst, računalniška koda in podobno.

Poleg sporočilnega prostora, v katerem so **izvorna sporočila**, lahko definiramo tudi **prostor šifriranih sporočil**. Slednji je sestavljen iz sporočil, ki jih prav tako sestavljajo posamezni simboli, ki pa imajo lahko drugačno abecedo kot navadna sporočila.

V konceptu šifriranja s **ključi** je potrebno definirati tudi **prostor ključev**. Slednjega sestavljeni posamezni ključi, ki se uporabljajo pri šifriranju sporočil.

Vsak šifrirni ključ iz prostora ključev določa edinstveno bijektivno preslikavo iz prostora navadnih sporočil v prostor šifriranih sporočil. Ta preslikava je t.i. **enkripcijska funkcija** oz. **enkripcijska transformacija**. Potrebno je posebej poudariti, da mora biti omenjena funkcija bijektivna, če želimo iz vsakega šifriranega sporočila dobiti nazaj originalno unikatno sporočilo.

Na podoben način vsak dešifrirni ključ iz prostora ključev določa bijekcija iz prostora šifriranih sporočil v prostor navadnih sporočil. Takšno preslikavo imenujemo **dekripcijska funkcija** oz. **dekripcijska transformacija**.

Proces uvajanja omenjenih dveh vrst preslikav običajno na kratko poimenujemo **enkripcija** in **dekripcija**.

Enkripcijska shema je sestavljena iz enkripcijskih in dekripcijskih transformacij, vključno s pripadajočimi šifrirnimi in dešifrirnimi ključi, ki imajo to lastnost, da dešifrirni ključ pomaga pri dekripcijski transformaciji in sicer tako, da poprej enkriptirano sporočilo s pomočjo šifrirnega ključa povrne nazaj v izvorno sporočilo. Takšnima ključema običajno pravimo kar **par ključev**, celotni enkripcijski shemi pa na kratko kar **šifra**.

Enkripcijsko shemo lahko označimo kot zlomljivo, če lahko tretja oseba brez poznavanja para ključev razvozla originalno sporočilo iz šifriranega in to v nekem primernem časovnem okviru.

S matematičnimi postopki za razbijanje kriptografskih se sicer ukvarja posebna veda, ki ji pravimo **kriptoanaliza**. Slednja išče napake v enkripcijskih postopkih in na ta način prav tako prispeva k informacijski varnosti. Obe vedi skupaj, torej kriptografijo in kriptoanalizo, običajno poimenujemo **kriptologija**.

3. Bločni šifrirni postopki

3.1 Enkripcija s simetričnim ključem

Enkripcija s simetričnim ključem je poseben primer enkripcijske sheme, za katero velja, da je za vsak par enkripcijsko/dekripcijskih ključev računsko enostavno določiti dekripcijski ključ, če poznamo le enkripcijskega oz. določiti enkripcijski ključ, če poznamo le dekripcijskega.

V praktičnih enkripcijskih shemah s simetričnim ključem sta enkripcijski in dekripcijski ključ običajno enaka, zato se za tovrstno shemo običajno uporablja kar ime enkripcija

s simetričnim ključem. V uporabi je tudi nekaj drugi izrazov, npr. enkripcija z enim ključem, privatnim ključem in konvencionalna enkripcija.

Poglejmo si delovanje enkripcije s simetričnim ključem na primeru.

Primer: Naj bo abeceda enkripcije kar slovenska abeceda. Prostor izvornih sporočil in prostor šifriranih sporočil naj predstavljata znakovne nize dolžine 5 simbolov iz slovenske abecede. Za enkripcijski ključ si izberemo permutacijo nad abecedo. Postopek enkripcije nato poteka tako, da najprej razbijemo sporočilo v skupine simbolov oz. bloke, kjer je vsaka skupina dolga natanko 5 simbolov. Če dolžina sporočila ni večkratnik števila 5, manjkajoča mesta zapolnimo z izbranim simbolom. V drugi fazi nato nad vsakim znakom iz te skupine izvedemo permutacijo. V fazi dekripcije se podobno izvede inverzna permutacija nad vsakim simbolom v blokih šifriranega sporočila. Za še boljšo predstavo si izberimo, da permutacija pomeni preslikavo simbola za 3 mesta desno v abecedi.

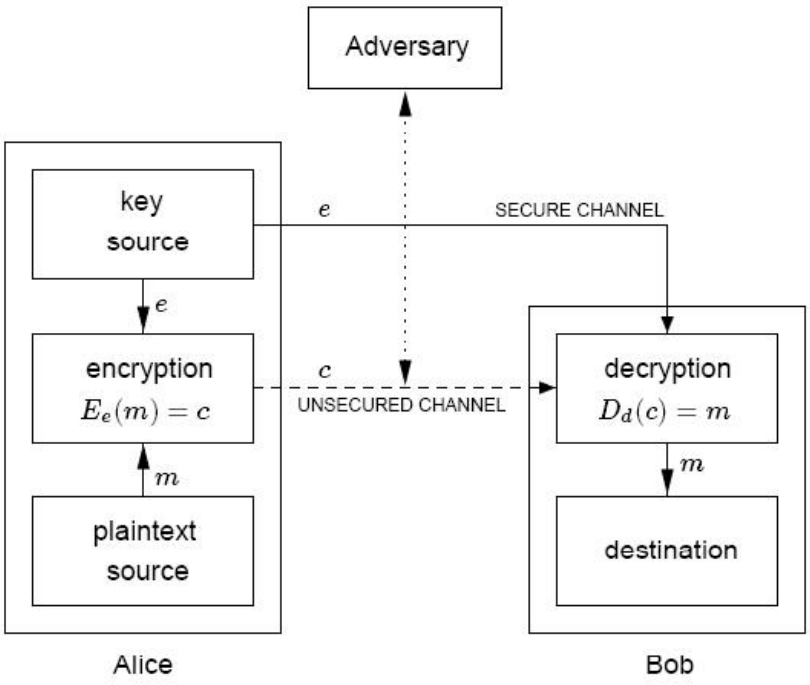
Sporočilo:

TAENK RIPCI JANIV ARNA

bi se enkriptiralo v

ZČHRN TLŠEL MČRLA ČTRČ .

Komunikacijo z enkripcijo s simetričnim ključem nazorno prikazuje bločni diagram na sliki 1. Na sliki je že dodan tudi varen prenosni kanal, ki je namenjen zagotavljanju zasebnosti in avtentičnosti.



Slika 1 : Komunikacija z enkripcijo s simetričnim ključem

Eden izmed najbolj izrazitih problemov v komunikaciji s simetričnim ključem je iskanje učinkovite metode za dogovor o uporabi konkretnega ključa in varno izmenjavo tega ključa med entitetami v komunikaciji. Ta problem običajno imenujemo *problem distribucije ključa*.

V komunikaciji z enkripcijo s simetričnim ključem se predpostavlja, da vse entitete v komunikaciji poznajo enkripcijsko/dekripcijske transformacije (oz. enkripcijsko shemo). Številne raziskave so pokazale, da je edina informacija, ki jo je potrebno skriti, dekripcijski ključ. To pa seveda pomeni, da mora biti tudi enkripcijski ključ tajen, saj je po osnovnih konceptih enkripcije s simetričnim ključem enkripcijski ključ mogoče določiti iz dekripcijskega. Zato je na zgornji sliki prikazan prenos šifriranega sporočila po nezavarovanem kanalu, medtem ko enkripcijski ključ, ki je bil uporabljen za šifriranje izvornega sporočila, prenesemo po varnem kanalu.

V kriptografskih raziskavah se enkripcijske sheme s simetričnim ključem običajno loči v **bločne šifrirne postopke** in **pretočne šifrirne postopke**. Prvi so glavna tema te seminarske naloge.

Bločni šifrirni postopek je enkripcijska shema, ki izvorno sporočilo, ki ga želimo prenašati, razdeli v nize (imenovane bloki) stalne dolžine, ki so sestavljeni iz simbolov iz abecede. Bločni šifrirni postopek enkriptira po en blok naenkrat.

V praksi je večina shem s simetričnim ključem prav bločnih šifrirnih postopkov. Med slednjimi velja omeniti dva pomembna podrazreda – bločne šifrirne postopke z zamenjavo (**substitucijski bločni šifrirni postopki**) in bločne šifrirne postopke s premeščanjem (**transpozicijski bločni šifrirni postopki**).

3.2 Uvod v bločne šifrirne postopke

Bločni šifrirni postopki s simetričnim ključem so dandanes izjemno pomemben sestavni del mnogih kriptografskih sistemov. Sami po sebi zagotavljajo zasebnost, pojavljajo pa se še v številnih sistemih za zagotavljanje varnostnih vidikov pri prenosu (elektronskih) sporočil. Zaradi prilagodljivosti tako omogočajo konstrukcijo pseudonaključnih generatorjev števil, pretočnih šifrirnih postopkov, kod za avtentikacijo sporočil in zgoščevalnih funkcij. Prav tako pa se bločni šifrirni postopki uporabljajo kot osnovna komponenta v tehnikah za avtentikacijo sporočil, mehanizmih za zagotavljanje celovitosti podatkov, protokolih avtentikacijo entitet v komunikaciji ter sheham za digitalne podpise.

Zaradi splošne razširjenosti bločnih šifrirnih postopkov je potrebno poudariti, da nobeden izmed postopkov ni primeren za vsako aplikacijo, čeprav za določeno nalogo nudi zelo visoko stopnjo varnosti. Vzrok je v prilagoditvah za specifično praktično aplikacijo, ki izhajajo iz potreb po hitrosti, pomnilniških zahtev, strojnih in programskih omejitev platform, ter tolerancah pri specifičnih načini uporabe šifrirnih postopkov. Vse te omejitve narekujejo razvoj različnih tipov bločnih šifrirnih postopkov, ki jih nato izberemo in uporabimo po potrebi glede na specifično varnostno nalogo. V tej seminarski nalogi se bomo omejili na nekatere najbolj znane in najbolj pogosto uporabljane bločne šifrirne postopke.

3.3 Bločni šifrirni postopki – splošno

Bločni šifrirni postopki so lahko izvedeni s *simetričnim ključem* ali pa z *arhitekturo javnega ključa*. V tej seminarski nalogi se bodo posvetili prvemu tipu enkripcijske sheme.

Definicija: Bločni šifrirni postopek je funkcija, ki mapira n bitne bloke izvornega sporočila v n bitne bloke šifriranega sporočila, kjer je n dolžina bloka. Kot primer takšne funkcije si lahko predstavljamo preprosto šifriranje z zamenjavo z velikim naborom znakov. Funkcija ima kot parameter k bitni ključ K. V splošnem predpostavljamo, da se ključ v takšni funkciji izbere naključno. Z uporabo enake dolžine izvornega in šifriranega sporočila preprečimo povečanje podatkov.

Zato da je omogočena unikatna dekripcija, mora biti enkripcijska funkcija tipa 1-1 (torej takšna, da se lahko obrne). Za n bitno izvorno in šifrirano sporočilo in nespremenljiv ključ je enkripcijska funkcija bijekcija, ki določa permutacijo n bitnega vektorja. Vsak ključ v principu določa drugačno bijekcijo. Število ključev je |K|, efektivna velikost ključa pa je log |K|.

Definicija: N bitni bločni šifrirni postopek je funkcija E definirana kot E : V_n x K → V_n in sicer tako, da je za vsak ključ K iz prostora ključev E(P,K) preslikava, ki se jo lahko obrne (enkripcijska funkcija za K) iz V_n v V_n, zapišemo pa jo kot $E_K(P)$. Inverzna preslikava je dekripcijska funkcija, označena kot $D_K(C)$. Formula C = $E_K(P)$ pomeni, da smo šifrirano sporočilo C dobili iz izvornega sporočila P z uporabo ključa K.

Najbolj splošen bločni šifrirni postopek ima vključene vse možne zamenjave. Za zapis ključa takšnega pravega naključnega bločnega šifrirnega postopka bi potrebovali približno 2^n krat toliko bitov kot je v bloku sporočila (kjer je n število bitov v bloku), to pa je popolnoma nepraktično. Ne glede na to pa morajo realni bločni šifrirni postopki vsaj navidezno posnemati idejo naključnosti.

Definicija: Pravi naključni bločni šifrirni postopek je n bitni bločni šifrirni postopek, ki vključuje vseh 2^n! bijektivnih preslikav na 2^n elementih. Vsak od 2^n! ključev določa eno takšno permutacijo.

Če je velikost bloka n premajhna, je šifrirni postopek lahko preveč ranljiv na napade, ki temeljijo na statistični analizi. Tako na primer lahko s preprosto frekvečno analizo blokov šifriranega sporočila razbijemo postopek. Po drugi strani pa preveliki bloki lahko precej otežijo implementacijo šifrirnih postopkov, saj težavnost le-te drastično narašča z velikostjo bloka. V praksi so zato potrebne preprostejše funkcije, ki pa so zato le navidezno naključne.

Kot smo nakazali že v poglavju o postopkih s simetričnim ključem, lahko bločne šifrirne postopke v grobem razdelimo v substitucijske in transpozicijske. Za prvo skupino šifrirnih postopkov je značilno, da šifriranje izvajajo s pomočjo zamenjave simbolov v izvornem sporočilu z nekimi drugimi simboli oz. skupino simbolov iz enkripcijske abecede. To lahko poteka na zelo preprost način, tako da vsak simbol v sporočilu posebej zamenjamo z drugim simbolom iz abecede z nekim permutacijskim ključem. Takšen bločni šifrirni postopek imenujemo prepost šifrirni postopek z zamenjavo oz. *mono-abecedni postopek z zamenjavo*. Tak postopek ne zagotavlja visoke stopnje varnosti, saj ga je z opazovanjem porazdelitve simbolov v šifriranem sporočilu dokaj enostavno razbiti.

Nekoliko bolj varen je *homofonični bločni postopek z zamenjavo*. Za ta postopek je značilno, da je za vsak simbol iz abecede izvornega sporočila definiran nabor nizov simbolov, v katere se lahko ta simbol preslika. Dolžina teh nizov ni predpisana. Enkripcija nato poteka tako, da vsak simbol iz izvornega sporočila preslikamo v naključno izbran niz iz nabora temu simbolu pripadajočih nizov. S takšnim šifrirnim postopkom naredimo distribucijo šifriranih simbolov precej bolj uniformno, a na račun večje količine prenašanih podatkov. Po drugi strani pa tudi dekripcijo ni tako enostavno izvesti kot pri prej omenjenem postopku.

Naslednja skupina bločnih postopkov z zamenjavo so *poli-abecedni postopki*. Za njih je značilnen prostor ključev, ki je sestavljen iz razvrščenega niza permutacij, ki so vse definirane nad abecedo izvornega sporočila. Enkripcija nato poteka tako, da za vsak blok simbolov uporabimo različno permutacijo oz. različen ključ. Analogno poteka tudi dekripcija. Tovrstni postopki so primerni predvsem zato, ker ne ohranjajo frekvenco simbolov v sporočilu. Vendar pa na tak način šifriranih sporočil vseeno ni bistveno težje razbiti kot pri prej omenjenih postopkih, saj se z analizo frekvence simbolov v vsakem posameznem bloku lahko brez večjih težav prebijemo tudi do izvorne vrednosti teh blokov.

Druga večja skupina bločnih šifrirnih postopkov so postopki s premeščanjem. Najbolj enostavna različica tovrstnih postopkov preprosto premeša simbole znotraj bloka. V splošnem postopki s premeščanjem predstavljajo nabor vseh možnih transformacij, ki premešajo vrstni red simbolov v bloku. Enkripcija poteka tako, da izberemo eno izmed takšnih transformacij, ki ustrezno premeša simbole. Takšen preprost primer postopka s premeščanjem je očitno zelo slabo zavarovan, saj se ohranja število simbolov v bloku, kar predstavlja zelo dobro izhodišče za vse potencialne napadalce in razbijalce enkripcijske sheme.

Ker posamezni bločni šifrirni postopki z zamenjavo in premeščanjem ne zagotavljajo dovolj visoke stopnje varnosti, se v praksi običajno ne uporabljajo samostojno, temveč se jih sestavi v t.i. **produktne bločne šifrirne postopke**. Slednje zgradimo s kompozicijo posameznih bločnih šifrirnih postopkov, kar v praksi predstavlja matematično operacijo kompozicije funkcij. Za slednjo velja, da je definirana tako, da zapovrstjo izjavamo funkcije nad rezultati prejšnje funkcije. Seveda mora veljati, da se definicijsko območje druge funkcije ujema z zalogo vrednosti prve funkcije. V splošnem takšna kompozicija lahko zajema poljubno število funkcij oz. v našem primeru poljubno število enostavnih bločnih šifrirnih postopkov.

Poglejmo si primer takšnega produktnega bločnega šifrirnega postopka. Naj bodo prostori izvornih in šifriranih sporočil vsi znakovni nizi dolžine šest. Prvi šifrirni postopek naj bo funkcija eksluzivno ali (XOR – binarna funkcija z dvema vhodnima spremenljivkama, ki ima vrednost 1, če ima natanko ena od vhodnih spremenljivk vrednost 1, sicer pa ima funkcija vrednost 0). Ta funkcija je poli-abecedni šifrirni postopek z zamenjavo. Šifrirni postopek s premeščanjem pa naj bo preprosto premešanje vrstnega reda simbolov, kot jih dobimo iz prvega postopka. Takšna skupna shema predstavlja produktni bločni šifrirni postopek, o katerih bomo več besed spregovorili v naslednjih poglavjih.

Zelo pomemben del postopkov s simetričnim ključem so pari enkripcijsko/dekripcijskih ključev. Prostor ključev sestavljajo vsi pari, ki jih lahko v

določenem postopku uporabimo, npr. vse možne permutacije znakov v sporočilu. Običajno se varnost enkripcijske sheme ovrednoti ravno z enkripcijsko/dekripcijskimi ključi oz. številom vseh razpoložljivih ključev. Velja tudi, da je potreben, a ne vedno zadosten, pogoj za to, da je enkripcijska shema varna, da je prostor ključev dovolj velik, da je postopek varen tudi pri izčrpnem iskanju ustreznega para ključev.

Ker gre pri šifrirnih postopkih za zagotavaljanje določenega vidika varnosti elektonskih sporočil, se je potrebno dotakniti tudi uspešnosti oz. učinkovitosti bločnih šifrirnih postopkov. Kot smo že omenili, je osnovna funkcija postopkov s simetričnih ključem zagotavljanje zasebnosti. Potencialni napadalec bi torej želel iz šifriranega sporočila želel pridobiti izvorno sporočilo. Pri tem obstaja več možnosti, s kakšno strategijo bi se napadalec lahko lotil takšnega napada. Ob tem je pri evaluaciji postopkov potrebno vedno predpostaviti, da ima napadalec lahko tako dostop do vseh podatkov, ki se prenašajo preko kanala s šifriranim sporočilom kot tudi, da pozna vse podrobnosti enkripcijske funkcije z izjemo skritega ključa, ki je torej glavni nosilec varnosti celotnega postopka. Glede na omenjene predpostavke torej ločimo več razredov kriptoanalitičnih napadov:

- Samo šifrirano sporočilo – drugi podatki napadalcu niso znani
- Poznano izvorno sporočilo – napadalec pozna pare izvornih in šifriranih sporočil
- Poznana določena izvorna sporočila – za določena izvorna sporočila po napadalčevi želji so znana tudi šifrirana sporočila

Pri evaluaciji bločnih šifrirnih postopkov lahko uporabimo številne kriterije, npr.:

- Ocenjen varnostni nivo – zaupanje v varnost postopkov narašča, če so bili že podvrženi kriptoanalitičnim napadom in so jih uspešno prestali v določenem časovnem obdobju npr. preko več let.
- Velikost ključa – efektivna bitna dolžina ključa oz., bolj natančno, entropija prostora ključev določa zgornjo mejo varnosti šifrirnega postopka (z upoštevanjem izčrpnega iskanja). Daljši ključi pa običajno predstavljajo dodatne stroške (generacija ključa, distribucija, shranjevanje, težave pri pomnenju gesel …)
- Proizvodnja – slednje se nanaša na kompleksnost kriptografske preslikave in stopnjo do katere je preslikava prilagojena za določeno platformo oz. medij
- Velikost bloka – ta vpliva tako na varnost (zaželjena je večja velikost) in kompleksnot (večji bloki so dražji za implementacijo). Velikost lahko precej vpliva tudi na delovanje samega postopka, npr. ko je potrebno polnjenje bloka, ker število bitov ni večkratnik števila bitov v bloku
- Kompleksnost kriptografske preslikave – algoritemska kompleksnost vpliva na stroške tako v smislu razvoja in fiksnih stroškov (strojnih in programskih) kot tudi stroškov delovanja v realnem času. Določeni postopki tako posebej zahtevano strojno ali programsko implementacijo

- Povečanje podatkov – v večini primerov je zaželjeno ali celo obvezno, da enkripcija ne poveča velikosti izvornih sporočil. Določeni bločni postopki (homofonična substitucija in naključna enkripcija) pa velikost sporočil vsekakor povečajo.
- Razširjanje napak – dekripcija šifriranih sporočil, ki vsebujejo napake, lahko povzroči različne napake v izvornih sporočilih, med drugim razširjanje napak v naslednjih blokih. Določene karakteristike napak so sicer v določenih aplikacijah povsem sprejemljive, na razširjanje napak pa običajno vpliva tudi velikost bloka.

3.4 Načini delovanja bločnih šifrirnih postopkov

Bločni šifrirni postopki lahko delujejo v različnih načinih. V osnovi bločni postopki šifrirajo izvorna poročila v fiksnih n bitnih blokih. Za sporočila, ki presegajo n bitov, pa se lahko poslužimo različnih pristopov. Najenostavnejši je ta, da sporočilo razbijemo v n bitne bloke in šifriramo vsakega posebej. Ta pristop se imenuje ECB (electronic-codebook) in ima več slabih lastnosti, ki v večini aplikacij favorizirajo druge pristope.

Najbolj poznani načini uporabe bločnih šifrirnih postopkov so sicer poleg ECB še CBC, CFB in OFB. Poglejmo si na kratko njihove glavne značilnosti.

- ECB – electronic-codebook:
    - Vhodni podatki so n bitno bloki izvornega sporočila in k bitni ključ, izhodni pa bloki šifriranega sporočila
    - Identična izvorna sporočila se z istim ključem vedno zašifrirajo v identično šifrirano sporočilo
    - Verižne odvisnosti: bloki se šifrirajo povsem neodvisno od drugi blokov. Sprememba vrstnega reda blokov šifriranega sporočila zato povzroči tudi spremembo vrstnega reda izvornega sporočila
    - Razširjanje napak: ena ali več napak v enem bloku šifriranega sporočila vpliva samo na dekripcijo tega specifičnega bloka. Za običajne postopke je dekripcija takšnega bloka potem naključna, 50% pridobljenega izvornega sporočila pa ima napake.
    - Tovrstni bločni postopki ne skrijejo podatkovnih vzorcev, zato uporaba ECBja ni priporočljiva, če so sporočila daljša od enega bloka ali če določen ključ uporabljamo na več kot enem bloku.

- CBC – cipher-block chaining
    - Vhodni podatki: k bitni ključ, n bitni inicializacijski vektor, n bitni bloki izvornega sporočila
    - Izhodni podatki: bloki šifriranega sporočila.

- Enkripcija poteka tako, da se uporabi XOR funkcija nad trenutnim blokom izvornega sporočila in prejšnjim šifriranim blokom. Za inicializacijo seveda potrebujemo ničelni blok šifriranega sporočila, ki ga dobimo s pomočjo inicializacijskega vektorja. Podobno poteka tudi obraten postopek pri dekripciji.
- Identično šifrirano sporočilo dobimo iz identičnega izvornega sporočila le, če se ne spremeni ključ, inicializacijski vektor ali prvi blok izvornega sporočila.
- Verižne odvisnosti: zaradi verižnega mehanizma je določeni šifrirani blok odvisen tako od pripadajoče izvornega bloka kot tudi od vseh predhodnih blokov izvornega sporočila (vsa odvisnost od prehodnih blokov se sicer pokaže v vrednosti prvega predhodnega bloka). Spreminjanje vrstnega reda blokov šifriranega sporočila tako vpliva na dekripcijo.
- Razširjanje napak: napaka v enem samem bitu v bloku šifriranega sporočila vpliva na dekripcijo pripadajočega bloka kot tudi naslednjega bloka šifriranega sporočila. Dešifrirani blok iz takšnega bloka z napako je običajno povsem naključen (50% napaka), naslednji dešifriran blok pa vsebuje bitno napako na natanko istem mestu kot jo je vseboval blok v šifriranem sporočilu. Potencialni napadalec zato lahko povzroči predvidljive spremembe v dešifriranem bloku s spreminjanjem pripadajočega bloka v šifriranem bloku.
- Reševanje napak: CBC način delovanja bločnih šifrirnih postopkov so samo-sinhrinizirajoči v tem smislu, da če pride do napake v določenem bloku šifriranega sporočila, ne pa tudi v njemu naslednjem bloku, bo drugi naslednji blok pravilno dešifriranem. Reševanje iz napak pa ni možno, če do napak pride na mejnih bitih v bloku, saj se s tem podre struktura blokov v sporočilu.
- Pomemben člen v CBC načinu je tudi inicializacijski vektor, ki sicer ni nujno skrit, potrebno pa je zagotoviti njegovo celovitost, saj bi škodoželjno spreminjanje tega vektorja omogočilo napadalcu, da izvede predvidljive spremembe v prvem bloku dešifriranega sporočila.

- CFB – cipher feedback
  - Medtem ko CBC način procesira n bitov izvornega sporočila naenkrat, pa nekatere aplikacije zahtevajo, da se brez zakasnitev šifrira in prenese npr. r bitov izvornega teksta, kjer je $r < n$ (običajno je $r = 1$ ali $r = 8$). V tem primeru se uporabi CFB način.
  - Vhodni podatki: k bitni ključ, n bitni inicializacijski vektor, r bitni bloki izvornega sporočila
  - Izhodni podatki: r bitni bloki šifriranega sporočila
  - Enkripcija poteka tako, da iz inicializacijskega vektorja določimo vrednost I_1, ki služi kot začetna vhodna vrednost pomika v pomikalnem registru. Ta vrednost se nato enkriptira v O_j, ki je vrednost izhoda bločnega šifrirnega postopka. Iz te vrednosti se vzame r najbolj levih

bitov in uporabi kot ključ pri izračunu (podobno kot pri CBC z XOR funkcijo) šifrirane vrednosti bloka izvornega sporočila. Dobljeno šifrirano vrednost se premakne na desni konec premikalnega registra, to pa se uporabi kot vrednost spremenljivke I_j v naslednjem koraku enkripcije. Dekripcije poteka zelo podobno, le da v obratnem vrstnem redu.

- Podobno kot pri CBC načinu se tudi ne sme spreminjati inicializacijski vektor, če želimo dobiti identično izvorno sporočilo iz identičnega šifriranega sporočila. Tudi tu ta vektor ni nujno skrit, čeprav nekatere aplikacije zahtevajo nenapovedljivo vrednost vektorja.
- Verižne odvisnosti: podobno kot enkripcija CBC verižni mehanizem enkripcije tudi pri CFB povzroča odvisnost bloka šifriranega sporočila tako od trenutnega bloka izvornega sporočila kot tudi tudi od prejšnjega bloka šifriranega sporočila. Pravilna dekripcija zahteva, da je predhodnih n/r blokov šifiranega sporočila pravilnih, zato da pomikalni register vsebuje pravilne vrednosti.
- Razširjanje napak: ena ali več napak v enem r bitnem bloku šifriranega sporočila vpliva na dešifriranje tega in še naslednjih n/r blokov (dokler ni sprocesiranih n bitov šifriranega sporočila, napaka ne bo izginila iz pomikalnega registra). Pridobljena vrednost originalnega sporočila se bo od točne razlikovala natanko v bitu, ki je bil napačen v šifriranem sporočilu, pri ostalih nepravilno dešifrirani bloki pa so običajno naključni vektorji (50% napak).
- Reševanje napak: CFB je podobno kot CBC samo-sinhrinizirajoč, vendar pa potrebuje n/r blokov, da se sinhronizira nazaj.
- Za CFB je značilno tudi to, da v primerjavi s CFB povzroči n/r krat manj računanja, saj vsaka enkripcija pomeni samo r bitov proizvedenega šifriranega teksta.

- OFB – output feedback
  - ta način uporabe bločnih šifrirnih postopkov se uporablja tedaj, ko si ne želimo nobenega razširjanja napak
  - deluje podobno kot CFB, omogoča enkripcijo različnih dolžin blokov (znakov), razlika pa je v tem, da namesto šifriranega sporočila za feedback služi izhod enkripcijske blokovne funkcije E
  - Vhodni podatki: k bitni ključ, n bitni inicializacijski vektor, r bitni bloki izvornega sporočila
  - Izhodni podatki: r bitni bloki šifriranega sporočila
  - Enkripcija poteka zelo podobno kot pri metodi CFB, le da se v zadnjem koraku za nov I_j za naslednjo fazo enkripcije vzame vrednost O_j iz prvega koraka te faze enkripcije.
  - Podobno kot pri CFB in CBC tudi tu sprememba inicializacijskega vektorja povzroči, da se identični izvorni sporočili preslikata v različni šifrirani sporočili.

- Verižna odvisnost: izvorno sporočilo ne vpliva na ključe, ki se uporabljajo v posameznih fazah enkricpije.
- Razširjanje napak: ena ali več napka v kateremkoli znaku šifriranega sporočila vpliva le na dešifriranje tega znaka, v točno tistem bitu, kot je napaka.
- Reševanje napak: OFB način lahko okreva po napakah, vendar pa se more sam sinhronizirati nazaj po izgubi bitov šifriranega sporočila, saj mu to uniči zaporedje dekripcijskih ključev (v tem primeru je zato potrebna eksplicitna resinhronizacija)
- Zahtevnost postopka je v primerjavi z CFB načinom manjša. Ker določeni koraki v vsaki fazi enkripciji niso odvisni od izvornega sporočila, jih lahko izračunamo že vnaprej, če poznamo ključ in inicializacijski vektor.
- Inicializacijski vektor ni nujno skrit, vendar pa ga moramo spremeniti, če drugič uporabimo isti OFB ključ, saj bi sicer napadalcem lahko precej olajšali razbijanje algoritma.

Izčrpno iskanje

Enkripcijski ključ stalne dolžine določa zgornjo mejo varnosti bločnega šifrirnega postopka, saj močno vpliva na izčrpno iskanje. Čeprav takšen ključ zahteva ali znano izvorno sporočilo ali pa izvorno sporočilo z redundantnimi informacijami, pa se je takšen pristop zelo uveljavil, saj je v večini primerov implementiran tako, da je računsko zelo učinkovit. Tovrsten bločni šifrirni postopek je običajno načrtovan tako, da je zelo računsko potratno zamenjati ključ v šifrirnem postopku (kar precej oteži izčrpno iskanje ključa), medtem ko enkripcija s ključem stalne dolžine ostane relativno učinkovita.

Za n bitni bločni šifrirni postopek s k bitnim ključem in podano majhno število (npr. (k+4)/n) parov izvornih in šifriranih sporočil lahko z izčrpnim iskanjem ugotovimo ključ v času 2^{k-1} matematičnih operacij. Izčrpno iskanje običajno izvedemo tako, da izbrano šifrirano sporočilo dešifriramo s testnim (potencialno pravim) ključem in preverjamo, če se pridobljeno dešifrirano sporočilo ujema z znanim (pravim) izvornim sporočilom. Običajno predvidevamo, da bomo pravi ključ našli po preizkušanju približno polovice izmed vseh možnih ključev.

Večkratna enkripcija

Če je bločni šifrirni postopek občutljiv na izčrpno iskanje (npr. zaradi neustrezne dolžine ključa), lahko varnost šifriranja izboljšamo z večkratnim šifriranjem istega bloka. Obstaja več načinov, kako izvesti takšno večkratno šifriranje. Uporabimo jih lahko tudi na sporočilih, ki so daljši od enega bloka in sicer z uporabo enega od prej omenjenih načinov uporabe bločnih šifrirnih postopkov. Poglejmo si na kratko nekaj načinov, kako zgraditi večkratno enkripcijo.

Kaskadni šifrirni postopek je spojitev 2 ali več bločnih postopkov, ki ima vsak svoj neodvisni ključ. Izvorno sporočilo je vhodni podatek v prvi postopek, izhodni podatek iz tega je nato vhodni podatek v drugi postopek … izhod iz zadnjega postopka pa je šifrirano sporočilo kaskadne spojitve postopkov. V najpreprostejšem primeru imajo vsi postopki k bitni ključ, vhodi in izhodi iz postopkov pa so n bitna sporočila. V kaskado lahko povežemo poljubne postopke ali pa tudi več enakih postopkov.

Večkratna enkripcija je podoben pristop kot kaskada L enakih šifrirnih postopkov, vendar pa posamezni postopki niso nujno neodvisni med sabo. Šifrirni postopki v določeni fazi večkratne enkripcije pa so lahko tako enkripcijski kot tudi dekripcijski bločni šifrirni postopki. Dva pomembna primera tovrstne enkripcije sta **dvojna enkripcija**, kjer se dvakrat zaporedno izvede ista enkripcija a z različnima ključema, ter **trojna enkripcija**, kjer je prva in tretja faza enkripcija z istima ključema (da privarčujemo pri shranjevanju in upravljanju s ključi), druga faza pa je dekripcija, ki uporablja nek drug ključ.

Napad na dvojno enkripcijo bi lahko naivno izvedli z izčrpnim iskanjem, ki pa je časovno še veliko bolj zahteven kot pri enojni enkripciji. Obstaja pa t.i. meet-in-the-middle tip napada, kjer napadalec prestreza sporočila in zahteva poznano izvorno sporočilo in ima časovno zahtevnost 2^{2k}, shranjevalne potrebe pa reda 2^{k}.

Podobno kot smo jih definirali že pri enojni enkripciji, je tudi pri večkratni enkripciji možno definirati večkratne načine delovanja bločnih šifrirnih postopkov. Tako na primer kombinacija treh enojnih CBC načinov zagotavlja t.i. trojni notrani CBC ali pa tudi trojni zunanji CBC.

3.5 DES – Data Encryption Standard

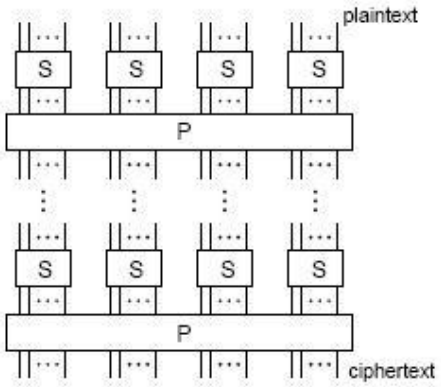
DES je najbolj znan bločni šifrirni postopek s simetričnim ključem. Kot prvi komercialno zanimiv moderen algoritem se je pojavil v 70. letih prejšnjega stoletja z odprtimi in v podrobnosti razloženimi postopki za implementacijo. Definiran je kot ameriški standard FIPS 46-2.

V osnovi DES temelji na dveh zelo splošnih konceptih – produktnih šifrirnih postopkih ter **Feistelovih šifrirnih postopkih**. Oba koncepta vključujeta ponavljanje skupne sekvence oz. zaporedja operacij.

Osnovna ideja produktnih šifrirnih postopkov je v tem, da zgradimo kompleksno enkripcijsko funkcijo s kombiniranjem večih preprostih operacij, ki same po sebi ne zagotavljajo dovolj visoke stopnje varnosti, sestavljene skupaj pa nivo varnost dvignejo na ustrezen nivo. Osnovne operacije, ki so gradniki produktnih postopkov, so na primer premeščanje (transpozicija), translacija in linearne transformacije, aritmetične operacija, modularne multiplikacije in preproste zamenjave.

Kot smo omenili že v uvodnih poglavjih, je produktni šifrirni postopek kombinacija dveh ali več transformacij, ki jo sestavimo z namenom, da bi bil rezultirajoči postopek bolj varen kot pa posamezne komponente, iz katerih je sestavljen.

Substitucijsko-permutacijsko omrežje je primer produktnega šifrirnega postopka, ki je sestavljen iz več korakov, ki vsak zajema po eno substitucijo (zamenjavo) in permutacijo. Prikazuje ga slika 2.



Slika 2 : Substitucijsko permutacijsko omrežje SP

Večina SP omrežij je dejansko iterirajoč bločni šifrirni postopek. Za slednega je značilno, da je postopek, ki vključuje zaporedno ponavljanje posebne interne funkcije, ki jo imenujemo »round« funkcija. Parametri takšnega postopka so število iteracij (krogov – rounds), bitna velikost bloka n in bitna velikost ključa k, iz katerega izpeljemo podključe – ključe posamezne iteracije. Za to da je celoten postopen obrnljiv (za enolično dekripcijo), je za vsak podključ interna funkcija takšnega postopka bijektivna preslikava vhodnih spremenljivk.

Feistelov šifrirni postopek je primer takšnega iterirajočega šifrirnega postopka, ki preslika 2t-bitno izvorno sporočilo (L_0,R_0) za t bitne bloke L_0 in R_o v šifiriran tekst (R_r,L_r) z uporabo r iteracij, ker je r ≥ 1. V vsakem krogu se na identičen način izvede preslikava in sicer:

v i-ti iteraciji se (L_{i-1},R_{i-1}) preslika v (L_i,R_i) tako da velja: $L_i = R_{i-1}$ in $R_i=L_{i-1} \oplus f(R_{i-1},K_i)$, kjer je K_i podključ, ki ga dobimo iz šifrirnega ključa K.

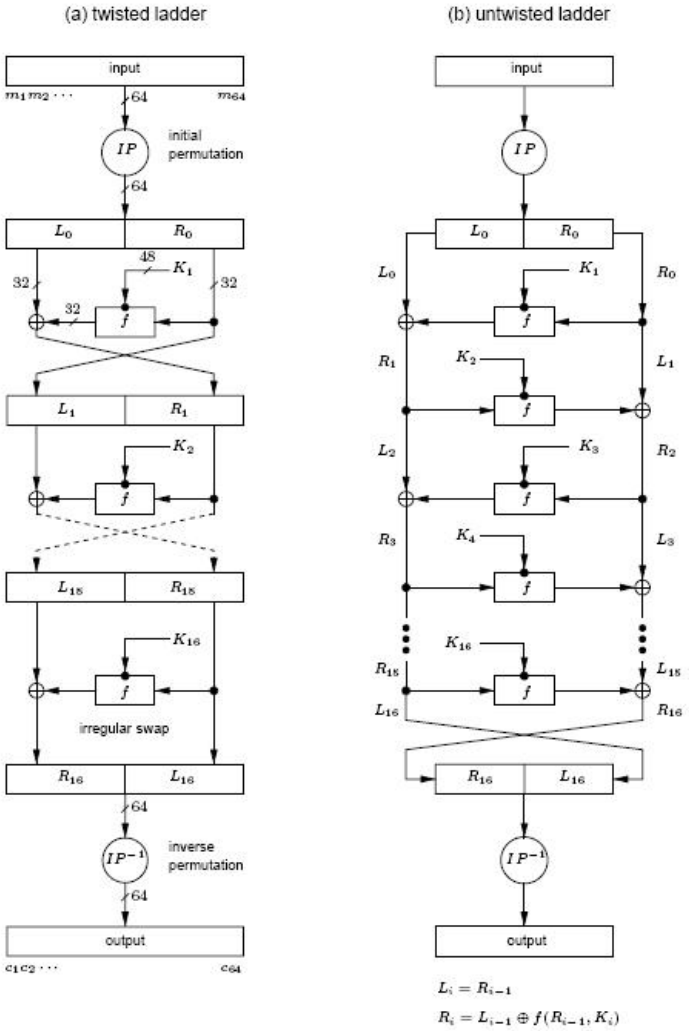
Običajno je v Feistelovem postopku r ≥ 3 in sodo število. Velja opozoriti, da je v Feistelovem postopku natančno predpisano, da sta v izhodnem sporočilu leva in desna polovica zamenjani. Dekripcija poteka na podoben način, le da se podključe uporablja v obratnem vrstnem redu. Funkcija f v zgoraj opisanem postopku je lahko prav tako nek produktni šifrirni postopek, značilno pa je to, da ni nujno obrnljiva, da bi bila izvedljiva inverzna preslikava v Feistelovem postopku.

3.5.1 DES algoritem

DES je Feistelov šifrirni postopek, ki procesira izvorna sporočila v blokih dolžine n = 64 bitov in na koncu proizvede bloke šifriranega sporočila, ki so prav tako 64 bitni. Shematski prikaz algoritma je na sliki 3.

Efektivna velikost skritega ključa K je k = 56 bitov, oz. bolj natančno, vhodni ključ K je podan kot 64 bitni ključ, od tega pa 8 bitov lahko uporabimo kot paritetne bite. S tako velikim ključem imamo na voljo 2^{56} možnih vrednosti ključa, s katerimi tako lahko

izvedemo le 2^{56} od 2^{64}! Možnih bijekcij na 64 bitnih blokih izvornega sporočila. Splošno je sprejeta razlaga, da so bili paritetni biti, ki zmanjšujejo efektivno velikost ključa za 8 bitov, namerno definirani zato, da so stroški izčrpnega iskanja manjši za faktor 256.



Slika 3 : DES enkripcijski algoritem

DES enkripcija se izvaja v 16 korakih. Iz vhodnega ključa K se generira šestnajst 48 bitnih podključev K_i, za vsak korak enkripcije po eden. V vsakem koraku uporabimo 8 stalnih, skrbno izbranih 6 do 4 bitnih substitucijskih preslikav S_i (t.i. S škatel), ki jih skupno označimo kar z S. 64 bitno izvorno sporočilo razdelimo v dva 32 bitna dela, ki ju označimo z L_0 in R_0. Vsak korak enkripcije je funkcionalno ekvivalenten, saj vzame 32 bitni polovici sporočila L_{i-1} in R_{i-1} iz prejšnjega koraka in proizvede 32 bitna izhoda L_i in R_i za vsak i med 1 in 16 in sicer takole:

$L_i = R_{i-1}$ in

$R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$, kjer je $f(R_{i-1}, K_i) = P(S (E (R_{i-1} \oplus K_i)))$

E tu predstavlja fiksno razširitveno permutacijo R_{i-1} iz 32 bitov na 48 bitov (vsi biti so uporabljeni vsaj enkrat, nekateri pa tudi dvakrat). P je še ena fiksna permutacija na

32 bitih. Začetna permutacija bitov se zgodi že pred prvim krogom enkripcije, po koncu zadnjega kroga najprej zamenjamo levo in desno polovico teksta, zatem pa še enkrat izvedemo bitno permutacijo, inverzno tisti pred prvim krogom. Dekripcija poteka z enakim ključem in enakim algoritmom, le da se podključe v določenem krogu uporablja v obratnem vrstnem redu kot v enkripcijskem postopku. Začetna in končna permutacija ter razširitev E in permutacija P so prikazani v tabelah 2 in 3, shema funkcije f pa je na sliki 4.

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

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

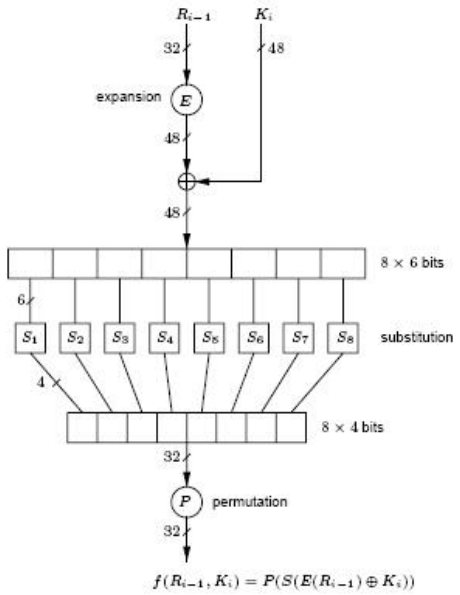
Tabela 2 : DES začetna permutacija IP in inverz IP

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

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

Tabela 3 : DES razširitev E in permutacija P

V poenostavljenem pogledu bi lahko na DES gledali tudi takole: na desni polovici sporočila (potem ko 32 bitov razširimo v 8 znakov po 6 bitov) se izvede od ključa odvisna substitucija na vsakem od 8 znakov, nato pa se uporabi fiksna bitna transpozicija in bite v posameznih znakih premeša na ta način, da na izhodu spet dobimo 32 bitov.



Slika 4 : DES funkcija f

Algoritem za generiranje ključev kot vhodni podatek jemlje 64 bitni vhodni ključ (vključno z biti paritete), na izhodu pa daje 16 48 bitnih podključev. V prvi fazi vsakega koraka genereracije podključev se določi posebna spremenljivka v_i, ki ima vrednost 1, če je i (korak generacije podključa, i zavzame vrednosti med 1 in 16) 1,2,9 ali 16 in vrednost 2, če ima i drugačno vrednost. (v_i so vrednost za levi pomik v 28 bitnih krožnih rotacijah, ki so opisane v nadaljevanju). 56 bitov ključa (efektivna velikost ključa) razdelimo v dve polovici (C_0,D_0), kjer vrstni red bitov v eni in drugi polovici določa posebna tabela PC1 (glej tabelo 4). V zadnji fazi vsakega koraka generacije podključev se nato dejansko zgenerira 48 bitni podključ in sicer tako, da najprej glede na vrednost spremenljivke v_i izvedemo levi krožni premik bitov v obeh polovicah C_{i-1} in D_{i-1} , nato pa tako premaknjeni polovici ponovno sestavimo skupaj in s pomočjo tabele PC2 (glej tabelo 4) izberemo 48 bitov, ki predstavljajo podključ v trenutnem koraku generacije podključev. Podobno postopamo tudi v vseh ostalih korakih, s tem da se polovici C_i in D_i v vsakem koraku seveda spreminjata.

PC1						
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
above for C_i; below for D_i						
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

PC2					
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

Tabela 4 : DES generiranje podključev - selekcija bitov PC1 in PC2

DES dekripcija se izvaja z istim algoritmom in istim ključem, le da se podključi uporabijo v obratnem vrstnem redu, torej od 16. proti prvemu. Postopek je sledeč: vpliv inverzne začetne permutacije bitov se izniči z začetno permutacijo v prvem koraku dekripcije, ta pa vrne kombinacijo (R_{16},L_{16}) (spomnimo se, da na koncu enkripcije zamenjamo obe polovici šifriranega teksta). Na levi polovici sporočila se izvede operacija $R_{16} \oplus f(L_{16},K_{16})$, kar je enako (ker je $L_{16} = R_{15}$ in $R_{16} = L_{15} \oplus f(R_{15},K_{16})$ $L_{15} \oplus f(R_{15},K_{16}) \oplus f(R_{15},K_{16}) = L15$. Prvi korak dekripcije je torej vrnil vrednosti (R_{15},L_{15}) oz. je invertiral 16. korak enkripcije. Opazimo lahko, da je invertiranje posameznega kroga neodvisno od funkcije f in specifične vrednosti ključa K_i. Ostalih 15 korakov dekripcije poteka podobno kot prvi in eden za drugim invertirajo soležne korake v enkripcijskem postopku, v skladu z obrnjenim vrstnim redom podključev. Slednje lahko generiramo z enakim algoritmom, kot je bil opisan zgoraj, ali pa jih že v fazi generacije obrnemo, torej začnemo s 16. podključem in končamo s prvim.

3.5.2 Značilnosti in prednosti algoritma DES

Za bločne šifrirne postopke obstaja precej zaželjenih lastnosti, ki naj bi jih vsebovali. Ti so med drugim: vsak bit šifriranega sporočila mora biti odvisen od vseh bitov ključa in vseh bitov izvornega sporočila. Prav tako ne sme biti nobenih očitnih statističnih odvisnosti med izvornim in šifriranim sporočilom, spreminjanje kateregakoli posameznega bita ključa ali izvornega sporočila mora spremeniti vsako šifrirano sporočilo z verjetnostjo ½ (torej z enako verjetnostjo, kot bi se bit povsem naključno spremenil). Spreminjanje bita v šifriranem sporočilo se mora odražati v nenapovedljivi spremembi v pridobljenem izvornem sporočilu. DES empirično gledano vsem tem pogojem zadostuje, nekatere posebnosti tega algoritma pa so opisane v nadaljevanju.

DES ima posebno lastnost, da če izvedemo bitni komplement tako na ključu kot na izvornem sporočilu, se bitno komplementira tudi šifrirano sporočilo. Dvojni komplement se namreč izniči v fazi, ko računamo novo vrednost funkcije f v enačni za desni del sporočila oz. konkretneje v XOR operaciji v funkciji f. Po še eni XOR operaciji dobimo komplementarno vrednost levega dela šifriranega sporočila, torej točno tisto, kar smo malo prej zapisali. Opisana lastnost DES algoritma pa ne predstavlja bistvenega olajšanja za potencialne napadalce, saj si z njo ne morejo kaj posebej pomagati. V najboljšem primeru se število ključev, ki jih morajo pregledati pri izčrpnem iskanju, zmanjša iz 2^{55} na 2^{54}.

Za DES je značilno, da so nekateri izmed ključev zelo ali pa vsaj delno šibki (glej tabeli 5 in 6). Če sta namreč podključa K_1 in K_{16} enaka, potem dobimo tako pri enkripciji kot pri dekripciji enako zaporedje ključev, $K_1=K_{16}$, $K_2=K_{15}$ itn. Enkripcija in dekripcija zato potem v bistvu sovpadata. Takšnim ključem pravimo palindromični ključi in so v smislu enkripcijskih postopkov šibki ključi. Šibek ključ za DES algoritem sicer definiramo kot tak ključ, ki, če ga uporabimo v dveh zaporednih enkripcijskih postopkih nad izvornim sporočilom, vrne nazaj izvorno sporočilo, čemur v matematiki pravimo involucija. Par DES delno šibkih ključev pa sta taka dva ključa (K_1,K_2), ki vrneta spet izvorno sporočilo, če sta zaporedno uporabljena za enkripcijo izvornega sporočila. Enkripcija z eni ključem šibkega para ključev deluje povsem enako kot deluje dekripcija z drugim ključem iz para ključev. Kot se izkaže, ima DES 4 šibke

ključe in 6 parov delno šibkih ključev, ki so zapisani v naslednji dveh tabelah, kjer so ključi zapisani v polovicah C in D, eksponenta 14 in 28 pa pomenita 14 oz. 28 ponovitev vrednosti bita 0 ali 1:

C_0	D_0
$\{0\}^{28}$	$\{0\}^{28}$
$\{1\}^{28}$	$\{1\}^{28}$
$\{0\}^{28}$	$\{1\}^{28}$
$\{1\}^{28}$	$\{0\}^{28}$

Tabela 5 : DES šibki ključi

Prvi ključ		Drugi ključ	
C_0	D_0	C_0	D_0
$\{01\}^{14}$	$\{10\}^{14}$	$\{10\}^{14}$	$\{10\}^{14}$
$\{01\}^{14}$	$\{10\}^{14}$	$\{10\}^{14}$	$\{01\}^{14}$
$\{01\}^{14}$	$\{0\}^{28}$	$\{10\}^{14}$	$\{0\}^{28}$
$\{01\}^{14}$	$\{1\}^{28}$	$\{10\}^{14}$	$\{1\}^{28}$
$\{0\}^{28}$	$\{01\}^{14}$	$\{0\}^{28}$	$\{10\}^{14}$
$\{1\}^{28}$	$\{01\}^{14}$	$\{1\}^{28}$	$\{10\}^{14}$

Tabela 6 : DES delno šibki ključi

DES algoritma se lahko lotimo z linearno in diferencialno kriptoanalizo. Linearna kriptoanaliza DES algoritma je, ob predpostavki, da je mogoče dobiti veliko znanih izvornih sporočil, še najmočnejši način napada na DES algoritem. Linearna kriptoanaliza je možna tudi v situaciji, ko poznamo le šifrirana sporočila, pod pogojem, da je poznamo tudi nekaj redundance v pripadajočih izvornih sporočilih (npr. pariteta bitov ali pa 0-ti biti visokega reda v ASCII znakih).

Diferencialna kriptoanaliza je najbolj splošno kriptoanalitično orodje za napade na različne bločne šifrirne postopke, med drugim tudi DES. V primerjavi z linearno kriptoanalizo pa diferencialna običajno deluje le na izbranih izvornih sporočilih.

Kompleksnost napadov na DES algoritem prikazuje tabela 7. Odstotki pomenijo uspešnost pri napadih s specifičnimi parametri, kompleksnost procesiranja pa prikazuje oceno stroškov pri napadih, kar pri izčrpnem napadu na DES pomeni število operacij, ki jih je potrebno izvesti. Kar se tiče kompleknosti napadov glede shranjevanja podatkov, imata oba tipa napada minimalne zahteve.

Metoda napada	Kompleksnost podatkov		Kompleksnost shranjevanja	Kompleksnost procesiranja
	znani	izbrani		
Izčrpno predračunanje	-	1	2^{56}	1
Izčrpno iskanje	1	-	zanemarljivo	2^{55}
Linearna kriptoanaliza	2^{43} (85%)	-	za tekste	2^{43}
	2^{38} (10%)	-	za tekste	2^{50}
Diferencialna kriptoanaliza	-	2^{47}	za tekste	2^{47}
	2^{55}	-	za tekste	2^{56}

Tabela 7 : Moč DES algoritma proti različnim napadom

Navkljub nekaterim zanimivim lastnostim pa je DES algoritem možno razbiti tako z izčrpnim iskanjem kot tudi z linearno in diferencialno kriptoanalizo. S temi postopki je možno pridobiti DES ključ, posledično pa seveda tudi izvorno sporočilo, seveda pa zato potrebujemo tudi dovolj šifriranih sporočil.

Čeprav je bil DES algoritem ob času svojega nastanka zelo dobro sprejet in implementiran v številne aplikacije, pa se je z leti izkazalo, da vendarle ni tako zelo varen, oz. vsaj ne toliko varen, da bi uspel ustrezen nivo varnosti zagotavljati v moderni računalniški družbi, kjer se zmogljivosti računalnikov iz leta v leto močno povečujejo. Napadi na algoritem DES z grobo silo (izčrpno preiskovanje) so tako npr. na začetku leta 1998 potrebovali še 39 dni, da so algoritem razbili. Že leta 1999 je mreža 100.000 prostovoljcev potrebovali le še 22 ur, da so zlomili DES. Danes velja, da algoritem DES ni več dovolj varen za uporabo v praktičnih aplikacijah, saj je dolžina ključa premajhna, da bi preprečevala razbitje, že vsaka malo bolje opremljena skupina napadalcev pa DES lahko zlomi.

3.6 AES – Advanced Encryption Standard

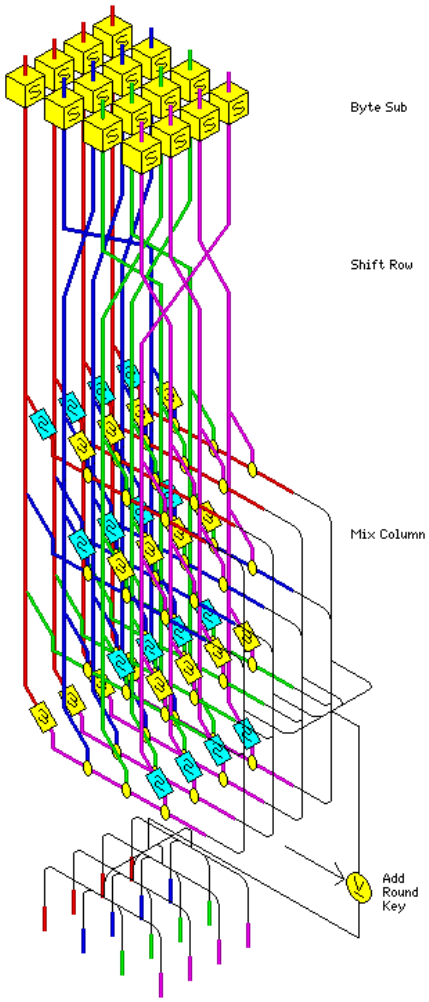
Raziskovalci so se že dokaj kmalu lotili izboljševanja algoritma DES oz. iskanje nekega bolj varnega algoritma. Ena izmed idej je bil t.i. trojni DES (**3-DES**), kar pomeni, da bi DES enkripcijo izvedli trikrat zapored. To bi pomenilo trojno enkripcijo, zato bi bil postopek trikrat počasnejši, bil pa bi 2^{56} krat varnejši za napad z grobo silo. Pri tem je zanimivo, da predlagan postopek ne bi pomenil, da se dejansko trikrat zapored izvede enkripcija z istim ključem, ampak je bila v drugi fazi predvidena dekripcija z nekim drugim ključem.

Kljub temu da je ta postopek precej bolj varen kot klasičen DES, pa se je mednarodna kriptografska skupnost vseeno odločila poiskati naslednika DESa in sicer kar prek mednarodnega razpisa. Postavljene so bili določene zahteve, raziskovalci-kriptografi pa so bili pozvani, naj predlagajo bločni šifrirni postopek nove generacije. Na razpis je prispelo kar precej predlogov, na koncu pa je bil za AES –

Advanced Encryption Standard, kot se je za novi algoritem uveljavilo ime, izbran algoritem RIJNDAEL, ki sta predlagala nizozemska raziskovalca Daemen in Rijmen.

Za Rijndael je značilno, da je v prvi vrsti zelo varen algoritem, po drugi strani pa je tudi precej hiter. Operira z bloki dolžine 256 bitov (šestnajst 8 bitnih znakov), uporablja pa ključe različne dolžine. Minimalna dolžina ključa je 128 bitov, kar je v primerjavi z algoritmom DES precejšen korak naprej, kar se tiče varnosti. Po oceni tehnologije iz leta 2000 naj bi za razbijanje algoritmov, ki temeljijo na 128 bitnem ključu, ne zadoščali niti najsodobnejši računalniški sistemi, s kakršnimi razpolaga na primer vojska. Tudi danes lahko vsekakor zatrdimo, da je takšen algoritem varen. Čeprav torej že 128 bitni ključ zadošča za dovolj visok nivo varnosti kot ga potrebujemo v tem trenutku, pa je v algoritmu Rijndael možno uporabiti tudi ključe dolžine 196 in 256 bitov. Rijndael dekripcija poteka v obratnem vrstnem redu kot enkripcija, možna pa je uporaba tudi drugačnih parametrov kot pri enkripciji.

Nekoliko kompleksnejša shema Rijndael algoritma je na spodnji sliki.



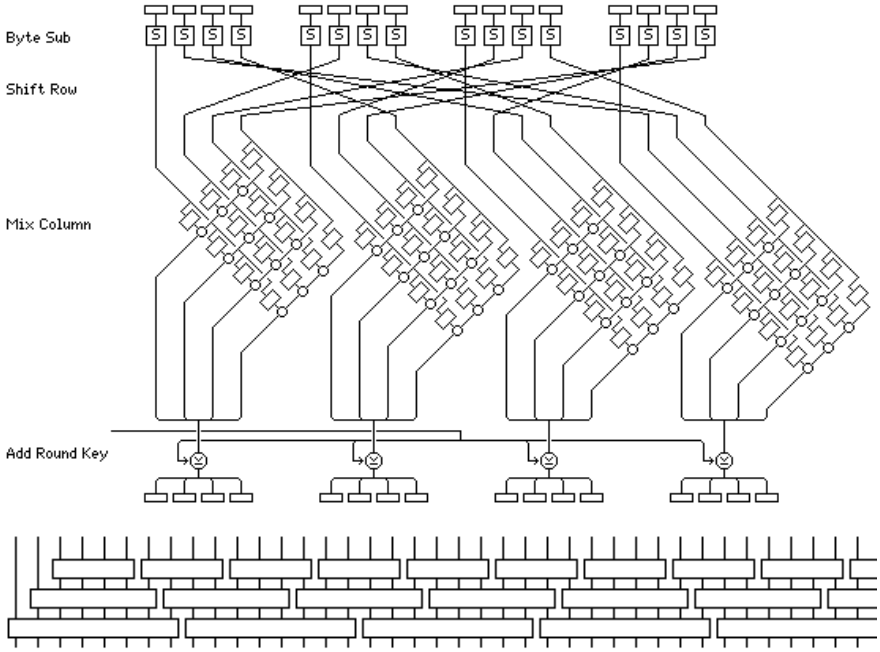
Slika 5 : Shema Rijndael algoritma

Kot je razvidno že iz prikazana sheme algoritma, so za Rijndael značilne 4 osnovne operacije:

- Bajtna zamenjava (byte substitution) : S škatla (substitucija)

- Mešanje vrstic (shift row) : P škatla
- Mešanje stolpcev (mix column) : substitucija, ki temelji na aritmetiki končnih polj
- Dodajanje ključa (add round key) : XOR trenutnega bloka z delom razširjenega ključa

Algoritem je z drugega zornega kota prikazan tudi na spodnji sliki. Spodnja polovica slike prikazuje zanimivo lastno Rijndael algoritma. Možno je namreč sestavljati različne dolžine ključa kot lego kocke.



Slika 6 : Rijndael algoritem z drugega zornega kota

3.7 FEAL – The Fast Data Encipherment Algorithm

FEAL je družina algoritmov, ki so igrali zelo pomembno vlogo pri razvoju in izboljšavah številnih naprednih kriptoanalitičnih tehnik, med drugim tudi linearne in diferencialne kriptonanalize. FEAL-N preslikav 64 bitna izvorna sporočila v 64 bitna šifrirana sporočila s 64 bitnim tajnim ključem. Gre za N fazni Feistelov bločni šifrirni postopek, ki je precej podoben DES algoritmu, bistvena razlika pa je precej bolj enostavna f funkcija. Celoten postopek je nekoliko razširjen z začetnimi in končnimi koraki, v katerih izvedemo XOR operacijo nad polovicami sporočil, prav tako pa tudi nad podključi in polovicami sporočil.

Cilj pri načrtovanju algoritma FEAL je bila hitrost in enostavnost, še posebno za programsko opremo, ki deluje na 8 bitnih mikroprocesorjih (npr. čipne kartice). FEAL uporablja bajtno orientirane operacije (8 bitno seštevanje po modulu 256, 2 bitni

pomik v levo, XOR), izogiba se bitnim permutacijam in uporabi tabel, zagotavlja pa majhno velikost kode.

Na začetku je bila za komercialno uporabo predlagana različica s 4 fazami enkripcije (FEAL-4), ki je bila mišljena kot precej bolj hitrejša alternativa DESu, a se je zanjo že kmalu izkazalo, da je precej manj varna, kot so si avtorji želeli. Tudi FEAL-8, ki je bil predlagan zatem, se je izkazal za ne dovolj varnega. Šele FEAL-16 in FEAL-32 zagotavlja varnost, ki je primerljiva z DES, vendar pa se s številom faz enkripcije močno povečuje kompleksnost procesiranja. Poleg tega se da s pomočjo tabel, kamor si shranjujemo določene podatke, močno povečati hitrost DES implementacij, pri FEAL algoritmih pa je to precej težje izvedljivo.

Poglejmo si na kratko, kako deluje algoritem FEAL-8. V prvem koraku izračunamo 16 16-bitnih podključev s posebnim algoritmom, ki bo opisan v nadaljevanju. Podobno kot pri DES tudi tu razdelimo 64 bitni blok izvornega sporočila v dva dela, M_L in M_R. Kot smo omenili v prejšnjem odstavku, se pred pričetkom zaporedne enkripcije v FEAL algoritmu izvede še XOR operacija nad začetnimi vrednostmi podključev in izvornim sporočilom in sicer takole: $(M_L , M_R) \oplus ((K_8, K_9) , (K_{10}, K_{11}))$. Med obema polovicama tako dobljenega sporočila (L_0, R_0) nato še enkrat izvedemo XOR operacijo. Sledi 8 korakov, kjer se izvajajo identične operacije. Levi del novega sporočila je enak desnemu delu prejšnjega sporočila (iz prejšnjega koraka enkripcije), desni del novega sporočila pa se izračuna po formuli: $R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$, torej enako kot pri DES, le da je funkcija f tokrat precej drugačna. Funkcija f kot vhodna parametra vzame desni del prejšnjega sporočila in ključ iz prejšnje faze FEAL enkripcije. Funkcija na to izvede preslikavo med 32 bitnim delom sporočila in 16 bitnim ključem in na izhodu vrne 32 bitno vrednost. Znotraj funkcije se dvakrat izvedeta dve bajtno naravnani substituciji (t.i. S škatli). Vsaka od teh substitucij preslika para 8-bitnih vhodnih podatkov v 8-bitni izhod, v skladu s spodnjo tabelo. Obe S škatli prav tako dodata en bit d (vrednosti 0 ali 1) v 8 bitne argumente, ignorirata višek najbolj zgornjega bita in rezultat krožno zavrtita v levo za 2 bita (ROT2). Izvajata torej funkcijo: $S_d (x,y) = ROT2(x + y + mod 256)$. Po 8 fazah, kjer se ponovijo identične operacije, se najprej izvede XOR operacija med levo in desno polovico trenutne vrednosti šifriranega sporočila, rezultat pa postane nov levi del šifriranega sporočila. Zatem se izvede še XOR operacija $(R_8 , L_8) \oplus ((K_{12}, K_{13}) , (K_{14}, K_{16}))$, v zadnjem koraku enkripcije pa se vrstni red polovic še zamenja in tako dobimo končno vrednost šifriranega sporočila.

	$U \leftarrow f(A,Y)$	**$U \leftarrow f_K(A,B)$**
t_1	$(A_0 \oplus A_1) \oplus Y_0$	$A_0 \oplus A_1$
t_2	$(A_2 \oplus A_3) \oplus Y_1$	$A_2 \oplus A_3$
U_1	$S_1 (t_1, t_2)$	$S_1 (t_1, t_2 \oplus B_0)$
U_2	$S_0 (t_2, U_1)$	$S_0 (t_2, U_1 \oplus B_1)$
U_0	$S_0 (A_0, U_1)$	$S_0 (A_0, U_1 \oplus B_2)$
U_3	$S_1 (A_3, U_2)$	$S_1 (A_3, U_2 \oplus B_3)$

Tabela 8 : Vrednost U za FEAL funkcije f in fk

Generacija podključev v FEAL algoritmu poteka tako, da najprej izvedemo inicializacijo, kjer nastavimo začetne vrednosti treh 8 bajtnih vektorjev: $U^{(-2)} = 0$, $U^{(-1)}$ (prvih 32 bitov ključa) in $U^{(0)}$ (drugih 32 bitov ključa). V postopku generacije podključev manipuliramo z 32 bitnim vektorjem U, ki je sestavljen iz 4 8-bitnih delov U_0, U_1, U_2 in U_3. Nove vrednosti vektorja U se v vsaki fazi generacije FEAL podključev izračunajo s pomočjo funkcije f_K ($U^{(i-2)}$, $U^{(i-1)} \oplus U^{(i-3)}$), katere delovanje je prav tako prikazano v zgornji tabeli. Iz vektorjev U_i pa nato v vsaki fazi generacije podključev dobimo dva nova podključa (U_0 in U_1 določata K_{2i-2}, U_2 in U_3 pa podključ K_{2i-1}), kar v 8 fazah prinese 16 FEAL podključev.

FEAL dekripcijski postopek v bistvu poteka identično kot malo prej opisani postopek dekripcije z istim ključem in šifriranim sporočilom (R_8, L_8) kot vhodnim podatkom, le da je vrstni red uporabljenih ključev obrnjen. Pri tem se podključi ((K_{12}, K_{13}) , (K_{14}, K_{16})) uporabijo pri začetnem XORu, podključi ((K_8, K_9) , (K_{10}, K_{11})) pa pri končnem XORu.

Kot smo nakazali že na začetku opisa FEAL algoritma, lahko FEAL s 64 bitnim ključem posplošnimo na N faz, kjer je N sodo število. Priporočljivo je, da je N izbran tako, da velja $N = 2^x$, kjer torej x = 3 pomenil FEAL-8. FEAL-N uporablja N+8 16 bitnih podključev: K_0 … K_{N-1} v določeni fazi enkripcije i; K_N … K_{N+3} za začetni XOR in K_{N+4} … K_{N+7} za končni XOR. Generacija podključev poteka na analogen način kot smo ga opisali način, le da postopek posplošimo na število faz N.

Algoritem FEAL lahko razširimo tako, da uporabimo 128 bitni ključ, s čimer dobimo algoritem FEAL-NX, za katerega je značilno, da uporablja nekoliko drugačen postopek generacije podključev. Ključ najprej razdelimo v 64-bitni polovici (K_L, K_R). K_R nato razdelimo naprej na dva 32 bitna dela (K_{R1}, K_{R2}). Za vsak i med 1 in (N/2)+4 definiramo $Q_i = K_{R1} \oplus K_{R2}$ za i = 1 mod 3, $Q_i = K_{R1}$ za i = 2 mod 3 in $Q_i = K_{R2}$ za i = 0 mod 3. Izraz ($U^{(i-1)} \oplus U^{(i-3)}$) v izrazu za f_K zamenjamo z ($U^{(i-1)} \oplus U^{(i-3)} \oplus Q_i$). Za $K_R = 0$, FEAL-NX preide v navadni FEAL-N, kjer je K_L 64 bitni FEAL-N ključ K.

Spodnja tabela prikazuje znane napade na FEAL algoritme. LC in DC pomenita napade z linearno in diferencialno kriptoanalizo. Zapisani časi veljajo za standardne osebne računalnike oz. delovne postaje.

Metoda napada	Kompleksnost podatkov		Kompleksnost shranjevanja	Kompleksnost procesiranja
	Znani podatki	Izbrani podatki		
FEAL-4 LC	5	-	30 kB	6 minut
FEAL-6 LC	100	-	100 kB	40 minut
FEAL-8 LC	2^{24}	-	280 kB	10 minut
FEAL-8 DC	-	2^7 parov	-	2 minuti

FEAL-16 DC	-	2^{29} parov	-	2^{30} operacij
FEAL-24 DC	-	2^{45} parov	-	2^{46} operacij
FEAL-32 DC	-	2^{66} parov	-	2^{67} operacij

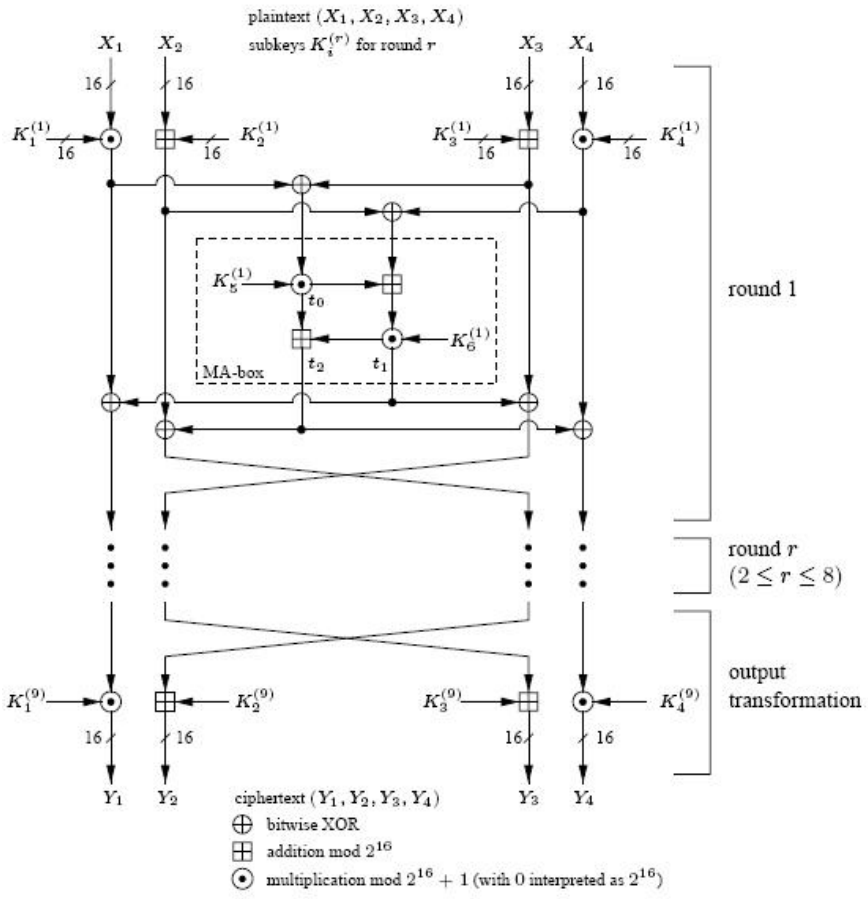
Tabela 9 : Moč FEAL algoritma proti različnim napadom

3.8 IDEA – International Data Encryption Algorithm

Bločni šifrirni postopek IDEA šifrira 64 bitne bloke izvornega sporočila v 64 bitne bloke šifriranega sporočila z uporabo 128 bitnega vhodnega ključa. Shema algoritma je na sliki 7. Njegova podlaga je posplošen model Feistelove strukture. Sestavljen je iz 8 računsko identičnih faz, ki jim sledi še transformacija izhodnega rezultata. Določena faza enkripcije r uporablja 16 bitne podključe $K_i^{(r)}$, kjer je i med 1 in 6, za transformacijo 64 bitnega vhoda X v štiri izhodne bloke dolžine 16 bitov, ki so potem vhod v naslednjo fazo enkripcije. Osma faza vrne vrednost na enak način, ta izhodna vrednost pa nato preide v dodatno transformacijo, ki uporabi štiri dodatne podključe $K_i^{(9)}$, kjer je i med 1 in 4, za generacijo končne oblike šifriranega sporočila. Tudi ti dodatni podključi so izpeljani iz vhodnega 128 bitnega ključa K.

Prevladujoč koncept v algoritmu IDEA je mešanje operacij iz treh različnih algebrajskih skupin 2^n elementov. Ustrezne skupinske operacije na podblokih a in b bitne dolžine n = 16 so bitna XOR operacija $a \oplus b$, seštevanje po modulu 2^n (a+b) mod AND 0xFFFF in modificirano množenje po modulu 2^n+1.

Za lažjo nadaljnjo razlago označimo omenjene operacije z (1), (2) in (3)



Slika 7 : IDEA enkripcijski algoritem

Postopek enkripcije se, podobno kot pri DES in FEAL algoritmih, prične z generacijo podključev – najprej generiramo šest ključev za 8 faz ponavljanja jedrnega dela algoritma, nato pa še 4 ključe za transformacijo izhoda. Vhodno 64 bitno sporočilo razdelimo na štiri 16 bitne bloke (X_1,X_2,X_3,X_4), nato pa zaporedno osemkrat izvedemo sledeč postopek:

$X_1 = X_1$ (3) $K_1^{(r)}$, $X_4 = X_4$ (3) $K_4^{(r)}$, $X_3 = X_3$ (2) $K_3^{(r)}$, $X_2 = X_2$ (2) $K_2^{(r)}$

$t_0 = K_5^{(r)}$ (3) (X_1 (1) X_3), $t_1 = K_6^{(r)}$ (3) (t_0 (2)(X_2 (1) X_4)), $t_2 = t_0$ (2) t_1

$X_1 = X_1$ (1) t_1, $X_4 = X_4$ (1) t_2, a = X_2 (1) t_2, $X_2 = X_3$ (1) t_1, X_3 = a

Tako dobljene vrednosti (X_1,X_2,X_3,X_4) nato še transformiramo in sicer tako, da velja:

$Y_1 = X_1$ (3) $K_1^{(9)}$, Y4 = X_4 (3) $K_4^{(9)}$, $Y_2 = X_3$ (2) $K_2^{(9)}$, $Y_3 = X_2$ (2) $K_3^{(9)}$

Algoritem za generacijo podključev pri IDEA algoritmu deluje takole:

Vhodni ključ K razdelimo na osem 16 bitnih delov, ki postanejo prvih osem podključev. Ostalih 48 podključev pa dobimo tako, da vhodni ključ K krožno premikamo v levo za 25 bitov in v vsakem koraku (torej po vsakem krožnem premiku) trenutna vrednost 16 bitnih blokov predstavlja naslednjih osem podključev za IDEA algoritem.

IDEA dekripcija poteka na zelo podoben način kot enkripcija, z istim vhodnim ključem K in šifriranim sporočilom kot vhodnim podatkom v dešifrirni postopek, nekoliko drugačen pa je postopek generacije podključev. Najprej se zgenerirajo vsi enkripcijski podključi, ti pa se nato uporabijo za izračun dekripcijskih podključev v skladu s spodnjo tabelo.

Faza r	$K'_1^{(r)}$	$K'_2^{(r)}$	$K'_3^{(r)}$	$K'_4^{(r)}$	$K'_5^{(r)}$	$K'_6^{(r)}$
r = 1	$(K_1^{(10-r)})^{-1}$	$-K_2^{(10-r)}$	$-K_3^{(10-r)}$	$(K_4^{(10-r)})^{-1}$	$K_5^{(9-r)}$	$K_6^{(9-r)}$
$2 \leq r \leq 8$	$(K_1^{(10-r)})^{-1}$	$-K_3^{(10-r)}$	$-K_2^{(10-r)}$	$(K_4^{(10-r)})^{-1}$	$K_5^{(9-r)}$	$K_6^{(9-r)}$
r = 9	$(K_1^{(10-r)})^{-1}$	$-K_2^{(10-r)}$	$-K_3^{(10-r)}$	$(K_4^{(10-r)})^{-1}$	-	-

Tabela 10 : IDEA dekripcijski ključi

Kar se tiče varnosti postopka IDEA, za popoln 8 fazni algoritem še ni znanega napada, ki bil bolj učinkovit kot izčrpno iskanje, razen seveda v primeru uporabe šibkih ključev. V tem trenutku se zdi, da varnost postopka IDEA omejujejo le težave, ki izhajajo iz relativno majhne dolžine bloka sporočil v primerjavi z dolžino ključa.

3.9 SAFER – Secure and Fast Encryption Routine

SAFER K-64 je ponavljajoč bločni šifrirni postopek s 64 bitnimi in šifriranimi sporočili in 64 bitnim ključem. Sestavljen je iz r enakih faz, ki jim sledi še izhodna transformacija. Sprva je bilo za ta algoritem predlaganih 6 faz, kar pa so kasneje spremenili v 8 faz z nekoliko spremenjenim algoritmom za generiranje podključev, kar je rezultiralo v algoritmu SAFER SK-64. V obeh primerih algoritma za generiranje podključev iz 64 bitnega zunanjega ključa razvijeta 2r+1 podključev, ki je vsak dolg 64 bitov (2 podključa se uporabiti v vsaki fazi enkripcijskega algoritma, 1 pa še v izhodni transformaciji). Algoritem SAFER je sestavljen skoraj v celoti iz preprostih bajtnih operacij, z izjemo bajtnih rotacij v postopku generiranja podključev. To je tudi razlog, da je SAFER zelo uporaben (v nasprotju z algoritmom FEAL) za uporabo v procesorjih z majhno velikostjo besed, npr. čipnih karticah.

V nadaljevanju bomo nekoliko bolj v podrobnosti opisali algoritem SAFER K-64., ki je shematsko prikazan na sliki 8. Postopek se prične z generacijo 2r+1 64 bitnih podključev po posebnem algoritmu, ki je opisan nekoliko kasneje. 64 bitno izvorno sporočilo razdelimo na 8 delov (X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8) oz bajtov. Vsaka faza se prične z XOR operacijo med 1.,4.,5. in 8. bajtom podključa in pripadajočim delom trenutne vrednosti vhodnega sporočila. Preostali 4 bajti ključa se nato prištejejo pripadajočim delom trenutne vrednosti vhodnega sporočila po modulu 256. V drugem koraku sledi izvajanje t.i. S škatel, ki so tukaj obrnljive bajt-bajt substitucije, ki uporabljajo eno fiksno 8 bitno bijekcijo. V tretjem koraku vsake faze se postopka XOR in seštevanja po modulu 256 opisana zgoraj zamenjata in izvedeta še enkrat

(XOR tokrat na 2.,3.,6. in 7. bajtu podključa itd). V četrtem koraku sledi uporaba linearne transformacije f (t.i. pseudo-Hadamardovega transforma), ki se izvede na 3 nivojski linearni plasti, ki je posebej namenjena za hitro razpršitev. Linearna transformacija f kot vhodne podatke jemlje dva 8 bitna podatka in izvede sledečo funkcijo:

f(L + R) = (2L+R, L+R), kjer seštevanje poteka po modulu 256

Trije nivoji takšne linearne transformacije se izvajajo takole:

$(X_j,X_{j+1}) = f(X_j,X_{j+1})$, za j= 1,3,5,7

$(Y_1,Y_2) = f(X_1,X_3)$, $(Y_3,Y_4) = f(X_5,X_7)$

$(Y_5,Y_6) = f(X_2,X_4)$, $(Y_7,Y_8) = f(X_6,X_8)$

Za j od 1 do 8 : $Y_j = X_j$

$(Y_1,Y_2) = f(X_1,X_3)$, $(Y_3,Y_4) = f(X_5,X_7)$

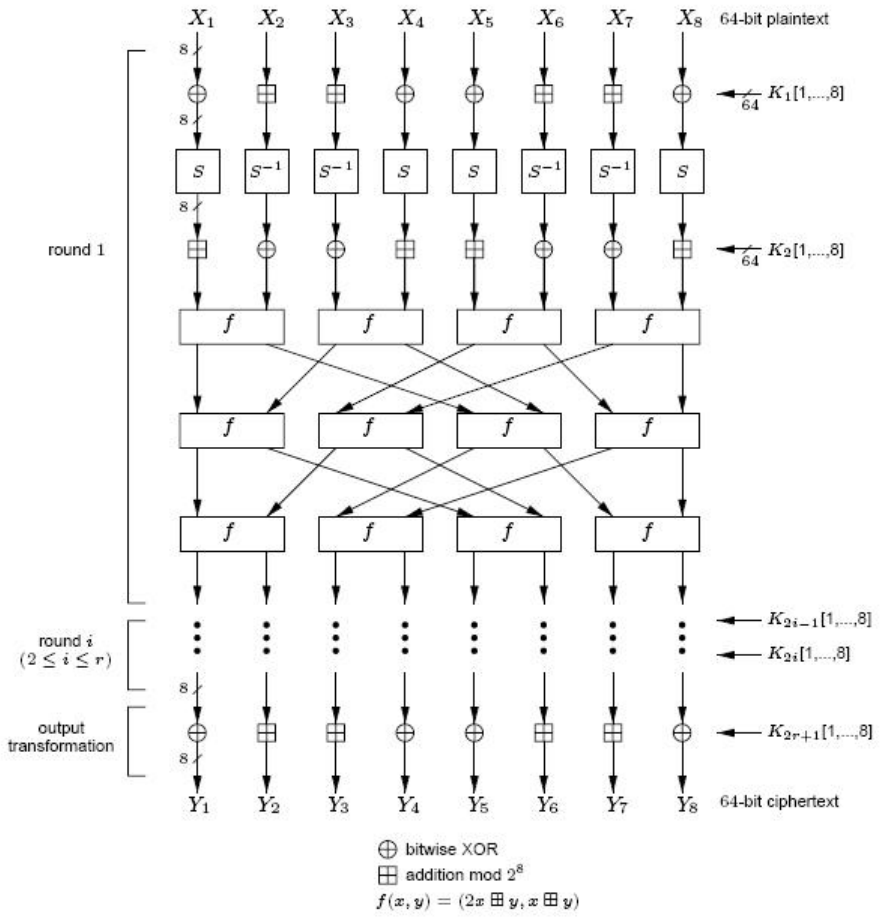
$(Y_5,Y_6) = f(X_2,X_4)$, $(Y_7,Y_8) = f(X_6,X_8)$

Za j od 1 do 8 : $Y_j = X_j$

S tem se zaključi ena faza enkripcije, ki se skupno ponovi r krat. Na koncu pa se izvede še izhodna transformacija in sicer takole:

Za j = 1,4,5,8: $Y_j = X_j \oplus K_{2r+1}[j]$, za j = 2,3,6,7: $Y_j= X_j + K_{2r+1}[j] \bmod 256$.

Za razliko od Feistelovih bločnih šifrirnih postopkov se operacije za enkripcijo in dekripcijo v algoritmu SAFER razlikujejo. Na SAFER lahko gledamo tudi kot substitucijsko permutacijsko omrežje.



Slika 8 : SAFER K-64 postopek

Algoritem za generacijo podključev v šifrirnem postopku SAFER jemlje na vhodu 64 bitni zunanji ključ K in na izhodu vrne 2r+1 64 bitnih podključev K_1 … K_{2r+1}, deluje pa takole:

Z R[i] označimo 8 bitno pomnilniško mesto, z $B_i[j]$ pa j-ti bajt B_i, ki predstavlja t.i. key bias. Slednjega v SAFER algoritmu uporabljamo zaradi eliminacije šibkih ključev, ki smo jih omenili pri DES in IDEA algoritmih. 64 bitov vhodnega ključa shranimo v osem 8 bitnih pomnilniških mest. Ti bajti predstavljajo že vrednost prvega podključa K_1. Za vse naslednje podključe i = 2 do 2r+1 pa najprej premaknemo bajte ključa za 3 bite v levo in nato po modulu 256 prištejemo key bias. Torej:

Za j od 1 do 8: R[j] = R[j] ←

Za j od 1 do 8: $K_j[j] = R[j] + B_i[j] \bmod 256$

S škatle in inverzne S škatle, ki se uporabljajo v postopku SAFER enkripcije, so konstante, ki jih dobimo po sledečem postopku.

g = 45, s[0] = 1, $S_{inv}[1] = 0$

za i med 1 in 255:

$t = g \cdot S[i-1] \bmod 257$

$S[i] = t$

$S_{inv}[t] = i$

Na koncu: $S[128] = 0$ in $S_{inv}[0] = 128$

Naj omenimo še to, da se pri generaciji podključev uporablja funkcija $S(x) = g^x \bmod 257$, kjer je g primitiv vrednosti g = 45. Ta preslikava je nelinearna. Inverzna S škatla temelji na logaritemski funkciji z osnovo g.

Aditivni key biasi, ki se uporabljajo pri generaciji podključev, so 8 bitne konstante, ki naj bi se obnašale kot naključna števila, denifirani pa so takole: $B_i[j] = S [S [9i+j]]$ za i med 2 in 2r+1 in j med 1 in 8. Primer takšnega biasa je: B_2(22,115,59,30,142,112,189,134).

Že na začetku poglavja o algoritmu SAFER smo omenili, da je bolj varna od osnovne različice algoritma SAFER K-64 t.i. SAFER SK-64 različica, za katero je značilen ojačan postopek za generacijo podključev. V primerjavi z osnovno verzijo slednja vključuje tri spremembe. Po inicializaciji R[i] v prvem koraku se nastavi: $R[9] = R[1] \oplus R[2] \oplus \ldots \oplus R[8]$. Temu sledi tudi sprememba v obeh korakih, ki se ponovita 2r+1 enkrat in sicer: rotacija bitov v levo sedaj velja tudi za R[9], torej za j med 1 in 9. Prava tako pa se spremeni postopek za generacijo novega podključa in sicer tako, da velja:

za j od 1 do 8: $K_j[j] = R[((i+j-2] \bmod 9) +1] + B_i[j] \bmod 256$

Do danes še ni znan noben napad z izjemo izčrnega iskanja, ki bi razbil SAFER SK-64 algoritem.

SAFER K-64 dekripcija uporablja enak ključ K in podključe K_i kot se uporabljajo pri enkripciji. Vsak enkripcijski korak se pri dekripciji ponovi, a v obratnem vrstnem redu. Začne se z vhodno transformacijo s podključem K_{2r+1} (XOR odštevanje), ki razveljavi enkripcijsko izhodno transformacijo. Sledi r dekripcijskih faz, kjer se zaporedoma uporablja podključe K_{2r} do K_1 (v vsaki fazi seveda dva podključa). Vsaka faza se začne s 3 koračnim inverznim linearnim nivojem s funkcijo f_{inv} (L,R) = (L – R, 2R – L), kjer je odštevanje po modulu 256. Nato sledi še faza odštevanje-XOR, pa inverzna substitucija, (z zamenjanim S in S_{inv}) ter na koncu še XOR-odštevanje.

3.10 RC5

RC5 bločni šifrirni postopek ima besedno orientirano arhitekturo s spremenljivo dolžino besede w = 16, 32 ali 64 bitov. Značilen je izjemno kompakten opis, postopek pa je primeren tako za strojno kot tudi za programsko implementacijo. Število faz algoritma r in bajtna dolžina ključa b sta prav tako spremenljiva. Zato ga bolj točneje lahko označimo kot RC5-w, RC5-w/r in RC5-w/r/b. Zapis RC5-32/12/16 predstavlja tipično izbrane parametre, torej r = 12 faz algoritma, w = 32 bitov dolga

beseda in b = 16 bajtni ključ. Za 32 bitov dolgo besedo je priporočljiva uporaba 12 faz algoritma, medtem ko je za 64 bitov dolge besede priporočljiva uporaba 16 faz algoritma.

Poglejmo si nekoliko bolj podrobno algoritem RC5. Tako izvorna kot šifrirana sporočila so dolžine 2w. Vsaka faza algoritma izračuna novo vrednost vsake w bitov dolge polovice podatkov in pri tem uporabi 2 podključa za vhodno transformacijo ter nato še po 2 podključa v vsaki fazi. V prvem koraku torej izračunamo 2r+2 podključa K_0 … K_{2r+1} z nekoliko kasneje opisanim algoritmom.

Vhodna transformacija je dejansko vsota po modulu 2^w med prvimi w biti izvornega sporočila (besede) A in prvim podključem K_0 ter vsota po modulu 2^w med drugimi w biti izvornega sporočila B in drugim podključem K_1.

Sledi r faz enkripcije, kjer se v vsaki fazi na novo preračunata leva in desna polovica sporočila A in B in sicer na sledeč način:

A = ((A $\oplus$ B) (premik v levo za B bitov) + K_{2i} mod 2^w

B = ((B $\oplus$ A) (premik v levo za A bitov) + K_{2i+1} mod 2^w

Izhodno šifrirano sporočilo dobimo tako, da sestavimo obe polovici A in B.

Kot je torej razvidno iz opisanega postopka, sta edini operaciji, ki se uporabljata v RC5 algoritmu, vsota po modulu 2^w in rotacija. XOR operacija je linearna, medtem ko vsoto lahko smatramo kot nelinearno, odvisno kakšno metriko za linearnost si izberemo. Glavna točka nelinearnosti pa je operacija rotacije, ki krožno premika w bitne besede v levo za y bitov, kjer števol premikov y lahko zmanjšamo z modulom w.

Postopek generacije ključev razširi b bajtni ključ v 2r+2 podključev K_i, ki so vsi dolžine w bitov.

Najprej definiramo spremenljivki u = w/8 (število bajtov na sporočilo (besedo)) in c = [b/u] (število besed, ki jih lahko zapolni ključ K). Ključ na desnem koncu po potrebi zapolnimo z ničelnimi biti, da tako dosežemo število bajtov, ki je deljivo z u (npr. K[j] = 0, za b $\leq$ j $\leq$ c.u -1).

Za i med 0 in c-1: L_i $L_i = \sum_{j=0}^{u-1} 2^{8j} K[i.u + j]$, torej zapolnimo L_i od spodnjega do zgornjega bajta tako, da uporabimo vsak bajta ključa K po enkrat.

Ključu K_0 priredimo vrednost P_w iz tabele 11. Za i med 1 in 2r+1 pa izračunamo vrednost ključa po sledeči formuli: $K_i = K_{i-1} + Q_w$ (tudi vrednosti slednjega so v tabeli 11).

V zadnjem koraku RC5 enkripcije najprej inicializiramo vrednosti spremenljivk i, j, A in B na 0, t pa priredimo max (c, 2r+2). Za s med 1 in 3t pa nato izračunamo:

$K_i = K_i + A + B$ (obe seštevanji po moduli 2^w) in rotacija za 3 bite v levo

$A = K_i$, i = i +1 mod (2r + 2)

$L_j = L_j + A + B$ (obe seštevanji po moduli 2^w) in rotacija za (A + B mod 2^w) bitov v levo

$B = L_j$, j = j +1 mod c

Po koncu 3t iteracij zadnjega koraka dobljene vrednosti K_0, K_1 … K_{2r+1} predstavljajo w bitne podključe za RC5 algoritem.

w	16	32	64
$\mathbf{P_w}$	B7E1	B7E15163	B7E15162 8AED2A6B
$\mathbf{Q_w}$	9E37	9E3779B9	9E3779B9 7F4A7C15

Tabela 11 : RC5 konstante (hex oblika)

RC5 dekripcija uporablja enake podključe kot enkripcija, podobno kot enkripcija pa tudi tu vhodno sporočilo (v tem primer je to seveda šifrirano sporočilo) razdelimo na dve polovici A in B. Na slednjih nato izvajamo operaciji XOR in odštevanje po modulu 2^w in sicer:

B = $((B - K_{2i-1}) \bmod 2^w$ in rotacija v desno za A bitov) $\oplus$ A

A = $((A - K_{2i}) \bmod 2^w$ in rotacija v desno za B bitov) $\oplus$ B

Na koncu pa še:

M = $((A - K_0) \bmod 2^w,\ (B - K_1) \bmod 2^w)$.

3.11 Ostali bločni šifrirni postopki

Poleg bločnih šifrirnih postopkov, ki smo jih opisali v prejšnjih poglavjih, so kriptoanalogi razvili še številne druge algoritme, ki so bolj ali manj uspešno zagotavljali varnost v komunikacijah med entitetami. V nadaljevanju na kratko opisujemo nekaj izmed teh postopkov.

LOKI'91 je bil predlagan kot alternativa algoritmu DES z večjim 64 bitnim ključem, 64 bitno velikostjo bloka in 16 krogi enkripcije. Od DESa se najbolj razlikuje v fazi generiranja ključev in funkciji f. Funkcija f v vsakem krogu uporablja 4 identične S škatle, ki preslikajo 12 bitne vhode na 8 bitne izhode, od 12 vhodnih bitov pa s 4 izberemo eno izmed 16 funkcij, ki predstavljajo eksponencialne funkcije s fiksnim eksponentom. V algoritmu LOKI'91 sicer niso našli nobene slabosti, ki bi omogočala razbitje enkripcije, testi so celo pokazali, da je algoritem odporen tako na linearno kot na diferencialno kriptoanalizo, vendar pa algoritem ni varen za uporabo v zgoščevalnih funkcijah in drugih načinih uporabe. Izkazalo pa se je tudi, da je algoritem občutljiv na t.i. key-related napade.

KHUFU in **KHAFRE** sta bločna šifrirna postopka, ki sta precej podobna DEL algoritmu, predlagana pa sta bila kot hitra programsko naravnana alternativa DESu. Uporabljata 64 bitne bloke, 8 x 32 bitne S škatle in spremenljivo število krogov enkripcije (običajno 16, 24 ali 32). Khufu ključi so lahko dolgi tudi to 512 bitov, medtem ko je bit dolžina Khafre ključev večkratni števila 64 (običajno 64 in 128); 64 bitni so XORani na bloke sporočil pred in po vsakem od 8 krogov. Medtem ko en

krog DES enkripcije vključuje 8 S škatel, ki delajo preslikavo med 6 bitnimi vhodi in 4 bitnimi izhodi, pa en krog Khufru enkripcije vključuje le eno S škatlo, ki preslika 8 bitne vhode na 32 bitne izhode, s tem da je S škatla v vsakem izmed 8 krogov različna. S škatle se generirajo pseudo naključno s pomočjo uporabnikovega ključa. Khafre uporablja fiksne S škatle, ki se prav tako generirajo pseudo naključno iz S škatel, ki se izdelajo iz naključnih števil. Po doslej znanih napadih sta Khufu s 16 krogi in Khafre s 24 krogi bolj varna proti napadom kot DES.

Blowfish je še eden v vrsti DESu podobnih algoritmov. Deluje na blokih dolžine 64 bitov in ključi, ki so lahko dolgi tudi do 448 bitov. Računsko precej zahteven postopek razširitve ključa naredi osemnajst 32 bitnih podključev in 8 x 32 bitne S škatle, ki se izpeljejo iz vhodnega ključa.

CAST je poseben načrt procedure za družino DESu podobnih bločnih postopkov, ki vključuje m x n bitne S škatle (m < n), temelji pa na zavitih funkcijah. CAST procedure imajo spremenljivo dolžino ključa in spremenljivo število krogov enkripcije.

3-WAY je bločni šifrirni postopek z 96 bitno dolžino bloka in ključa. Načrtovan je bil za hitro delovanje tako v programski kot strojni implementaciji, odporen pa naj bi bil proti linearnim in diferencialnim napadom. V jedru ima 3 bitne nelinearne S škatle in linearno preslikavo polinomskega množenja.

SHARK je bločni šifrirni postopek tipa substitucijsko permutacijsko omrežje SP, ki ga lahko opišemo kot posplošitev algoritma SAFER. Vključuje močno nelinearne S škatle in idejo MDS kod za difuzijo, s čimer precej zmanjšamo število krogov enkripcije, ki zagotovijo zadostno stopnjo varnosti postopka. Postopka BEAR in LION sta 3 krožni neuravnoteženi Feistolovi omrežji. Vsi trije postopki še niso bili ustrezno preizkušeni.

SKIPJACK je tajen bločni šifrirni postopek, s katerim upravlja ameriška agencija za nacionalno varnost. O njem je znanega bolj malo, z izjemo tega, da uporablja 80 bitni tajen ključ.

GOST 28147-89 je enkripcijski algoritem, ki ga je uporabljala nekdanja Sovjetska zveza. Uporablja 32 krožne Feistelove strutkure, točno delovanje S škatel pa ni znano.

4. Zaključki

Kriptografija je že več stoletij izjemno pomembno področje človekovega vsakdana, v današnjih časih, ko je velika večina komunikacije že elektronska, pa je pomen kriptografskih raziskav še toliko bolj raziskav. Kot je razvidno iz te seminarske

naloge, raziskovalci vlagajo veliko truda v iskanje in implementacijo ustreznih matematičnih algoritmov, ki bi zagotavljali različne aspekte komunikacijske varnosti – zasebnost, avtentičnost, nezanikanje, celovitost. Doslej so bili predlagani že številni postopki, od katerih so se nekateri uveljavili tudi kot mednarodni standardi, drugi pa ostajajo v obliki predlogov v akademski in industrijski sferi. Vendar pa z razvojem računalniške opreme in dostopnostjo le-te prav vsakemu posamezniku postajajo vedno bolj »napredni« tudi potencialni napadalci, zato je življenska doba večine doslej implementiranih algoritmov omejena. Neprestano je potrebno iskati izboljšave postopkov in povečevati odpornost algoritmov proti različnim vrstam napadov, saj je le na ta način mogoče zagotoviti, da bo moderne komunikacije varne, to pa v zadnjem času postaja vedno večja skrb in tudi vrednota v sodobni družbi.

5. Literatura

[1] Bruce Schneier: Applied Cryptography

[2] A. J. Menzes, P. C. van Oorschot, S. A. Vanstone: Handbook of Applied Cryptography

[3] Sašo Tomažič: Varnost v telekomunikacijah in kako jo zagotoviti

[4] Sašo Tomažič: Varne komunikacije preko interneta