Sašo Tomažič

# Varnost informacijsko komunikacijskih sistemov

# Kazalo

# 1 Uvod

Živimo v informacijski družbi, v kateri se več kot polovica bruto družbenega proizvoda ustvari z ustvarjanjem, prenosom, obdelavo in prodajo informacij oziroma informacijskih storitev. Informacijsko komunikacijski sistemi (IKSi) so postali nujen del vsakodnevnega življenja. Z njimi se srečujemo, ko telefoniramo, gledamo televizijo, plačujemo z bančno kartico, dvigamo denar na bankomatu, pošljemo elektronsko pošto, brskamo po svetovnem spletu in še pri mnogih drugih vsakodnevnih opravilih. Življenja brez IKS si praktično ne znamo več predstavljati.

Uporaba informacijsko komunikacijskih tehnologij (IKT) na vseh področjih življenja je zagotovo precej olajšala življenje in dvignila njegovo kvaliteto, vendar je hkrati s tem prinesla tudi nevarnost njihove zlorabe. Z ozirom na veliko vrednost, ki jo imajo informacije v informacijski družbi, je postala kraja informacij izredno privlačna. Z vdorom v IKS nekega podjetja je mogoče priti do vseh informacij o poslovanju, tržni strategiji, novih produktih in podobno, kar lahko izkoristi konkurenca. Z vdorom v bančni je mogoče prenašati denar iz enega računa na drug račun oziroma ustvarjati nov denar, ki je v informacijski družbi zgolj informacija v bančnem informacijskem sistemu, in na ta način neupravičeno pridobiti premoženje. V vojaškem informacijskem sistemu lahko nasprotnik pridobi informacije, ki mu prinesejo taktično in strateško prednost, kar lahko odloča o zmagi ali porazu. Vdor v na osebni računalnik lahko omogoči dostop do osebnih podatkov, dostop do bančnega računa in tudi krajo identitete, kar lahko lastniku povzroči nepopravljivo škodo. Veliko škodo lahko povzroči tudi izguba določenih podatkov zaradi neprevidnega upravljanje z njimi oziroma z informacijsko komunikacijskim sistemom, na katerem se hranijo.

Zaradi velike vrednosti , ki jo imajo lahko določeni podatki, in zardi velike škode, ki jo lahko povzroči njihova izguba ali zloraba, jih je potrebno zaščititi, ne glede na to ali gre za pomembne podatke državnih ustanov in podjetij ali pa za osebne podatke uporabnikov IKSa. V nadaljevanju bomo, zaradi preprostosti ločevali zgolj med podjetji in fizičnimi osebami. Pod podjetje bomo razumeli tako podjetja kot tudi državne ustanove, društva, organizacije in podobno, medtem ko bomo s pojmom fizične osebe označevali uporabnike nekega zasebnega ali javno dostopnega IKSa za svoje osebne namene.

Klasično so bili podatki shranjeni oziroma so se prenašali večinoma v tiskani obliki na papirju. Z razvojem IKT se danes večina podatkov shranjuje in prenaša v elektronski obliki, zato se bomo v nadaljevanju posvetili le zaščiti podatkov v elektronski obliki. Za zaščito podatkov v elektronski obliki je potrebno pravilno načrtovati in tudi pravilno uporabljati IKSe v katerih se shranjujejo oziroma prenašajo podatki. V nadaljevanju bomo spoznali različne postopke, ki

omogočajo načrtovanje varnih IKSov, kakor tudi potrebne ukrepe in pravila obnašanja uporabnikov, ki omogočajo njihovo varno uporabo.

Da bi lahko načrtovali varne IKSe, moramo najprej poznati nevarnosti, ki so jim podatki v takih sistemih izpostavljeni. Te so predvsem:

- *Izguba podatkov.* Do izgube podatkov lahko pride iz različnih razlogov. Povzročijo jo lahko okvare na napravah IKSa, izpad električne energije, nenamerno ali zlonamerno brisanje podatkov, nesreče, kot so požar, poplava ali potres in podobno.

- *Nedostopnost podatkov.* Tudi podatki, ki niso izgubljeni, lahko postanejo v nekem času, ko jih potrebujemo, nedostopni. Tudi nedostopnost je lahko posledica napak na napravah IKSa. Lahko pa je tudi posledica nenamernih ali namernih motenj na komunikacijski poti.

- *Kraja ali nedovoljen vpogled v podatke.* Do kraje ali nedovoljenega vpogleda lahko pride s pomočjo prestrezanja podatkov na prenosni poti, z vdorom v IKS ali z nepooblaščenim dostopom do različnih naprav, predvsem računalnikov, IKSa, ki so nezadostno zaščiteni.

- *Neželeno spreminjanje podatkov.* Do spremembe podatkov lahko pride zaradi napak pri shranjevanju in prenosu podatkov. Spreminjanje podatkov pa je lahko tudi zlonamerno. Zlonamerne spremembe se lahko izvajajo s prestrezanjem podatkov pri prenosu ali pa z vdorom v IKS.

Vidimo, da so nevarnosti, ki so jim izpostavljeni IKSi različnega izvora, od naravnih, na katere človek nima vpliva, do tistih, ki jih predstavlja človek s svojim ravnanjem. Zadnje so lahko posledica neprevidnega ali neodgovornega ravnanja upravičenih uporabnikov sistema ali pa so to napadi na IKS, ki jih izvajajo ljudje, ker želijo pridobiti neko korist, oziroma, zgolj povzročiti določeno škodo. Zaščita proti tem je najzahtevnejša, saj stalno iščejo varnostne luknje v delovanju IKSov in razvijajo nove načine za njihovo izkoriščanje.

Zavedati se moramo, da popolnoma varnega IKSa ni. Govorimo lahko le o zmanjševanju nevarnosti. Ne glede na to, kako dobro ščitimo IKS, vedno obstaja določena verjetnost, da se bodo zaradi tehničnih težav podatki izgubili ali pokvarili, ali pa da bo nekemu napadalcu uspelo vdreti v sistem. Naloga pri zagotavljanju varnosti IKS je čim bolj zmanjšati to verjetnost.

Različni varnostni mehanizmi, ki jih vpeljujemo v IKS po eni sicer povečujejo varnost sistema, vendar lahko na drugi strani zmanjšujejo njegovo uporabnost, ker opravičenim uporabnikom otežkočajo dostop do njega in njegovo uporabo. Zagotavljanje varnosti torej predstavlja nek kompromis med varnostjo in uporabnostjo. Najbolj varen bi bil zagotovo sistem, do katerega nima nihče dostopa in ni povezan v nobeno omrežje, vendar bi bil tak sistem tudi popolnoma neuporaben.

Zavedati se moramo tudi, da je varovanje IKS povezano z določenimi stroški. Varovanje, pri katerem bi stroški presegali vrednost informacij, ne bi bilo racionalno. Po drugi strani pa lahko štejemo IKS za dovolj varen pred napadi, če stroški napada bistveno presegajo vrednost, ki bi jo napadalec z uspešnim napadom pridobil.

V tekstu doslej smo večkrat uporabili pojma informacija in podatek. Po eni strani govorimo o informacijskem sistemu po drugi strani pa o zaščiti podatkov. V praksi se ta dva pojma pogosto uporabljata izmenljivo, kot sinonima, čeprav v resnici nimata čisto enakega pomena. Informacija je veličina, ki pove koliko novega, nepoznanega ali nepričakovanega smo izvedeli, ko smo dobili nek podatek. Informacija je torej vsebovana v podatku, ali povedano drugače, podatek je na nek način predstavljena oziroma zapisana informacija. Koliko informacije vsebuje nek podatek je odvisno od načina zapisa tega podatka in tudi od tega, kaj prejemnik podatka že prej ve.

Čeprav smo privzeli splošno uporabljeno ime IKS (informacijsko komunikacijski sistem) in ne podatkovno komunikacijski sistem, imamo v IKSih opravka predvsem s podatki in nas najpogosteje ne zanima koliko informacije ti podatki vsebujejo. Informacija nas zanima predvsem z vidika vrednosti podatkov, saj jim vrednost daje zgolj informacija, ki jo vsebujejo. S tehničnega vidika izvedbe IKS je informacijska vsebina podatkov nebistvena, s stališča njegovega načrtovanja in zaščite podatkov pa je bistvena, saj le na tej osnovi lahko ločujemo med pomembnimi in manj pomembnimi, javnimi, privatnimi in tajnimi ter splošno dostopnimi in internimi podatki.

## 1.1 Dokumenti in celovitost podatkov

Dokumenti so podatki, ki imajo poseben status, saj imajo določeno pravno formalno veljavo, ki je odvisna od vrste dokumenta. Primeri dokumentov so razne pogodbe, računi, potrdila in podobno. Da bi imeli lahko določeni podatki status dokumenta, morajo imeti določene lastnosti, ki jih obravnavamo pod skupnim pojmom *celovitost podatkov* (ang. data integrity). Celovitost podatkov ima več vidikov, od vrste dokumenta pa je odvisno ali mora ustrezati vsem ali pa zgolj nekaterim od teh vidikov. Varovanje celovitosti dokumentov je eden od najpomembnejših vidikov varnosti IKSov. Vidiki celovitosti podatkov so:

- *Verodostojnost*. Dokument je verodostojen, če imamo zagotovilo, da od takrat, ko je bil potrjen oziroma sprejet, ni bil spremenjen. Podatki, ki jih je mogoče spreminjati, ne da bi bilo to kasneje možno ugotoviti, ne morejo imeti statusa dokumenta.

- *Avtentičnost*. Dokument je avtentičen, če lahko z gotovostjo trdimo, kdo je njegov avtor. Avtentičnost lahko obravnavam ločeno ali pa tudi v sklopu verodostojnosti. Če je namreč avtorstvo dokumenta del njegove vsebine, vsebina pa je verodostojna, potem je zagotovljena tudi njegova avtentičnost.

- *Neovrgljivost*. Dokument je neovrgljiv, če njegov avtor ne more zanikati avtorstva oziroma če njegov podpisnik ne more zanikati, da ga je podpisal. Avtentičnost sama po sebi še ne zagotavlja neovrgljivosti, saj lahko avtor dokumenta zanika njegovo avtorstvo, če tega ne moremo dokazati.

- *Časovna opredeljenost*. Dokument je časovno, če lahko z gotovostjo ugotovimo, kdaj je

bila zagotovljena njegova verodostojnost in avtentičnost, kar onemogoča, da bi dokumente sprejemali za nazaj.

- *Tajnost.* Določeni dokumenti so lahko zaupni ali zasebni, kar pomeni, da mora biti njihova vsebina tajna, dostopna samo določenim osebam, ostalim pa mora biti skrita. Tajnost ni nujna lastnost vsakega dokumenta, saj so lahko mnogi dokumenti tudi javno dostopni.

- *Trajnost.* Določeni dokumenti so trajnega pomena, zato je potrebno poskrbeti, da ne pride do njihovega uničenja ali izgube za ves čas njihove veljavnosti oziroma dokler jih potrebujemo. Trajnost ni nujen pogoj, da bi imeli določeni podatki status dokumenta, vendar pa je potrebna, če želimo, da dokument služi svojemu namenu. Izgubljen ali uničen dokument namreč ne more služiti svojemu namenu.

Tradicionalno so dokumenti tiskani na papirju. Njihova verodostojnost, avtentičnost in neovrgljivost se v tem primeru zagotavlja z lastnoročnimi podpisi. Za časovno opredeljenost običajno skrbi notarska služba, ki na dokument vpiše tudi čas notarske overovitve, poleg tega pa lahko hrani tudi kopijo dokumenta.

V IKSih so dokumenti shranjeni v elektronski obliki. Brez posebnih mehanizmov, ki bi to preprečevali, je elektronske dokumente zelo preprosto neopazno spreminjati. Za zagotavljanje verodostojnosti, avtentičnosti, neovrgljivosti, tajnost in časovne opredeljenosti potrebujemo pri elektronskih dokumentih različne postopke, ki so zasnovani na šifriranju podatkov. Različnim postopkom šifriranja podatkov in njihove uporabe za zagotavljanje posameznih vidikov celovitosti dokumentov, je zato posvečenih več poglavij v knjigi.

## 1.2 Varnostna politika IKSa

Pod pojmom varnostna politika IKSa razumemo sveženj postopkov in ukrepov, ki jih neko podjetje formalno uvede za zaščito svojega IKSa. Ti ukrepi se nanašajo tako na infrastrukturo, programsko opremo kot tudi na upravljavce in uporabnike IKSa. Varnostno politiko je potrebno sproti posodabljati, v skladu z novimi informacijskimi tehnologijami in tudi z vedno novimi nevarnostmi.

Varnostna politika je obenem tudi dokument, ki bi ga moralo sprejeti vsako podjetje, katerega poslovanje je v veliki meri odvisno od IKSa. Zelo pomembno je, da varnostno politiko sprejme in tudi nadzira njeno izpolnjevanje najvišje vodstvo podjetja. Vsi zaposleni, ki imajo dostop do IKSa, bi morali poznati tiste dele tega dokumenta, ki se nanašajo nanje in se tudi zavezati k izpolnjevanju v dokumentu podanih zahtev. Opredeljene morajo biti tudi sankcije za kršenje varnostne politike.

Neformalno bi morale tudi fizične osebe sprejeti svojo varnostno politiko, ki bi se je morale držati, če bi želele zaščititi svoje podatke. Ukrepi in postopki, ki bi jih morale za svojo uporabo sprejeti fizične osebe so zelo podobni tistim, ki jih morajo sprejeti podjetja, vendar so ti lahko manj rigorozni, saj je vrednost podatkov, ki jih želijo ščititi ponavadi bistveno manjša kot pri pravnih osebah.

Z uvajanjem varnostne politike želimo zaščititi podatke v IKSu pred izgubo oziroma uničenjem, pred razkritjem nepooblaščenim osebam oziroma krajo, pred nenamenskimi in neavtoriziranimi spremembami. Obenem pa želimo zagotoviti njihovo razpoložljivost, to je zagotoviti dostop do podatkov pooblaščenim uporabnikom, ko jih ti potrebujejo.

Varnost IKSa zelo natančno opredeljuje standard ISO/IEC 27002, ki ga je izdala mednarodna organizacija za standardizacijo (ISO – International Organization for Standardization) v sodelovanju z Mednarodno elektrotehniško komisijo (IEC - International Electrotechnical Commission). Nekoliko novejši in zelo uporaben pa je tudi Standard dobre prakse za informacijsko varnost (The Standard of Good Practice for Information Security), ki ga je izdal Forum za informacijsko varnost (Information Security Forum). To je celovit praktičen vodič za oceno tveganja in upravljanje z informacijsko varnostjo v podjetjih. Ta standard je skladen s standardom ISO/IEC 27002.

V obeh dokumentih so zelo natančno opredeljeni elementi varnosti IKSa in priporočeni postopki za njeno zagotavljanje. Za uvajanje varnostne politike v podjetju sta oba dokumenta nepogrešljiva pripomočka. Varnostna politika je v večjih sistemih organizirana več nivojsko. V krovnem dokumentu so podana splošna načela varnostne politike, podrejeni dokumenti pa potem pokrivajo posamezna področja in natančna navodila za izvajanje varnostne politike. V nadaljevanju naštejmo le osnovne elemente, k jih mora opredeljevati varnostna politika.

- Opredelitev ekipe oziroma organizacijske enote, ki skrbi za izvajanje varnostne politike, njihovih pristojnosti in odgovornosti.
- Opredelitev postopka za spremembo varnostne politike.
- Opredelitev odgovornosti uporabnikov in sankcij za kršenje posameznih načel varnostne politike.
- Lista vseh elementov IKSa, ki so potencialno ogroženi, kar vključuje strojno opremo, programsko opremo prostore in človeške vire.
- Varovanje elementov IKSa pred nesrečami in naravnimi katastrofami (požari, potresi, poplave)in postopki za varovanje podatkov pred izgubo (arhiviranje, varnostne kopije, ...).
- Klasifikacija podatkov oziroma informacijskih virov v IKSu z ozirom na njihovo vrednost, stopnjo zaupanja in škodo, ki jo lahko povzroči njihova izguba ali kraja. Za vsak informacijski vir je potrebno definirati lastnika oziroma lastnike podatkov in določiti skrbnika.
- Metodologija ocene tveganja, ki vsebuje analizo šibkosti, potencialnih nevarnosti in izgub, ki jih lahko te povzročile. Za vsako nevarnost je potrebno oceniti tudi njeno verjetnost.
- Seznam ukrepov za zaščito posameznih naprav in podatkov v IKSu in tudi njihovo ceno. Ukrepi, ki po ceni presegajo škodo, ki bi lahko nastala, če jih ne bi sprejeli, namreč niso ekonomični.

- Pravice in kontrolo dostopa skrbnikov in uporabnikov. To obsega tako fizični dostop do posameznih naprav v IKSu kot tudi dostop do podatkov v različnih informacijskih virih. Nanaša se tako
- Pravila za varno in odgovorno ravnaje uporabnikov IKSa in načini obveščanja, ozaveščanja in izobraževanja uporabnikov.
- Pravila in postopke za uvajanje novih elementov IKSa, kar se nanaša tako na strojno kot tudi na programsko opremo. Opredeljen mora biti nabor dovoljene strojne in programske opreme in narejen popis obstoječe strojne in programske opreme.
- Pravila dostopa uporabnikov iz IKSa v javno omrežje in iz javnega omrežja v IKS in postopki zaščite omrežja.
- Postopki za zaščito pred zlonamerno programsko opremo (virusi, trojanski konji, črvi, ..)
- Pravila za identifikacijo in avtorizacijo uporabnikov IKSa.
- Pravila in varnostne kriterije, ki jih je potrebno upoštevati pri razvoju nove programske opreme.
- Postopki in ukrepi ob okvarah, nesrečah, izgubi podatkov in vdorih v IKS.
- Postopki za preverjanje izvajanje varnostne politike in postopki za preverjanje uspešnosti ukrepov varnosti IKSa.

Celovita varnostna politika torej zajema celo vrsto ukrepov z različnih področij, od organizacije dela, izobraževanja, prava in varovanja objektov do področij informatike inn telekomunikacij. V varnostni politiki morajo biti za vsako od teh natančno določeni postopki, ki so potrebni za njeno uveljavljanje v praksi.

Pri načrtovanju varnostne politike se moramo zavedati, da popolna varnost ne obstaja, varnostni ukrepi, ki jih uvedemo lahko samo zmanjšujejo verjetnost, da bi prišlo do škode. Določeni varnostni ukrepi lahko tudi otežkočajo dostop do podatkov upravičenim uporabnikom in s tem zmanjšujejo uporabnost IKSa. Varnostna politika tako predstavlja nekakšen kompromis med varnostjo, uporabnostjo in ceno, vendar mora biti ta kompromis biti zasnovan na čim bolj natančni oceni tveganja.

V nadaljevanju se bomo posvetili predvsem osnovnim principom zagotavljanja varnosti, ki sodijo na področji informatike in telekomunikacij. Obravnava ostalih področij presega okvire te knjige.

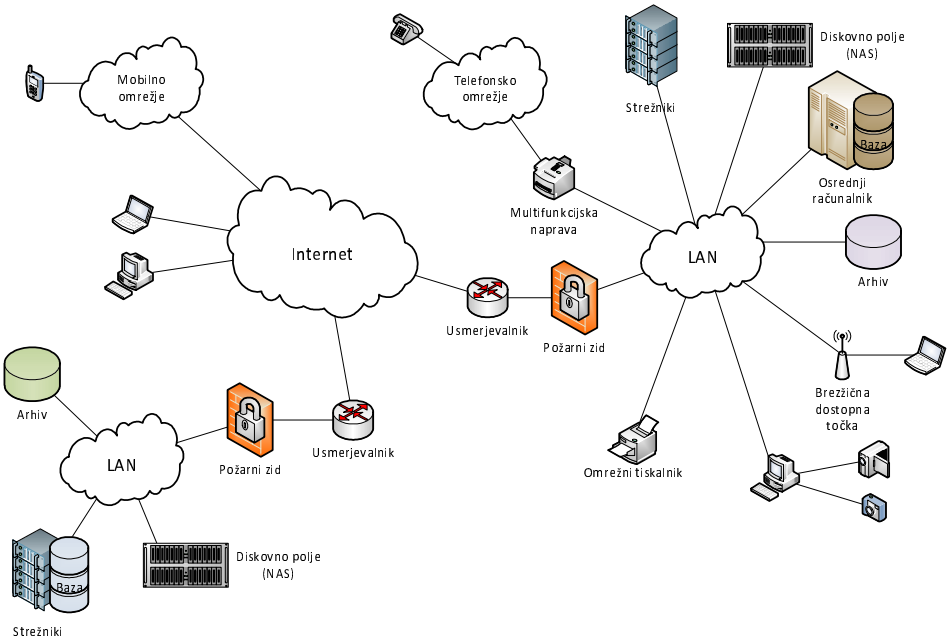
Najprej so v drugem poglavju na kratko predstavljeni osnovni gradniki IKSa, predvsem strojna in programska oprema, ki je potrebna za njegovo delovanje. Ker potrebujemo pri mnogih vidikih zagotavljanja celovitosti podatkov dobre postopke šifriranja, je naslednje poglavje posvečeno šifriranju.

# 2 Informacijsko komunikacijski sistem IKS

Informacijsko komunikacijski sistem (IKS) imenujemo skup strojne in programske opreme, ki omogoča učinkovit način za shranjevanje, urejanje, upravljanje, iskanje, prenos in prikaz podatkov. V širšem pomenu vključuje IKS tudi podatke, ki se hranijo v njem, njegove upravljavce in njegove uporabnike.

Osnovni namen uporabe informacijskih sistemov je pregleden, preprost in učinkovit dostop do podatkov, ki so ključnega pomena za dobro in učinkovito poslovanje in odločanje.

Preprost primer infrastrukture IKSa, v katerem sta preko javnega omrežja Internet povezani dve lokalni omrežji LAN (ang. local area network) je prikazan na sliki 2.1. Povezava v javno omrežje predstavlja eno največjih varnostnih tveganj IKSa, zato je zaščiti pred vdori za javnega omrežja potrebno posvetiti izredno veliko pozornosti.




Slika 2.1 – Preprost primer infrastrukture IKSa. Dve lokalni omrežji (LAN - Local Area Network) sta povezani preko javnega omrežja Internet. V vsakem od teh omrežji je med seboj povezana različna strojna in programska oprema. Do IKSa je mogoče dostopati preko delovnih postaj, ki so vključene v lokalno omrežje, kakor tudi iz računalnikov, ki so vključeni v javno omrežje Internet, in mobilnih terminalov, ki so vključeni v mobilno telefonsko omrežje.

## 2.1 Strojna oprema

Pod pojmom strojna oprema IKSa razumemo različne elektronske naprave, ki so namenjene shranjevanju, obdelavi, prenosu, pregledovanju in zaščiti podatkov. Sem štejemo računalnike, naprave za shranjevanje podatkov, obrobne naprave, komunikacijske naprave in različne podporne naprave. Večino funkcij, ki so potrebne za delovanje IKSa lahko opravljajo splošno namenski računalniki z ustrezno programsko opremo, posameznim funkcijam pa so lahko namenjene specializirane naprave.

### 2.1.1 Računalniška oprema

Splošno namenski računalniki lahko opravljajo različne funkcije v okviru IKSa. Predvsem nastopajo računalniki v funkciji različnih strežnikov in delovnih postaj. Poleg tega lahko opravljajo računalniki tudi različne komunikacijske in varnostne funkcije, čeprav se v večjih IKSih v ta namen običajno uporabljajo specializirane naprave.

#### Strežniški računalniki

V IKS je običajno vključenih večje število različnih strežnikov. To so lahko datotečni, podatkovni, poštni, domenski, spletni in drugi strežniki. V splošnem je strežnik vsaka naprava ali proces, ki omogoča dostop do svojih virov enemu ali več odjemalcem.

Katero od navedenih strežniških funkcij opravlja določen računalnik, je odvisno od programske opreme. Posamezen računalnik lahko opravlja tudi funkcije več različnih strežnikov ali pa so funkcije enega strežnika porazdeljene med več računalnikov.

Čeprav lahko funkcije strežnika opravlja skoraj vsak osebni računalnik se za strežnike v IKSih običajno uporabljajo zelo zmogljivi in zanesljivi računalniki, načrtovani posebej za ta namen. Strežniški računalniki morajo zagotavljati zadostne vire za njihov namen, podpirati morajo hkraten dostop večjemu številu odjemalcev in delovati zelo zanesljivo, saj so običajno v pogonu 24 ur na dan 7 dni v tednu. Poleg tega morajo omogočati tudi nadzor in upravljanje na daljavo.

Zaradi navedenih zahtev imajo namenski strežniki običajno enega ali več zmogljivih, večjedernih procesorjev, veliko delovnega pomnilnika, hitre in zanesljive diske, redundantne sisteme, kot je to na primer dvojno napajanje, več zmogljivih omrežnih vmesnikov, možnost nadzora (temperatura, delovanje procesorja, stanje diskov in pomnilnika, ..) in upravljanja (vklop, izklop, nastavitve, zagon programov, ...) na daljavo, tudi ko se strežniški program na računalniku zaustavi oziroma, ko je računalnik izklopljen.

Strežniški računalniki morajo biti za zanesljivo delovanje postavljeni v ustreznem delovnem okolju, ki zagotavlja primerno temperaturo in vlago. Za zanesljivo delovanje in razpoložljivost morajo imeti na voljo brezprekinitveno napajanje, ki zagotavlja avtonomijo delovanja vsaj za čas regularne zaustavitve sistema, da zaradi prekinitve v napajanju ne bi prihajalo do izgube podatkov ali do okvar na računalniku.

Strežniški računalniki so na voljo v različnih izvedbah, ki jih lahko uvrstimo v tri osnovne kategorije:

- *Samostoječi strežniki* so po obliki zelo podobni osebnim računalnikom. Navadno so to manj zmogljivi strežniki, ki so namenjeni manj zahtevnim aplikacijam ali manjšemu številu hkratnih uporabnikov.
- *Samostojni vgradni strežniki* so namenjeni vgradnji v strežniške omare. Tako kot samostoječi strežniki so tudi vgradni strežniki samostojni in samozadostni sistemi. To pomeni, da vsebujejo vse strojne komponente, ki jih potrebujejo za svoje delovanje.
- *Strežniške rezine* so prilagojene modularni vgradnji v posebno ohišje, ki vsem rezinam zagotavlja skupno napajanje, hlajenje, povezljivost in nadzorne funkcije. Podatki se navadno hranijo izven rezine.

#### Terminali in delovne postaje

Terminali so naprave, ki omogočajo vnos in prikaz podatkov. Na osrednji računalnik je lahko povezanih več terminalov hkrati. Terminali so lahko namenske naprave z zaslonom, tipkovnico in omrežnim vmesnikom, pogosto pa se kot terminale uporablja osebne računalnike ali mobilne naprava, kot so to tablični računalniki ali pametni telefoni.

Delovne postaje so samostojni polno funkcionalni računalniki, podobno kot samostoječi strežniki, le da so običajno manj zmogljive in tudi manj robustne in zato tudi precej cenejše. Kot delovne postaje se danes najpogosteje uporabljajo osebni računalniki. Delovne postaje so v IKS povezane preko omrežnih vmesnikov. Delovne postaje imajo tudi več vmesnikov za obrobne naprave. Za razliko od strežnikov, mora imeti vsaka delovna postaja obrobne, ki omogočajo vnos in prikaz podatkov. To so predvsem tipkovnica, miška in zaslon.

V manjših IKS se lahko delovne postaje uporabljajo tudi kot strežniki. V osebnem IKSu oziroma v manjšem IKSu gospodinjstva, lahko ena sama delovna postaja z obrobnimi napravami pokriva vse funkcije IKSa.

### 2.1.2 Obrobne naprave

Obrobne naprave so naprave, ki razširjajo funkcionalnost računalnikov a niso njihov sestavni del. Priključeno so lahko neposredno na računalnik ali pa v omrežje IKSa. Omogočajo lahko vnos, prikaz, tiskanje, pošiljanje in shranjevanje podatkov. Priključene so lahko neposredno na računalnike ali pa v omrežje IKSa. Primeri obrobnih naprav so zasloni, tipkovnice, miške, tiskalniki, skenerji, mikrofoni, kamere in podobno.

### 2.1.3 Naprave za shranjevanje

Shranjevanje podatkov je ena od ključnih nalog IKSov. Naprave za shranjevanje in nosilci podatkov so zato njihov nepogrešljiv del. Podatki se lahko v IKSih hranijo začasno, stalno ali

trajno. Za vsako od teh vrst shranjevanja se uporabljajo različne naprave. Pri izbiri naprav za posamezno od naštetih vrst shranjevanja so pomembne njihove lastnosti, oziroma lastnosti nosilcev, na katerih so podatki shranjeni.

- *Stalnost zapisa* (ang. volatility). Ločimo med nestanovitnimi (ang. volatile) nosilci z začasnim zapisom, ki potrebujejo za ohranjanje podatkov stalno napajanje (delovni pomnilnik računalnika) in stanovitnimi (ang. non-volatile) nosilci s stalnim zapisom, ki ohranijo podatke tudi, ko nimajo napajanja (magnetni disk, magnetni trak, diski brez gibljivih delov (SSD – Solid State Drive), optični disk).

- *Spremenljivost zapisa*(ang . mutability). Nekateri nosilci omogočajo večkratno zapisovanje oziroma spreminjanje zapisanih podatkov (delovni pomnilnik, magnetni disk, magnetni trak, SSD, ponovno zapisljivi optični disk), druge pa omogočajo zgolj enkraten zapis, ki ga ni mogoče spreminjati (predvsem optični disk za enkratni zapis).

- *Dostop do podatkov* (ang. accessibility) je lahko neposreden (delovni pomnilnik, disk brez gibljivih delov) ali zaporeden (magnetni trak). Pri prvem je čas, ki je potreben za dostop do kateregakoli podatka enak ne glede na njegovo lokacijo na nosilcu, pri drugem pa se do podatkov dostopa zaporedoma in je ta čas odvisen od lokacije podatka na nosilcu. Med tema dvema skrajnostima so nosilci, ki imajo podatke le deloma zapisane zaporedno (magnetni diski, optični diski). Pri teh se čas dostopa do podatkov sicer odvisen od njihove lokacije na nosilcu, vendar bistveno manj, kot pri nosilcih z zaporednim zapisom.

Poleg zgoraj omenjenih lastnosti so pomembne lastnosti še, hitrost zapisovanja, hitrost dostopa do podatkov, kapaciteta, gostota zapisa, cena na enoto količine podatkov in trajnost stalnega zapisa. Trajnost je pomembna predvsem pri podatkih, ki jih želimo hraniti za daljše obdobje, kot so to podatki v arhivih.

### 2.1.4 Komunikacijske naprave

V IKSih so strežniki, delovne postaje, naprave za shranjevanje in nekatere obrobne naprave med seboj povezane v komunikacijskem omrežju,kar omogoča, da si med seboj izmenjujejo podatke. Uporabnikom in upravljavcem IKSov to omogoča, da lahko s svojih delovnih postaj ali terminalov dostopajo do podatkov na različnih strežnikih, tudi iz oddaljenih lokacij.

Da bi lahko naprave povezali v omrežje morajo imeti ustrezne omrežne vmesnike. Najpogosteje so to vmesniki za omrežje ethernet ali brezžično lokalno omrežje.

Na voljo morajo biti tudi ustrezne prenosne poti, preko katerih se v omrežju prenašajo podatki. Kot prenosne poti se uporabljajo predvsem kovinski vodniki v obliki paric in koaksialnih kablov in optična vlakna v optičnih kablih.

Za prenos podatkov po omrežju so potrebne različne omrežne komunikacijske naprave.

- *Obnavljalnik* (ang. repeater) obnavlja signale, ki lahko oslabijo in se popačijo na prenosni poti.

- *Stičišče* (ang. hub) neposredno povezuje posamezne segmente prenosnih poti lokalnega omrežja.

- *Most* (ang. bridge) povezuje dva segmente lokalnega omrežja, vendar iz enega segmenta v drugega prepušča le podatke, ki so tja namenjeni.

- *Stikalo* (ang. switch) deluje podobno kot most, le da lahko povečuje več segmentov in ločuje tudi promet, ki je namenjen posameznim napravam znotraj enega omrežja.

- *Usmerjevalnik* (ang. router) usmerja promet preko več med seboj povezanih omrežij od njegovega izvora do ponora.

- *Prehod* (ang. gateway) omogoča prenos podatkov med omrežji, ki delujejo po različnih protokolih.

- *Dostopovna točka* (ang. access point) omogoča brezžični dostop do lokalnega omrežja.

- *Modem* omogoča oddaljeno povezavo v omrežje ali povezavo dveh oddaljenih omrežij preko najete ali zakupljene prenosne poti.

- *Posrednik* (ang. proxy) je nameščen na robu omrežja in v imenu naprav v omrežju posreduje zahteve v javno omrežje.

- *Požarni zid* (ang. firewall) , ki filtrira promet med javnim omrežjem in omrežjem IKSa, tako da prepušča le promet, ki je v skladu s politiko IKSa.

- *Prevajalnik omrežnih naslovov* (NAT – Network Address Translator) prevaja interne naslove naprav v omrežju v naslove javnega omrežja.

Omenjene naprave so lahko namenske ali pa lahko njihove funkcije opravljajo računalniki z ustrezno programsko opremo in omrežnimi vmesniki. Obstajajo tudi hibridne namenske naprave, ki opravljajo več omenjenih funkcij. V dostopovno točko so na primer lahko vključeni tudi usmerjevalnik, NAT in požarna pregrada.

### 2.1.5 Podporni sistemi

IKS ne bi mogel zanesljivo delovati brez dodatnih podpornih sistemov. Prostori v katerih je strojna oprema morajo biti ustrezno klimatizirani. Strežniki se napajajo preko posebnih sistemov za brezprekinitveno napajanje in so običajno nameščeni v posebnih strežniških omarah. Dostop do strojne opreme je pogosto nadzorovan, tako z elektronskimi ključavnicami kot tudi z video kamerami.

## 2.2 Programska oprema

Računalniki brez programske opreme ne delujejo. Poleg samih računalnikov pa sloni tudi delovanje večine namenskih naprav IKSa na vgrajenih procesorjih, tako da je tudi za njihovo delovanje potrebna ustrezna programska oprema.

Programsko opremo lahko razdelimo v več razredov.

- *Osnovna programska oprema računalniške naprave* (ang. firmware) zagotavlja delovanje naprave z vgrajenim procesorjem. To programsko opremo zagotavlja proizvajalec opreme in je nameščena v napravi že ob nakupu. Večino naprav ima tudi možnost posodabljanja osnovne programske opreme, ko proizvajalec izda novo, izboljšano različico.

- *Gonilnik* (ang. driver) je program, ki omogočajo drugim programom dostop do neke naprave, ki je vključena v ali priključena na računalnik, tako da tem ni potrebno natančno poznati njenega delovanja (gonilnik za disk, gonilnik za miško, gonilnik za zaslon, gonilnik za tiskalnik, ...). Gonilnike za posamezne naprave ponujajo proizvajalci teh naprav. Poleg gonilnikov za posamezne naprave določenih proizvajalcev obstajajo tudi generični gonilniki, ki so namenjeni napravam s standardnim vmesnikom.

- *Operacijski sistem* zagotavlja osnovno delovanje računalnika. Operacijski sistem se običajno zažene ob vklopu računalnika. Operacijski sistem omogoča zagon drugih programov in dostop tem programom do gonilnikov in s tem posredno do strojne opreme. Do gonilnikov in različnih storitev, ki jih nudi operacijski sistem, lahko programi dostopajo preko tako imenovanega aplikacijskega programskega vmesnika (API – Application Program Interface). Datotečni sistem, ki omogoča urejeno shranjevanje in pregledovanje podatkov na disku, je del operacijskega sistema. Operacijski sistem ni nujno od istega proizvajalca kot računalnik na katerem teče. Pogosto uporabljana operacijska sistema sta Microsoft Windows in Linux.

- *Strežnik* je program, ki teče na računalniku kot storitev (ang. service) v okviru operacijskega sistema. Strežnik nudi enemu ali več odjemalcev določeno storitve, odvisno od vrste strežnika. Omenimo nekaj strežnikov, ki pogosto nastopajo v IKSih.

  - *Datotečni strežnik* (ang. file server) nudi računalnikom, ki so nanj povezani, dostop do svojega datotečnega sistema, tako da so lahko datoteke uporabnikov shranjene centralno in dostopne z različnih računalnikov.
  - *Spletni strežnik* (ang. web server) je strežnik, ki omogoča spletnih odjemalcem dostop do spletnih strani, ki so shranjene na njem ali nekem datotečnem strežniku. Spletne strani so lahko tudi dinamične, tako da se izdelajo kot odziv na poizvedbo odjemalca, potrebne podatke pa pridobijo iz podatkovnega strežnika.
  - *Podtakovni strežnik* (ang. data server) nudi odjemalcem dostop do ene ali več baz podatkov (ang. data base). Odjemalci lahko delajo poizvedbe preko posebnega jezika za poizvedbe, najpogosteje je to SQL (ang. structured query language).

  - *Domenski strežnik* (DNS – Domain Name Server) je nekakšen imenik domenskih imen v omrežju Internet. Ob poizvedbi pretvori domenski naslov v IP (Internet Protokol) naslov omrežja Internet. Naslov www.fe.uni-lj.si na primer pretvori v IP naslov 212.235.187.85.
  - *Poštni strežnik* (ang. mail server) omogoča odjemalcem dostop do svojih poštnih predalov, sprejem, pošiljanje in posredovanje elektronske pošte.

  Poleg omenjenih strežnikov obstaja še cela vrsta drugih strežnikov, kot so strežniki za različne igre, aplikacijski strežniki, strežniki za pretočni video, iskalniki in podobno.

- *Razvojna orodja* so programi, ki omogočajo razvoj programske opreme. Običajno so integrirana v razvojnih okoljih, ki vključujejo urejevalnik besedil, prevajalnik, razhroščevalnik in druga orodja potrebna pri razvoju programske opreme.

- *Pripomočki* so programi, ki omogočajo upravljanje, nadzor in vzdrževanje sistemske programske opreme.

- *Aplikacije* so programi, ki so namenjeni neposredno končnim uporabnikom in jim omogočajo dostop do različnih storitev IKSa. Aplikacije so lahko samostojni programi, ki so nameščeni na delovnih postajah uporabnikov ali pa so to spletne aplikacije, ki tečejo na spletnih strežnikih, uporabniki pa do njih dostopajo preko spletnih brskalnikov.

# 3 Šifriranje

Šifriranje je postopek, ki razumljive podatke pretvori v obliko, ki je v splošnem nerazumljiva. Da bi podatke pretvorili nazaj v razumljivo obliko, moramo imeti nek tajni ključ, ki "odklepa" s šifrirnim postopkom "zaklenjene" podatke. Šifriranje je bilo prvotno namenjeno zgolj varovanju tajnosti podatkov pri njihovem prenosu in shranjevanju, kot smo omenili že v uvodu, pa lahko z različnimi šifrirnimi postopki zagotavljamo tudi druge vidike celovitosti podatkov.

Čeprav lahko šifriramo poljubne podatke, se šifriranje najpogosteje uporablja za tekstovna sporočila in dokumente, zato se je v šifriranju uveljavila terminologija, ki se nanaša na tekstovne dokumente in jo bomo uporabljali tudi v nadaljevanju. Nešifrirane podatke bomo tako imenovali *čistopis* (ang. plain text), šifrirane podatke pa *šifropis* (ang. cipher text). Postopek s katerim pretvorimo čistopis v šifropis imenujemo *šifriranje* (ang. encryption), obraten postopek, ki pretvori šifropis nazaj v čistopis pa *dešifriranje* (ang. decryption). Šifrirni postopek, ali krajše šifra, združuje postopka šifriranja in dešifriranja.

Prva znana uporaba šifriranja sega v stari Rim. Julij Cezar je šifriral sporočila, ki so jih prenašali njegovi sli preko zavojevanih ali vojnih ozemelj, da jih sovražniki ne bi mogli razumeti, tudi če bi jih zajeli. Šifrira, ki jo je uporabljal Cezar, je bila izredno preprosta. V tekstu sporočila je vse črke zamenjal s črkami, ki nastopajo v abecedi tri mesta za njimi. V slovenskem jeziku, bi črko a zamenjali s črko č, črko b s črko d in tako naprej, vse do črke v, ki bi jo zamenjali s črko a in črke ž, ki bi jo zamenjali s črko c. Če na ta način šifriramo čistopis "Julij Cezar", dobimo šifropis "Mžolm Fhbčt". Za dešifriranje potem uporabimo obraten postopek. Črke šifropisa zamenjamo s črkami, ki so v abecedi tri mesta pred njimi.

Opisana šifra je izredno preprosta in popolnoma neprimeren za današnjo uporabo. Kljub njeni preprostosti lahko v njej prepoznamo osnovni princip šifriranja. Tajnost sporočila pri tej šifri temelji na tajnosti samega postopka. Kdor ne pozna postopka, naj bi ne mogel prebrati sporočila.

Celoten postopek lahko razbijemo na dva dela, na samo šifro in na šifrirni ključ. V opisanem primeru je šifriranje ciklični zamik abecede za $\mathbf{K}$ znakov v levo, dešifriranje pa zamik abecede za $\mathbf{K}$ znakov v desno, ključ pa je enak tri ($\mathbf{K} = 3$). Šifra je potem lahko javna, to je vsem poznana, medtem, ko mora ostati ključ $\mathbf{K}$ skrit in ga smeta poznati samo pošiljatelj in prejemnik sporočila.

Danes je splošno uveljavljeno pravilo, da tajnost šifriranega sporočila ne sme biti osnovana na tajnosti šifre, temveč mora v celoti temeljiti na tajnosti ključa. To je še posebej pomembno, kadar določeno šifro uporablja večje število uporabnikov. Kadar neko šifro, čeprav tajno, pozna veliko število upravičenih uporabnikov, slej ko prej postane poznana tudi tistim, ki naj bi je ne poznali. Šifre, ki so temeljile na tajnosti postopka, so se v preteklosti vedno izkazale za slabe. Tak primer je tudi šifra v GSM mobilni telefoniji.

Z ozirom na vrsto ključev, ki se uporabljajo pri šifriranju delimo šifre na

- *simetrične šifre* pri katerih je isti ključ uporabljen za šifriranje in dešifriranje in
- *asimetrične šifre* pri katerih se ključ za dešifriranje razlikuje od ključa za šifriranje.

Poleg zgornje delitve pa delimo šifre še na

- *pretočne šifre* pri katerih šifriramo in dešifriramo sporočilo sproti, znak po znak, kakor ga dobivamo in
- *blokovne šifre* pri katerih sporočilo pred šifriranjem razbijemo na bloke določene dolžine in šifriramo posamezne bloke v celoti.

Več o različnih šifrah in njihovi uporabi bo govora v nadaljevanju.

## 3.1 Varnost šifer

Varnost šifre imenujemo njeno odpornost proti napadu na nanjo. Varnost je ena od najpomembnejših lastnosti šifer.

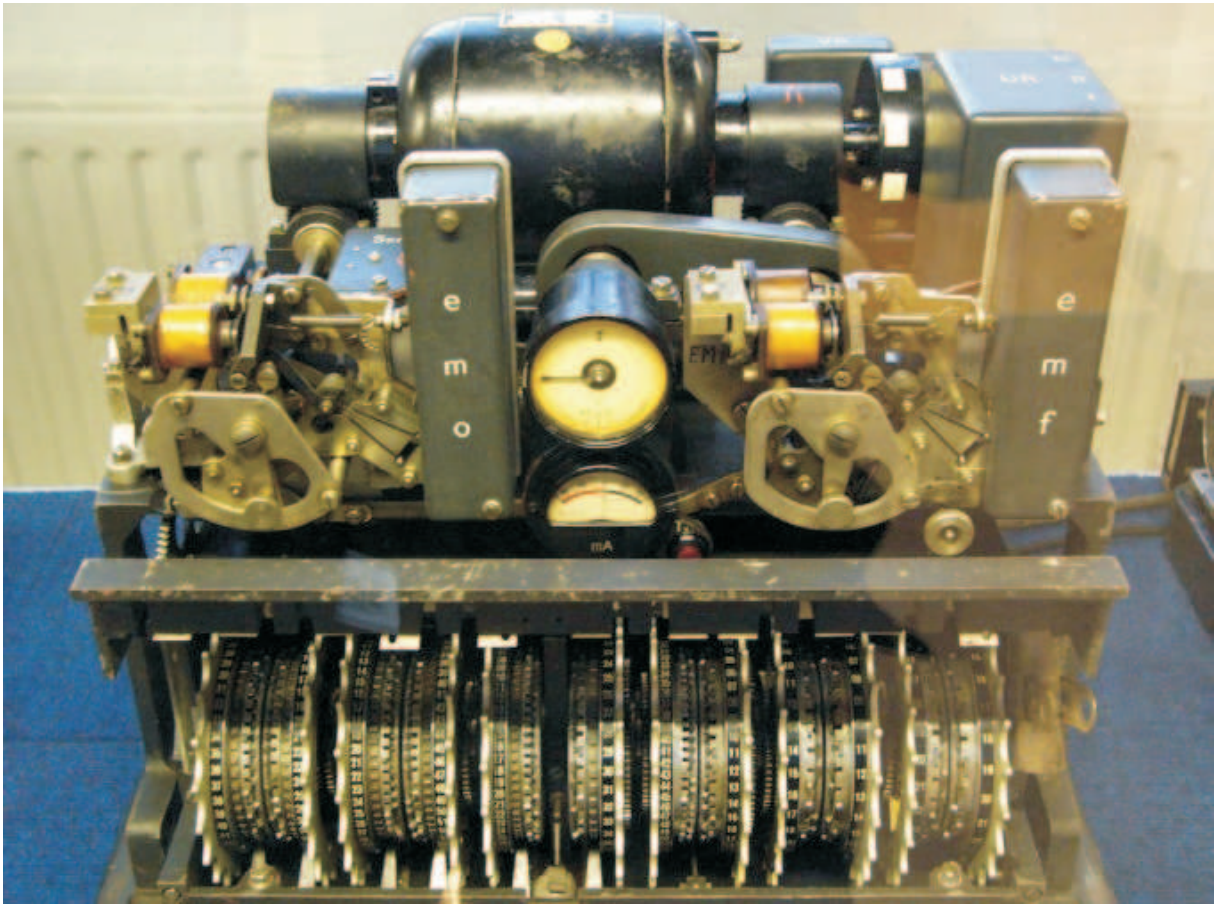
Da bi lahko ocenjevali varnost šifer moramo poznati vsaj osnovne vrste napadov nanje. Napad na šifro lahko izkorišča določene šibkosti v samem postopku ali pa šibkost šifrirnega oziroma dešifrirnega ključa. Napadi na šifre so prilagojeni konkretni šifri, kar pomeni, da je potrebno pri načrtovanju šifer zelo dobro preučiti njihove šibkosti in nato še praktično preizkusiti njihovo varnost. Praktično se varnost neke šifre preizkuša tako, da se širšemu krogu strokovnjakov postavi izziv napada nanjo. Običajno se, da bi bila motivacija večja, za uspešen napad ponudi neko nagrado.

Poleg specifičnosti, ki so prilagojene konkretnim šifram, se napadi na šifrirne postopke razlikujejo glede na cilj napada in na glede na to, kaj ima napadalec pri napadu na voljo. Glede na cilj napada ločimo:

- *Dešifriranje enega šifropisa.* To je osnovni cilj napada, kadar ima napadalec na voljo šifropis in bi želel poznati njegovo vsebino.
- *Razkrivanje ključa.* Z razkrivanjem ključa pridobi napadalec možnost dešifriranja vseh šifropisov, ki so bili šifrirani s tem ključem. S tem postane ključ neuporaben.

- *Razbijanje šifre.* Če tak napad na šifro uspe, lahko napadalec dešifrira vse šifropise, ki so bili šifrirani s to šifro, ne glede na uporabljeni ključ. S tem postane tudi šifra neuporabna.

V obeh zadnjih primerih skuša napadalec čim dalje prikrivati uspešnost napada, saj mu to omogoča dešifriranje vseh šifropisov, ki so narejeni z istim ključem oziroma z isto šifro. Dobro znan primer takega prikrivanja je iz II. svetovne vojne. Nemci so imeli že leta od leta 1918 na voljo napravo za šifriranje Enigma. Izpopolnjena različica Enigme je prikazana na sliki 3.1.




Slika 3.1 – Šifrirna naprava Enigma. To napravo so Nemci uporabljali za šifriranje vojaških sporočil med II. svetovno vojno.

To napravo so uporabljali za šifriranje vojaških sporočil, ki so jih prenašali preko radijskih zvez. Prepričani so bili, da se šifer Enigme ne da razbiti, vendar so pri tem predpostavljali, da bo napade nanje izvajal človek. Zaveznikom pa je uspelo že kmalu po začetku vojne dešifrirati šifropise narejene z Enigmo s pomočjo prvega računalnika na svetu Colossus I in kasneje izboljšane verzije Colossus II.

Ker so hoteli to res čim bolje izkoristiti, na veliko sporočil, ki so jih prestregli in uspešno dešifrirali, niso reagirali. Če bi namreč uporabili informacije, ki so jih na ta način pridobili, bi lahko dobili neko taktično prednost, vendar bi na ta način tudi Nemcem razkrili, da je bila šifra razbita. Zato so čakali na informacije strateškega pomena in to uspešno izkoristili pri invaziji

na Normandijo, ki je pomenila preokret v II. svetovni vojni.

Glede na to, kaj ima napadalec na voljo, delimo napade na:

- *Napad z znanim šifropisom.* Napadalec ima tu na voljo samo enega ali več šifropisov.
- *Napad z znanim šifropisom in šifro.* Tu ima napadalec na voljo enega ali več šifropisov, pozna pa tudi šifro. Kot smo že omenili, so danes šifre običajno znane, zato jih pozna tudi napadalec.
- *Napad z znanim parom čistopis šifropis.* Pri tem napadu uspe napadalcu pridobiti enega ali več parov čistopis-šifropis. Na osnovi teh parov skuša razkriti ključ, oziroma šifro, s katero so bili šifrirani.
- Napad z neomejenim številom parov čistopis-šifropis. Določene šifre (asimetrične šifre) imajo ločena ključa za šifriranje in dešifriranje. Ključ za šifriranje pri teh šifrah ni tajen, zato ga pozna tudi napadalec in lahko sam ustvari poljubno število parov čistopis-šifropis. Z njihovo analizo potem skuša odkriti tajni ključ za dešifriranje.

Najpreprostejši, ampak tudi najmanj učinkovit je napad, pri katerem napadalec preizkusi vse možne ključe. Takemu napadu rečemo tudi napad grobe sile. Možnost uspeha pri napadu grobe sile zmanjšamo oziroma onemogočimo z izbiro dovolj močnih ključev, to je ključev, ki imajo izredno veliko število možnih vrednosti.

Za opisan primer Cezarjeve šifre lahko ugotovimo, da je ključ izredno šibek, saj je možnih zgolj $L-1$ kjer je $L$ število črk v abecedi. Za slovensko abecedo je $L$ enak 25, če pa upoštevamo, da so v abecedi tudi številke, velike in majhne črke ter posebni znaki, pa je $L$ enak 96. V obeh primerih je nekomu, ki ne pozna ključa, zelo preprosto preizkusiti vse možne ključe in na ta način dešifrirati sporočilo. Sodobne šifre uporabljajo bistveno močnejše ključe in na ta način onemogočajo napad grobe sile. Tipično se danes uporabljajo vsaj 128-bitni ključi, ki imajo lahko $2^128$ različnih vrednosti.

Bolj zapleteni, a zato lahko uspešnejši napadi slonijo na analizi šifropisa, kadar imamo na voljo samo šifropis, oziroma na analizi povezave med čistopisom in šifropisom, kadar imamo na voljo pare čistopis-šifropis. Pri analizi se uporabljajo različni matematični postopki s področja statistike, verjetnosti in teorije števil.

Natančnejši opis postopkov kriptoanalize presega okvire te knjige. Za ilustracijo si oglejmo le primer dešifriranja šifropisa, ki je dobljen s preprosto zamenjavo črk v šifropisu z nekimi drugimi znaki oziroma črkami. Tak postopek je opisal Arthur Conan Doyle, avtor zgodb o detektivu Sherlocku Holmesu, v zgodbi Avantura plešočega moža (The Adventure of the Dancing Men). V zgodbi Sherlock Holmes uspešno dešifrira šifropise, kjer so črke teksta zamenjane s plešočimi možički, kot je to prikazano na sliki 3.2.

Tak način šifriranja imenujemo zamenjalno ali substitucijsko šifriranje. Šifriranje se izvaja preko substitucijske tabele, v kateri je vsakemu znaku abecede čistopisa pripisan znak iz abecede šifropisa. V zgornjem primeru je torej vsaki črki abecede predpisan ustrezen možiček.

Slika 3.2 – Primer šifropisa iz zgodbe Avantura plešočega moža avtorja Arthurja Conana Doyla. Črke čistopisa so zamenjane z sličicami plešočih možičkov. Tak način imenujemo tudi substitucijsko šifriranje.

Za dešifriranje je Sherlock Holmes uporabil tako imenovano frekvenčno analizo. Pri frekvenčni analizi upoštevamo pogostost posameznih znakov, oziroma kombinacij znakov v tekstu.

Presledek je v vsakem tekstu najpogostejši znak. Pri preprostem substitucijskem šifriranju lahko takoj ugotovimo, da je predstavlja znak, ki v šifropisu nastopa najpogosteje, presledek. V šifropisu nato najprej zamenjamo vse te znake s presledki. V angleškem tekstu se zelo pogosto pojavlja določni člen *the*. Zato poiščemo v šifropisu kombinacijo treh črk, ki se zelo pogosto pojavlja, in na ta način odkrijemo znake za tri črke: *t, h* in *e*. V šifropisu nato zamenjamo te znake z ustreznimi črkami. Črka, ki v angleškem tekstu najpogosteje stoji sama je nedoločni člen *a*. Ko nadomestimo v šifropisu vse znake, ki predstavljajo *a* s črko *a*, lahko skoraj zagotovo uganemo še kakšno besedo v tekstu, in na ta način ugotovimo pomen znakov, ki nastopajo v tej besedi. Ker je v tekstu, ki ga dešifriramo, vedno več znanih znakov, postaja tudi ugibanje besed vedno preprostejše. S postopkom nadaljujemo, dokler ne ugotovimo pomena vseh znakov in na ta način v celoti dešifriramo šifropis.

V praksi se preprosto substitucijsko šifriranje ne uporablja, zato tudi postopki kriptoanalize niso tako preprosti. Bistvo vsake kriptoanalize pa je, da skuša najti in izkoristiti šibke točke šifre.

Ko govorimo o varnosti šifer ločimo teoretično in praktično varnost. Teoretično varna je šifra, ki je tudi teoretično ni mogoče razbiti, četudi ima napadalec na voljo neomejeno sredstev in neomejeno časa.

Za zašito podatkov pa v praksi zadoščajo že praktično varne šifre. Za šifro rečemo, da je praktično varna, če je uspešen napad nanjo praktično neizvedljiv. S pojmom praktično neizvedljiv označujemo, da je verjetnost, da bo napadalec razbil šifro v omejenem času z omejenimi sredstvi, dovolj majhna. Kolikšna sme biti ta verjetnost in kolikšna sredstva in čas lahko napadalec uporabi za razbijanje šifre je odvisno od vrste in vrednosti podatkov, ki jih varujemo. Če stroški napada močno presegajo vrednost podatkov oziroma če je potreben čas, ki bi bil potreben za razbijanje šifre bistveno večji od časa aktualnosti podatkov, potem lahko rečemo, da je šifra praktično varna.

Pri simetričnih šifrah je ključ za šifriranje enak ključu za dešifriranje. Udeleženci komunikacije si morajo zato ključ pred komunikacijo na nek varen način izmenjati. Ključ je v splošnem pri teh šifrah bistveno krajši od sporočila. Iz praktičnih razlogov želimo tudi, da je možno isti ključ uporabiti večkrat.

Večkratna uporaba istega ključa postavlja dodatno zahtevo. Da bi lahko šteli šifro za praktično varno, mora biti praktično neizvedljivo, da na osnovi enega ali več parov čistopis-

šifropis pridobimo tajni ključ. Če ta pogoj ne bi bil izpolnjen, bi napadalec s tem, da je nekako pridobil tak par, lahko dešifriral vsa nadaljnja sporočila šifrirana z istim ključem.

Sodobne šifre so izredno močne. Načrtovane so tako, da so odporne proti vsem znanim postopkom kriptoanalize. Za uporabo šifriranja pri varovanju IKSov se priporočajo šifre, na katere ni znanih uspešno izvedenih napadov. Napadalci na IKS zato ponavadi ne skušajo razbijati šifer, temveč raje iščejo druge slabosti v varovanju IKSa, ki ga napadajo. Nekatere od teh slabosti, ki se jim želimo izogniti, bomo obravnavali tudi v naslednjih poglavjih.

## 3.2 Izločanje redundance

Tako pri napadu grobe sile kot tudi pri drugih napadih na šifre izkoriščajo napadalci redundanco v zapisu čistopisa. Ker so elektronska sporočila običajno zapisana binarno, to je kot niz binarnih simbolov ali bitov (logičnih ničel in enic), bomo v nadaljevanju, zaradi preprostosti, predpostavljali binarni zapis. Pri binarnem zapisu čistopisa lahko redundanco izračunamo po enačbi:

$$R = 1 - \frac{M}{N} \tag{3.1}$$

kjer je $N$ dejanska dolžina sporočila v bitih in $M$ minimalno število bitov, ki bi jih potrebovali za kodiranje sporočila, ne da bi se informacija izgubila in bi bilo možno sporočilo zopet dekodirati. $M$ je odvisen od informacije, ki jo vsebuje sporočilo, oziroma natančneje od njegove entropije, to je povprečne informacije, ki jo vsebujejo taka sporočila.

Kadar je v sporočilu prisotna redundanca, so določene kombinacije bitov čistopisa (določene kombinacije črk v tekstu) prepovedane. Na osnovi tega lahko na primer lektorji popravljajo tipkarske napake v tekstu, saj napake pogosto povzročijo nesmiselno kombinacijo črk.

Če bi uspeli iz čistopisa izločiti vso redundanco, bi bila vsaka kombinacija bitov smiselna. Pri napadu grobe sile, to je pri preizkušanju vseh možnih ključev, bi dobil napadalec pri vsakem preizkusu nek smiselno sporočilo. Napadalec zato ne bi imel nobene možnosti, da ugotovi, katero od teh sporočil je pravo. Šifriranje čistopisa brez redundance bi bilo zato tudi teoretično varno.

Postopkov, ki bi izločili vso redundanco iz poljubnega sporočila, žal ne poznamo, obstajajo pa postopki za stiskanje sporočil (kot npr. zip, rar, arj in podobni), ki iz sporočila izločijo večino redundance. Zato, da v sporočilu ni redundance, ne zadošča zgolj, da so vse kombinacije bitov v sporočilu dovoljene, temveč morajo biti tudi vse enako verjetne oziroma se morajo pojavljati enako pogosto, drugače si lahko napadalec pomaga s pogostostjo pojavljanja določenih kombinacij.

Čeprav s stiskanjem čistopisa ne moremo izločiti vse redundance, jo lahko s postopki za stiskanje močno zmanjšamo, kar močno otežkoča kriptoanalizo in na ta način povečuje varnost vsake šifre. Zato je smiselno, da čistopise pred šifriranjem vedno stisnemo. Večina programov za šifriranje ima stiskanje že vključeno v samo šifro.

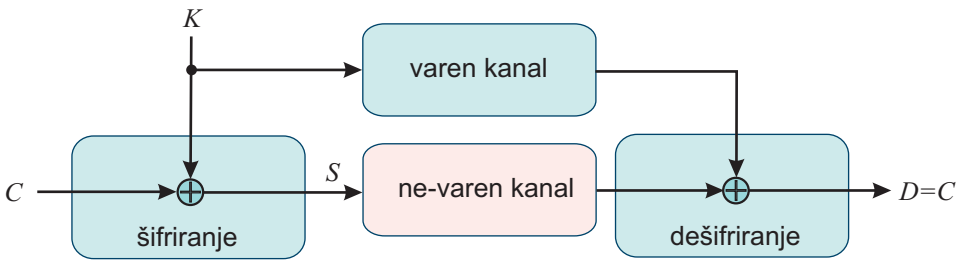
## 3.3 Pretočne šifre

Pretočne šifre šifrirajo posamezne simbole čistopisa po vrsti, kot nastopajo v čistopisu. Pretočne šifre so zato zelo primerne za šifriranje sporočil, ki jih prenašamo v realnem času, kot je to na primer šifriranje govornega signala v telefoniji.

### 3.3.1 Teoretično varna šifra

Varnost šifer je v veliki meri preučeval Claude Elwood Shannon [1], oče teorije informacij. On je tudi predlagal teoretično varno šifro z enkratno uporabo ključa. To je pretočna šifrira, pri kateri je ključ binaren niz enake dolžine kot čistopis. Ključ mora biti popolnoma naključen, uporabiti pa ga smemo samo enkrat, za šifriranje enega samega sporočila.

Shematsko je postopek šifriranja in dešifriranja prikazan na sliki 3.3.




Slika 3.3 – Teoretično varna šifra z enkratno uporabo ključa

Šifriranje je izredno preprosto in ga izvedemo s preprosto operacijo XOR med posameznimi biti čistopisa in ključa. Označimo z **C** čistopis, s $K_s$ šifrirni ključ in z **S** šifropis. Potem velja:

$$\mathbf{S} = \mathbf{C} \oplus \mathbf{K}_s \tag{3.2}$$

in

$$\mathbf{D} = \mathbf{S} \oplus \mathbf{K}_d \tag{3.3}$$

kjer je smo z $\oplus$ označili operacijo XOR, z **D** dešifriran šifropis in s $\mathbf{K}_d$ dešifrirni ključ. Dešifriran šifropis je enak čistopisu ($\mathbf{D} = \mathbf{C}$) tedaj in le tedaj, kadar je dešifrirni ključ enak šifrirnemu ključu ($K_d = K_s$). To je torej simetrična šifrira z istim ključem za šifriranje in dešifriranje ($\mathbf{K} = K_s = K_d$).

Namesto binarnega zapisa bi bili lahko čistopis, ključ in šifropis zapisani v številskem sistemu s poljubno bazo $L$, le da bi morali namesto operacije XOR pri šifriranju izvajati operacijo seštevanja po modulu $L$ in pri dešifriranju operacijo odštevanja po modulu $L$.

**Primer 3.1**

Kot primer vzemimo, da želimo poslati šifrirano sporočilo "Ejga!".

Najprej sporočilo binarno kodiramo, tako da vsak znak zapišemo z osmimi biti iz ASCII tabele:

$$\mathbf{C} = 01000101011010100110011101100001001000001$$

Ker smo vsak znak zapisali z osmimi biti, je dolžina čistopisa $N$ enaka

$$N = 5\,\text{znkov} \cdot 8\,\text{bitov/znak} = 40\,\text{bitov}$$

Za šifriranje moramo uporabiti $N$ bitov dolg naključen ključ. Naj bo ključ enak:

$$\mathbf{K} = 0110010011011011101101010111100100111110$$

V skladu z izrazom (3.2) je potem šifropis enak:

$$\mathbf{S} = 0010000110110001110100100001100000011111$$

Dešifriramo tako, da izvedemo XOR operacijo med šifropisom **S** in ključem **K**. Dobimo:

$$\mathbf{D} = 0100010101101010011001110110000100100001$$

kar je enako čistopisu **C**, tako da z dekodiranjem dobimo nazaj sporočilo "Ejga!".

---

Če ne poznamo ključa, je tak šifropis nemogoče dešifrirati. Edini napad, ki bi ga lahko izvedli, je napad grobe sile, to je preizkušanje vseh možnih $2^N$ (v zgornjem primeru 1.099.511.627.776) ključev. Tudi, če bi imeli dovolj časa in sredstev, da preizkusimo vse možne ključe, to ne bi pomagalo, saj bi kot rezultat dobili vsa možna sporočila dolžine $N$ in nikakor ne bi mogli ugotoviti, katero med njimi je pravo.

**Primer 3.2**

---

V primeru 3.1 bi s preizkušanjem ključa

$$\mathbf{K} = 0110101111010000101111000111110101100101$$

dobili sporočilo "Janez", s preizkušanjem ključa

$$\mathbf{K} = 0111011111011110101101100111001101111110$$

pa bi dobili sporočilo "Vodka".

---

Ker napadalec nima nobene možnosti, da bi ugotovil, katero od vseh možnih sporočil dolžine $N$ je bilo poslano, je šifro nemogoče zlomiti ne glede na čas in sredstva, ki jih napadalec vloži, zato pravimo, da je taka šifra teoretično varna.

Glavna pomankljivost teoretično varne šifre je težava v izmenjavi ključev. Isti, naključno generiran ključ, ki je enak dolžini sporočila, morata poznati tako pošiljatelj kot tudi prejemnik sporočila. To pomeni, da si morata ključ, pred komunikacijo izmenjati na nek varen način in ga tudi varno hraniti. Izredno dolg ključ, ki je narejen z nekim naključnim procesom, lahko oba hranita na disku, nato pa ga po vrsti uporabljata za vsako šifrirano sporočilo. Vsak del ključa pa se sme uporabiti samo enkrat, zato je skupna dolžina sporočil, ki jih je mogoče prenesti omejena z dolžino ključa.

Ker je šifra z enkratno uporabo ključa teoretično varna, obenem pa je, razen problema izmenjave ključev, izredno preprost postopek, so jo uporabljali v času hladne vojne za šifriranje rdečega telefona med Belo hišo in Kremljem.

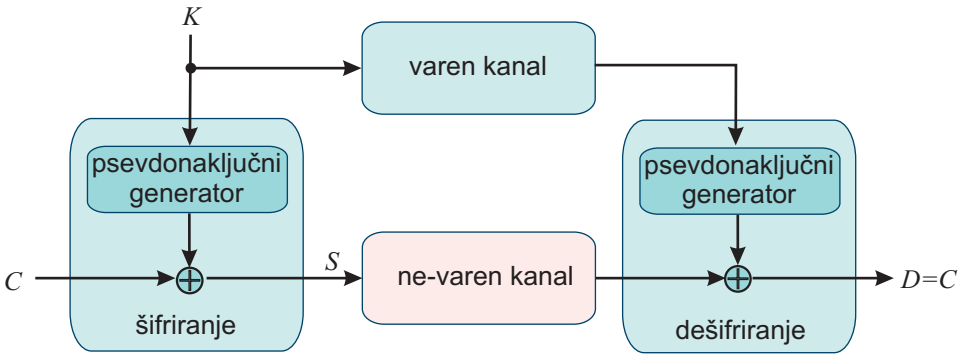
Nujno pri tej šifri je, da vsak ključ uporabimo samo enkrat. Če bi namreč isti ključ uporabili za šifriranje dveh sporočil, bi lahko napadalec, ki bi imel obe na voljo, dokaj preprosto dešifriral obe sporočili. Za to mu ne bi bilo potrebno preizkušati vseh ključev. Sporočili bi lahko dešifriral po majhnih delih, tako, da bi iskal ključe, ki dajo pri dešifriranju obeh šifropisov smiselen tekst. Večje kot bi bilo število sporočil šifriranih z istim ključem, lažje bi bilo njihovo dešifriranje.

Enak razmislek velja tudi, če bi pri šifriranju sporočila večkrat uporabili ključ, ki je krajši od sporočila. Napadalec bi preprosto preizkušal, kateri ključi dajo smiselen tekst pri dešifriranju različnih delov sporočila.

### 3.3.2 Razširjanje ključa

Pri opisani teoretično varni šifri nastopijo v praksi težave pri varni izmenjavi in hrambi zelo dolgega ključa. Ključ mora biti namreč vsaj toliko dolg, kot je skupna dolžina vseh sporočil, ki jih želimo s tem ključem izmenjati. Zato bi želeli, da je ključ krajši od sporočila in da je mogoče isti ključ večkrat uporabiti.

To dosežemo tako, da ključ uporabimo kot seme generatorja psevdonaključnega niza (PRS - Pseudo Random Sequence). Na ta način smo ključ razširili. Naključni niz, ki je odvisen od semena, namreč uporabimo kot ključ pretočne šifre, kot je to prikazano na sliki 3.4.



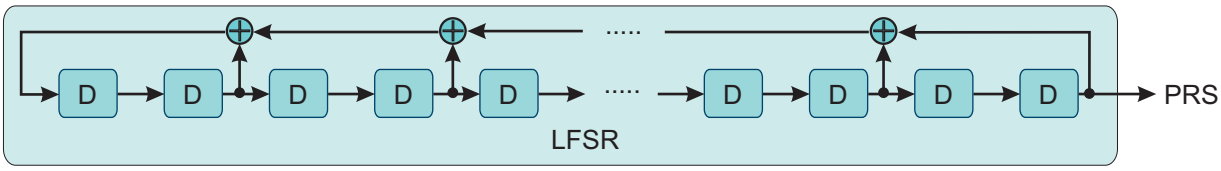

Slika 3.4 – Pretočna šifra z razširjanjem ključa

Za varnost take šifre je pomembno:

- da je PRS odvisen od ključa (semena) in na videz popolnoma naključen,
- da je ključ dovolj dolg, da onemogoča napad grobe sile in
- da iz nekega znanega PRS ni mogoče ugotoviti ključa.

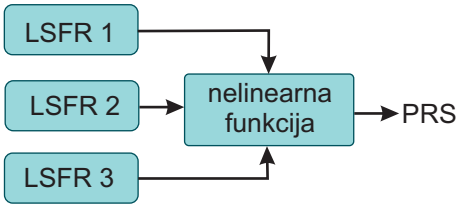
Pri pretočnih šifrah z razširjanjem ključa je torej zelo pomembna izbira ustreznega generatorja psevdonaključnega niza. Pri tem je pomembna tako njegova kriptografska moč, to je izpolnjevanje zgoraj naštetih lastnosti, kot tudi preprostost izvedbe in računska učinkovitost.

V ta namen se pogosto uporabljajo pomikalni registri z linearno povratno vezavo (LFSR - Linear Feedback Shift Register). LFSR dolžine $N$ sestavlja $N$ zakasnilnih D celic. D celica ob digitalnem taktu prenese bit iz svojega vhoda na svoj izhod. Določeni izhodi D celic so preko XOR vrat povezani v povratno vezavo, na vhod prve D celice, kot je to prikazana na sliki 3.5.




Slika 3.5 – Pomikalni register z linearno povratno vezavo

PRS na izhodu LFSR je odvisen od povratnih vezav in od začetnega stanja v celicah pomikalnega registra. PRS je vedno periodičen, to pomeni, da se začne po določenem številu izhodnih bitov ponavljati. Samo določene kombinacije povratnih vezav dajejo najdaljšo periodo PRS, ki je enaka $2^N - 1$ simbolov. LFSR z maksimalno dolžino PRS imenujemo LFSR maksimalne dolžine (MLLFSR - Maximum Length LFSR). Ključ bi v tem primeru predstavljale številke odcepov s povratnimi vezavami, vendar to ni najbolj primeren način, saj je število različnih ključev močno omejeno na kombinacije povratnih vezav, ki dajo najdaljšo periodo LFSR. Poleg tega je mogoče na osnovi $2N$ znanih bitov PRS ugotoviti, katere povratne vezave so bile uporabljene. Do teh bitov lahko pride napadalec zgolj na osnovi enega znanega para čistopis - šifropis, torej lahko tudi na osnovi enega para razkrije ključ.



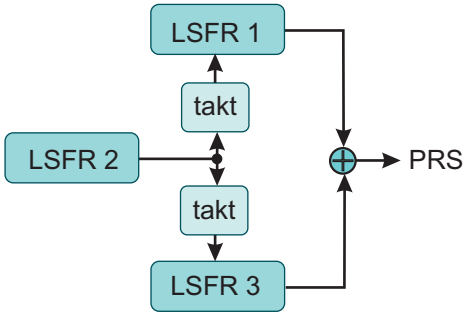

Slika 3.6 – Nelinearna kombinacija več LFSR za generacijo PRS

Namesto enega LFSR se zato uporablja več LFSR različnih dolžin, njihovi izhodi pa se potem kombinirajo z neko nelinearno funkcijo, kot je to prikazano na sliki 3.6. Nelinearne funkcije so namreč bistveno bolj odporne proti kriptoanalizi.

Pogosto se dosega nelinearnost s kontrolo digitalnega takta posameznih LFSR. Na sliki 3.7 je prikazan primer, ko LFSR 2 kontrolira uro LFSR 1 in LFSR 3.

Če je na izhodu LFSR 2 logična enica, se ob uri pomaknejo biti v LFSR 1, če pa je izhodu LFSR 2 logična nič, pa se ob uri pomaknejo biti v LFSR 3. Izhodni PRS dobimo z XOR kombinacijo izhodov LFSR 1 in LFSR 3. V tem primeru so povratne vezave v vseh LFSR stalne in lahko javno znane kot del šifre, ključ pa je začetno stanje vseh LFSR.

Primer pretočne šifre je tudi algoritem A5, ki se uporablja v GSM telefoniji. Uporablja ti LFSR dolžin 19. 22 in 23. Algoritem so najprej držali v tajnosti, vendar je sedaj že objavljen




Slika 3.7 – LFSR s kontroliranim digitalnim taktom

tudi na svetovnem spletu. Kriptoanalitiki so dokaj hitro odkrili njegove šibkosti, tako da sedaj že obstajajo naprave za njegovo razbijanje.

Ugotavljanje šibkosti posameznih šifer zahteva poznavanje vseh znanih napadov nanje in analizo odpornosti nanje. Mnoge šifre, ki so jih imeli na začetku za varne, so se izkazale za šibke, ko so bile izpostavljene širši skupnosti kriptoanalitikov. Sami zato ne moremo vedeti, ali je določena šifra zares varna in se moramo zato zanašati na mnenje specialistov s področja kriptoanalize. Pri tem je mnogo laže zaupati šifram, ki so bile pred uporabo javno objavljeni in tako izpostavljeni zelo širokemu krogu specialistov in obširnemu testiranju.

Poleg LFSR obstajajo številni drugi generatorji PRS, vendar so zaradi svoje preprostosti LFSR najpogosteje v uporabi. Omeniti velja, da so LFSR izredno primerni za strojno izvedbo, kjer se vse operacije izvajajo vzporedno, in mnogo maj za programsko izvedbo, kjer se morajo posamezne operacije izvrševati zaporedno, to je druga za drugo.

## 3.4 Blokovne šifre

Blokovne šifre razdelijo čistopis v bloke enake dolžine. Vsak blok potem ločeno preslikajo v ustrezen blok šifropisa. Na blokovno šifro lahko gledamo kot na substitucijsko šifro, ki je opisana v razdelku 3.1. Vsak blok čistopisa namreč zamenjamo z blokom šifropisa, kar lahko naredimo s pomočjo substitucijske tabele. Če bi bili bloki dolgi 8 bitov, bi dejansko preslikali 8-bitne znake čistopisa v 8-bitne znake šifropisa.

Kot smo videli, lahko pri kratkih blokih tako šifro razbijemo s frekvenčno analizo. Kriptografsko močno šifro dobimo, če so bloki dovolj dolgi, da je verjetnost njihovega ponavljanja zanemarljiva, kar onemogoča frekvenčno analizo. Pri tem je zaželeno, da se blokom čistopisa v substitucijski tabeli priredijo bloki šifropisa na videz popolnoma naključno, to je brez kakršnega koli pravila, ki bi ga lahko napadalec pri napadu na postopek odkril. Substitucijska tabela v tem primeru predstavlja tudi ključ za šifriranje in dešifriranje. Primer substitucijske tabele za kratke bloke dolžine $N = 4$ bitov, je prikazan na sliki 3.8.

Tak postopek pa je zanimiv predvsem teoretično, saj zaradi dolžine substitucijske tabele in s tem tudi dolžine ključa, ki bi ga bilo potrebno izmenjati, praktično ni uporaben. Pri $N$ bitov dolgih blokih bi morala imeti substitucijska tabela $2^N$ vrstic.

| C | S |
|---|---|
| 0000 | 0110 |
| 0001 | 0011 |
| 0010 | 1001 |
| 0011 | 0100 |
| 0100 | 0111 |
| 0101 | 0001 |
| 0110 | 1000 |
| 0111 | 1110 |
| 1000 | 1011 |
| 1001 | 0000 |
| 1010 | 1100 |
| 1011 | 1101 |
| 1100 | 1111 |
| 1101 | 0101 |
| 1110 | 1010 |
| 1111 | 0010 |

Slika 3.8 – Primer substitucijske tabele za bloke dolge $N = 4$ bite. Substitucijska tabela je uporabna pri šifriranju in dešifriranju in zato služi tudi kot ključ.

Za praktično izvedbo blokovnega šifriranja zato potrebujemo neko šifrirno funkcijo $f_s$, ki blok čistopisa $\mathbf{C}_n$ preslika v blok šifropisa $\mathbf{S}_n$ v odvisnosti od ključa $\mathbf{K}$:

$$\mathbf{S}_n = f_s(\mathbf{C}_n, \mathbf{K}) \tag{3.4}$$

na tak način, da obstaja tudi inverzna funkcija za dešifriranje $f_d$, ki blok šifropisa $\mathbf{S}_n$ preslika nazaj v blok čistopisa $\mathbf{C}_n$, če pri dešifriranju uporabimo isti ključ $\mathbf{K}$:

$$\mathbf{C}_n = f_d(\mathbf{S}_n, \mathbf{K}) \tag{3.5}$$

Blokovna šifra mora izpolnjevati določene pogoje, da je izvedljiva in varna.

- Dolžina ključa $\mathbf{K}$ mora biti dovolj velika, da onemogoča napad grobe sile.
- Dolžina blokov mora biti dovolj velika, da je verjetnost ponavljanja istih blokov minimalna.
- Preslikava, ki jo naredi funkcija $f_s$, mora biti na videz naključna, kar pomeni
  - da so bloki šifropisa statistično neodvisni od blokov ključa in
  - da so bloki šifropisa statistično neodvisni od ključa,

  kar onemogoča tako napad s frekvenčno analizo na osnovi znanega šifropisa kot tudi napad na osnovi znanih parov čistopis-šifropis.
- Funkciji $f_s$ in $f_d$ morata biti dovolj preprosti, da omogočata učinkovito izvedbo.

Preprost test varnosti blokovne šifre je sprememba enega bita čistopisa. Sprememba katerega koli bita čistopisa mora povzročiti spremembo približno polovice bitov na naključnih lokacijah v šifropisu. To je potreben, ne pa tudi zadosten pogoj, da je šifra varna.

Funkcijo $f_d$ običajno sestavlja kombinacija večjega števila osnovnih operacij, ki same ne nudijo zadostne varnosti, s kombiniranjem pa se njihova varnost poveča. Osnove operacije vključujejo:

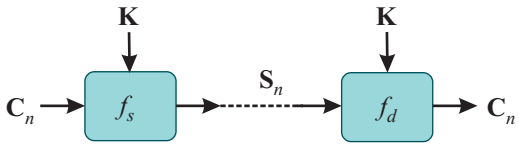
- substitucije, to je zamenjave posameznih simbolov ali skupin simbolov),
- permutacije, to je zamenjave mest posameznih simbolov ali skupin simbolov v bloku,
- aritmetične operacije, kot na primer seštevanje, odštevanje in množenje po modulu
- linearne preslikave, kot na primer množenje s konstantno matriko, in druge.

### 3.4.1 Načini blokovnega šifriranja

Blokovne šifre se, poleg samega postopka šifriranja bloka, razlikujejo tudi po tem, kako se bloki obravnavajo pri šifriranju, kadar je v čistopisu več blokov.

#### Elektronska kodna knjiga

Elektronska kodna knjiga (ECB – Electronic Code Book) je osnovni način delovanja blokovnih šifer. Pri tem načinu se vsak blok čistopisa šifrira ločeno, neodvisno od ostalih blokov. Identični bloki čistopisa se z istim ključem vedno preslikajo v identične bloke šifropisa. Napaka v enem bloku šifropisa povzroči napačno dešifriranje celega bloka, ne vpliva pa na dešifriranje ostalih blokov. Postopek šifriranja in dešifriranja je prikazan na sliki 3.9.



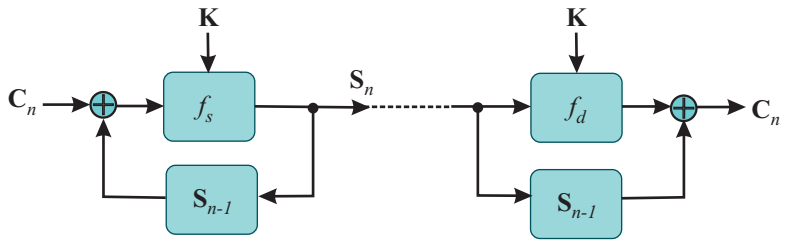

Slika 3.9 – ECB način šifriranja. Vsak blok se šifrira in dešifrira neodvisno od ostalih blokov

Elektronska knjiga ni najprimernejši način blokovnega šifrirana. Ker se posamezni bloki šifrirajo neodvisno drug od drugega, ima napadalec na voljo veliko število blokov, iz katerih lahko skuša odkriti ključ, s katerim je sporočilo šifrirano. Poleg tega lahko napadalec v sporočilo vriva šifrirane bloke iz preteklih šifropisov. Ti bloki se potem pravilno dešifrirajo.

#### Veriženje blokov

Pri načinu z veriženjem blokov (CBC – Cipher-Block Chaining) pred šifriranjem trenutnemu bloku čistopisa $\mathbf{C}_n$ po modulu 2 prištejemo prejšnji blok šifropisa $S_{n-1}$. Ker pri šifriranju prvega bloka čistopisa $C_1$ pretekli blok šifropisa $\mathbf{S}_0$ ne obstaja, uporabimo za $\mathbf{S}_0$ nek poljuben inicializacijski blok. Pri dešifriranju moramo dešifriranemu bloku zopet po modulu 2 prišteti

prejšnji šifriran blok, da dobimo nazaj čistopis. Šifriranje in dešifriranje je shematsko prikazano na sliki 3.10.



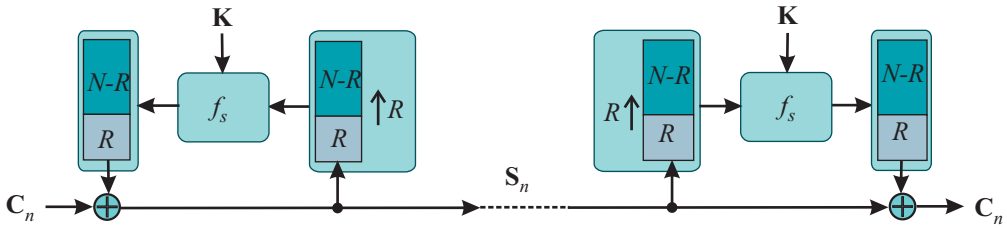

Slika 3.10 – Šifriranje z veriženjem blokov. Sprememba enega bloka, ključa ali inicializacijskega bloka, vpliva na šifriranje vseh nadaljnjih blokov.

Pri veriženju blokov dobimo iz identičnih čistopisov identične šifropise, če pri šifriranju uporabimo isti ključ in isti inicializacijski blok $\mathbf{S}_0$. Ker je šifriranje enega bloka odvisno od prejšnjega bloka, moramo bloke dešifrirati v enakem vrstnem redu, kot smo jih šifrirali. Napaka v enem šifriranem bloku vpliva na dešifriranje dveh blokov: bloka z napako in tudi naslednjega bloka, ki se mu ta blok prišteva.

Ker napadalec ne more dešifrirati drugega bloka, ne da bi prej dešifriral prvi blok, je napad na CBC način bistveno težji od napada na ECB.

### Šifropis v povratni vezavi

Pri načinu s šifropisom v povratni vezavi (CFB – Cipher FeedBack) se izvaja šifriranje v $N$-bitov dolgih blokov, pri čemer se šifrira v vsakem koraku samo $R < N$ novih bitov čistopisa. $R$ bitov na izhodu blokovnega šifrirnika se po modulu 2 prišteje $R$ bitom čistopisa in šifrira skupaj s preteklimi $N-R$ biti. Tako kot pri načinu CBC tudi tu potrebujemo $N$-bitov dolg inicializacijski šifropis $\mathbf{S}_0$. Vsi ostali bloki šifropisa so dolgi $R$-bitov. Potek šifriranja in dešifriranja v načinu CFB je prikazan na sliki 3.11.




Slika 3.11 – Šifriranje s šifropisom v povratni vezavi. Z $N$ bitom dolgim šifrirnikom se naenkrat šifrira samo $R < N$ bitov čistopisa.

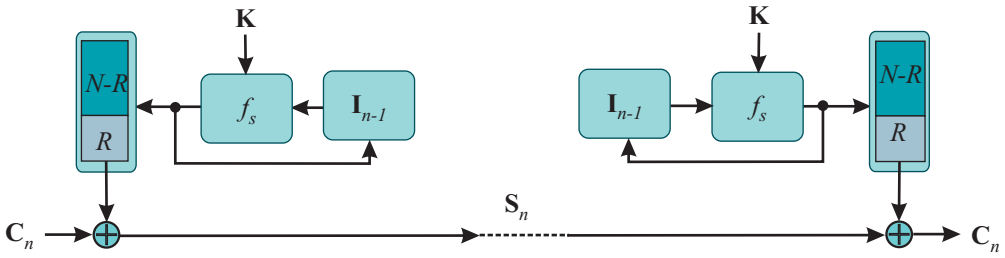
Tudi pri šifriranju s šifropisom v povratni vezavi dobimo z istim ključem $\mathbf{K}$ in istim inicializacijskim šifropisom $\mathbf{S}_0$ iz identičnih čistopisov identične šifropise. Bloke šifropisa moramo dešifrirati v enakem vrstnem redu kot smo jih šifrirali. Napaka v šifropisu se tu razširi še čez naslednjih $N$ bitov. Šifriranje naslednjih blokov je odvisno od prejšnjih blokov. Sprememba enega bita v prvem bloku čistopisa spremeni šifropis v vseh naslednjih blokih.

CFB je nekoliko varnejši pred napadi kot CBC, vendar je zato računsko zahtevnejši.

Omeniti velja tudi, da uporabimo pri načinu CFB isto funkcijo $f_s$ tako pri šifriranju kot pri dešifriranju, kar poenostavi načrtovanje samega šifrirnaega postopka kot tudi njegovo izvedbo.

### Izhod v povratni vezavi

Pri načinu z izhodom v povratni vezavi (OFB – Output FeedBack) vodimo pretekli izhod iz blokovnega šifrirnika $I_{n-1}$ nazaj na njegov vhod. Blokovni šifrirnik in inicializacijski izhod $\mathbf{I}_0$ sta dolga $N$ bitov, medtem ko so bloki čistopisa dolgi $R$ bitov ($R < N$). Šifriranje in dešifriranje po načinu OFB sta prikazana na sliki 3.12




Slika 3.12 – Šifriranje z izhodom v povratni vezavi. Dolžina kodirnika je $N$ bitov, bloki pa so dolgi $R < N$ bitov.

Identični šifropisi se šifrirajo identično z istim ključem $\mathbf{K}$ in istim inicializacijskim izhodom $\mathbf{I}_0$. Napaka v šifropisu se ne razširja. Povzroči samo napako na istem mestu čistopisa, kjer je nastala v šifropisu. Pri izgubi enega bita se izgubi sinhronizacija in dešifriran tekst ni več enak čistopisu. OFB je pravzaprav eden od načinov razširjanja ključa. Ker da isti inicializacijski vhod in isti ključ vedno enak razširjeni ključ, moramo pri ponovni uporabi istega ključa spremeniti inicializacijski vhod $\mathbf{I}_0$. Ni pa potrebno, da bi bil $\mathbf{S}_0$ tajen, zato ga lahko pošljemo s sporočilom. Računsko je postopek enako zahteven kot CFB.

Tudi pri načinu OFB uporabimo pri šifriranju in dešifriranju isto funkcijo $f_s$.

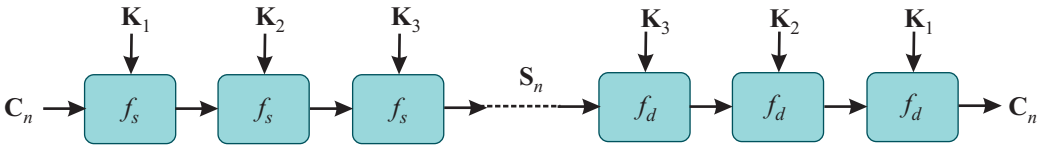
### Večkratno šifriranje

Določene računsko učinkovite in uveljavljene šifre imajo fiksno določeno dolžino ključa. Primer take šifre je DES (ang. Data Encryption Standard), pri katerem je ključ dolg 56 bitov. Z razvojem računalniške tehnologije se je izkazalo, da je tak ključ prekratek in omogoča napad grobe sile.

Moč šifre z omejeno dolžino ključa se da povečati z večkratnim šifriranjem, to je s kaskadno vezavo šifrirnikov z različnimi ključi, ki skupaj tvorijo daljši ključ, kot je to prikazano na sliki 3.13.

Pred razvojem novejših algoritmov se je zato pogosto uporabljalo trikratno šifriranje z DES algoritmom, tako imenovana 3DES šifra. Za dvakratno šifriranje namreč obstaja način, pri katerem zadošča samo enkratno preizkušanje posameznega ključa $K_1$ in $K_2$, tako da dvakratno šifriranje ne poveča bistveno varnosti postopka. Za trikratno šifriranje tak napad ni poznan.

Večkratno šifriranje lahko uporabimo v vseh zgoraj opisanih načinih.




Slika 3.13 – Večkratno šifriranje z različnim ključem efektivno podaljša dolžino ključa.

### 3.4.2 Šifra AES

Razvitih in v praksi uporabljenih je bilo izredno veliko število različnih blokovnih šifer. Med najbolj znanimi je DES, ki je bil dolga leta v komercialni uporabi, vendar je z razvojem tehnologije zaradi relativno kratkega 56 bitov dolgega ključa postal prešibak. Trenutno najbolj aktualna blokovna šifra je AES (ang. advanced encryption standard). AES je bil izbran kot najboljši postopek na tekmovanju, ki ga je leta 1997 objavil ameriški Nacionalni inštitut za standardizacijo in tehnologijo (NIST – National Institute of Standards and Technology). Najmanjša možna predpisana dolžina ključa je bila 128 bitov. Vsi prijavljeni postopki so bili dani v javno obravnavo. V avgustu 1999 je NIST izbral 5 finalistov, kandidatov za AES: MARS, RC6, Rijndael, Serpent, and Twofish. Kot kriterij za izbiro so služili predvsem ocenjena moč oziroma odpornost proti napadam, njegova računska učinkovitost in preprostost tako programske kot strojne izvedbe in izvedbe na pametnih karticah. Na koncu je zmagal postopek Rijndael, ki sta ga razvila belgijska raziskovalca Joan Daemen and Vincent Rijmen. Z rahlimi modifikacijami je bil leta 2001 sprejet kot standard AES.

V nadaljevanju je načelno predstavljen postopek šifriranja in dešifriranja AES, natančen opis algoritma pa je mogoče najti v standardu [5].

AES je simetričen blokovni algoritem z dolžino bloka 128 bitov in možnimi različnimi dolžinami ključa: 126, 192 in 256 bitov.

AES deluje nad 128 bitnimi bloki podatkov, ki predstavljajo stanja. Stanja so predstavljena obliki 4 x 4 matrik 8 bitnih zlogov, kot je to prikazano na sliki 3.14.

vhod

| | | | |
|---|---|---|---|
| $v_{00}$ | $v_{04}$ | $v_{08}$ | $v_{12}$ |
| $v_{01}$ | $v_{05}$ | $v_{09}$ | $v_{13}$ |
| $v_{02}$ | $v_{06}$ | $v_{10}$ | $v_{14}$ |
| $v_{03}$ | $v_{07}$ | $v_{11}$ | $v_{15}$ |

stanje

| | | | |
|---|---|---|---|
| $s_{0,0}$ | $s_{0,1}$ | $s_{0,2}$ | $s_{0,3}$ |
| $s_{1,0}$ | $s_{1,1}$ | $s_{1,2}$ | $s_{1,3}$ |
| $s_{2,0}$ | $s_{2,1}$ | $s_{2,2}$ | $s_{2,3}$ |
| $s_{3,0}$ | $s_{3,1}$ | $s_{3,2}$ | $s_{3,3}$ |

izhod

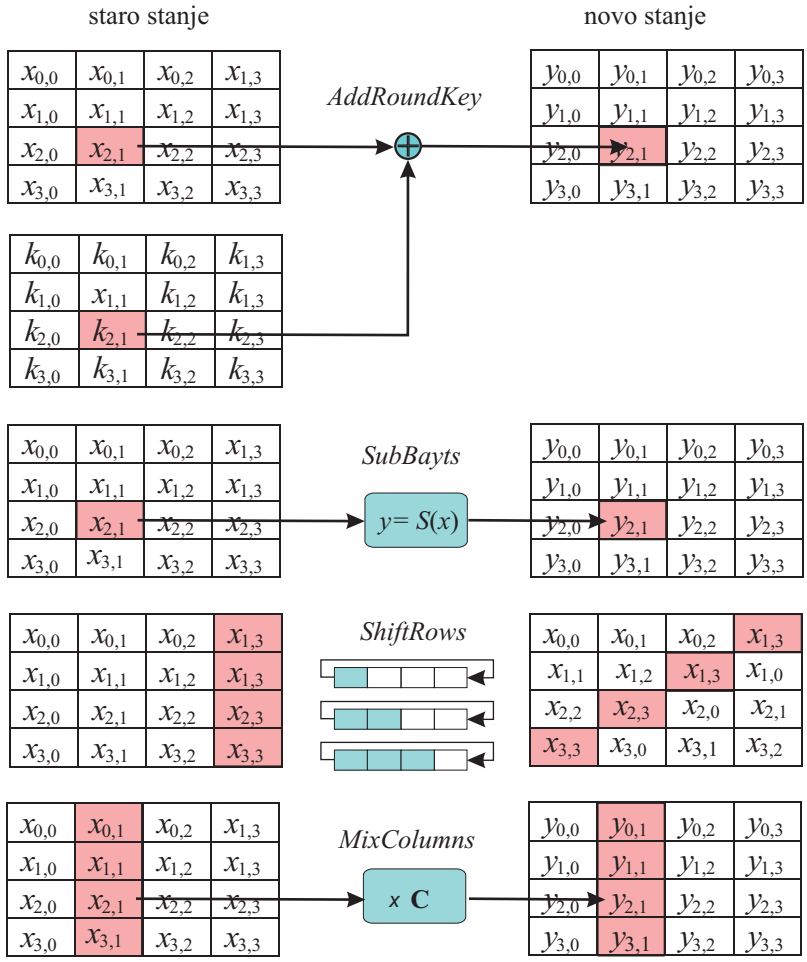
| | | | |
|---|---|---|---|
| $i_{00}$ | $i_{04}$ | $i_{08}$ | $i_{12}$ |
| $i_{01}$ | $i_{05}$ | $i_{09}$ | $i_{13}$ |
| $i_{02}$ | $i_{06}$ | $i_{10}$ | $i_{14}$ |
| $i_{03}$ | $i_{07}$ | $i_{11}$ | $i_{15}$ |

Slika 3.14 – Vhodni 128 bitni blok, 128 bitna vmesna stanja in izhodni 128 bitni blok so predstavljeni s 4 x 4 matrikami, ki vsebujejo 8 bitne zloge $v_n$, $s_{j,k}$ in $i_n$. Posamezen stolpec v matriki predstavlja 32 bitno besedo.

AES se izvaja kot zaporedje različnih funkcij nad stanji, ki se izvajajo v zaporednih ciklusih. Število ciklusov je odvisno od dolžine ključa in je enako 10 pri 128-bitnem ključu, 12 pri 192-bitnem ključu in 14 pri 256 bitnem ključu. V vsakem ciklusu se uporabi 128-bitni ključ ciklusa, ki je dobljen z razširjanjem osnovnega ključa po Rijndael algoritmu (ang. Rijndael key schedule). Nad stanji se izvajajo štiri različne funkcije:

- *AddRoundKey.* Prištevanje ključa ciklusa po modulu 2 (XOR).
- *SubBytes.* Zamenjava posameznih zlogov po neki nelinearni funkciji, ki je označena kot S škatla. Transformacija se lahko predstavi kot množenje z matriko in prištevanje konstantnega vektorja, lahko pa se uporabi substitucijska tabela.
- *ShiftRows.* Ciklični pomik vrstic matrike stanja v levo. Pomik je v vsaki vrstici drugačen in je enak zaporedni številki vrstice.
- *MixColumns.* Nad 32-bitno besedo, ki jo predstavlja en stolpec matrike stanja, se izvede reverzibilna linearna transformacija, množenja s konstanto po modulu.

Posamezne funkcije so ilustrirane na sliki 3.15.




Slika 3.15 – Funkcije, ki jih izvaja AES v posameznih ciklusih: *AddRoundKey, SubBytes, ShiftRows, MixColumns.*

Postopka šifriranja in dešifriranja potekata v naslednjih ciklusih:

- začetni ciklus
  - *AddRoundKey*

- vmesne cikluse
  - *SubBytes*
  - *ShiftRows*
  - *MixColumns*
  - *AddRoundKey*
- zaključni ciklus
  - *SubBytes*
  - *ShiftRows*
  - *AddRoundKey*

Kombinacijo operacij *SubBytes, ShiftRows* in *MixColumns* je mogoče izvesti tudi preko zaporedja vpogledov v štiri 256 vrstic dolge kodne tabele, kar lahko močno pospeši postopek šifriranja in dešifriranja.

Pravilno delovanje postopka je možno preveriti na osnovi znanih parov čistopis-šifropis in znanih ključev, ki jih je v ta namen objavi NIST. AES velja za dovolj močno šifro. S 256-bitnim ključem je odobrena njegova uporaba tudi za šifriranje tajnih dokumentov ameriške vojske.

## 3.5 Šifriranje z javnim ključem

Pri doslej obravnavanih simetričnih šifrah je izrednega pomena, da ostane ključ **K** tajen. Pri komunikaciji je zato zelo pomembna varna izmenjava ključev. Ker daje večkratna uporaba istega ključa napadalcem veliko večje možnosti za uspešen napad, je smiselno ključ pogosto menjavati, kar pa lahko, če ni na voljo nekega varnega kanala, težko izvedljivo.

Leta 1976 sta Whitfield Diffie in Martin Hellman iz univerze Stanford objavila članek "Nove smernice v kriptografiji" [9], ki je presenetil kriptografsko skupnost. V članku sta pokazala, da je možen praktično varen prenos šifriranih sporočil brez, da bi si obe strani pred tem izmenjali kakršno koli skrito informacijo.

Na prvi pogled se zdi to neizvedljivo, vendar je to možno tudi pri prenosu klasičnih sporočil. Vzemimo, da želi Alenka poslati Branetu[1] poslati pismo, vendar ne želi, da bi ga lahko kurir, ki ga prenaša prebral. Dokument zato zaklene v škatlo. To je primerljivo s klasičnim šifriranjem, saj mora imeti Brane enak ključ, da lahko škatlo odklene in prebere pismo. Ključ mora zato Alenka pred tem dati Branetu.

Vprašanje je, ali lahko Alenka pošlje zaklenjeno sporočilo Branetu, ne da bi imela enaka ključa. To je mogoče na naslednji način:

- Alenka da pismo v škatlo, ki jo je možno zakleniti z žabico.

[1] V literaturi o šifriranju nastopata običajno pri opisu poteka izmenjave sporočil Alice (A) in Bob (B). V tej knjigi smo v ta namen raje izbrali slovenska imena.

- Škatlo zaklene s svojo žabico.
- Kurir odnese škatlo Branetu. Ker je zaklenjena ne more prebrati pisma.
- Brane ne more odkleniti Alenkine žabice, ker nima ključa. Zato zaklene škatlo še s svojo žabico.
- Kurir odnese škatlo nazaj Alenki. Škatla je sedaj zaklenjena z dvema žabicama, zato tudi sedaj ne more prebrati pisma.
- Alenka s svojim ključem odklene svojo žabico in jo odstrani.
- Kurir odnese škatlo nazaj k Branetu. Škatla je sedaj zaklenjena z Branetovo žabico, zato kurir še vedno ne more prebrati pisma.
- Brane lahko sedaj odklene svojo žabico in prebere pismo.

Kurir je moral za tako izmenjavo opraviti trikratno pot, kar je pri izmenjavi klasičnih sporočil problematično, vendar pri izmenjavi elektronskih sporočil, to ne predstavlja večjega problema. Pri izmenjavi elektronskih sporočil pa je problem v tem, da bi v ta namen rabili šifro, s katero bi lahko dvakrat šifrirali sporočilo in ga potem dešifrirali v obratnem vrstnem redu. S tako šifro bi Alenka najprej šifrirala sporočilo s svojim ključem, ga poslala Borisu, ki bi šifrirano sporočilo še enkrat šifriral s svojim ključem. To sporočilo bi poslal nazaj Alenki, ki bi ga dešifrirala s svojim ključem in ga poslala Borisu. Boris bi sprejeto sporočilo dešifriral s svojim ključem in na ta način prišel do čistopisa..

Šifra za zgornji scenarij mora biti komutativna. Take šifre sicer obstajajo, vendar sta Diffie in Hellman rešila problem preprosteje tako, da sta za ustvarjanje skupnega šifrirnega ključa uporabila enosmerno funkcijo.

### 3.5.1 Enosmerne funkcije

V šifriranju se pojem enosmerna funkcija uporablja za funkcijo $f$, ki bijektivno preslika čistopis $C$ v šifropis $S$, tako, da je izračun:

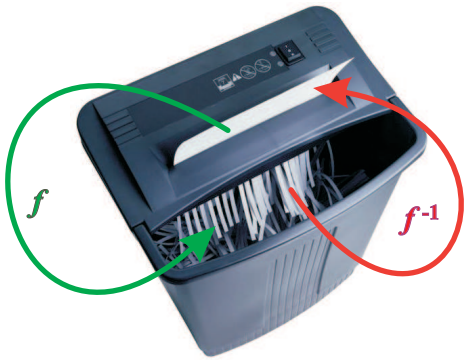
$$S = f(C) \tag{3.6}$$

relativno preprost, medtem ko predstavlja izračun inverzne funkcije $f^{-1}$, ki preslika šifropis nazaj v čistopis:

$$C = f^{-1}(S) \tag{3.7}$$

zahteven problem, ki ni rešljiv v omejenem času in z omejenimi sredstvi.

Za klasične dokumente predstavlja tako funkcijo rezalnik dokumentov. Dokument zlahka razrežemo na tanke pasove. Iskanje, zlaganje in lepljenje teh pasov nazaj v dokument pa je dokaj težek problem, kot je to ilustrirano na sliki 3.16.




Slika 3.16 – Ponazoritev enosmerne funkcije. Razrez papirja v trakove je preprosta operacija, medtem ko predstavlja sestavljanje dokumenta iz trakov težek problem. Zato se rezalniki uporabljajo za uničevanje dokumentov.

Diffie in Hellman sta kot enosmerno funkcijo predlagala diskretno eksponentno funkcijo nad končno množico $N$ števil med vključno 0 in $N-1$, oziroma nad končnim ali Galoisovim poljem (ang. Galois field) $GF(N)$.

$$S = a^C \bmod N \tag{3.8}$$

kjer je$a$ primitivni element končnega polja $GF(N)$. Element $a$ je primitiven element končnega polja $GF(N)$, če je z izrazom (3.8) mogoče izraziti vse nenične elemente polja. To pomeni, da z vsemi možnimi čistopisi $C$ dobimo tudi vse možne šifropise $S$. Čistopis $C$ in šifropis $S$ sta namreč tudi nenična elementa $GF(N)$.

Izračun $S$ po enačbi (3.8) je relativno preprost. Binarno sporočilo $C$ lahko predstavimo kot binarno zapisano število:

$$C = \sum_{m=0}^{n-1} c_m \, 2^m \tag{3.9}$$

kjer je $n$ dolžina sporočila in so $c_m \in \{0, 1\}$ posamezni biti sporočila od desne proti levi. Potem velja:

$$S = a^C = \prod_{m=0}^{n-1} K_m^{c_m} \bmod N \tag{3.10}$$

kjer so faktorji $K_m$ enaki:

$$K_m = a^{2^m} \tag{3.11}$$

in jih lahko izračunamo po rekurzivni enačbi:

$$K_{m+1} = K_m^2 \bmod N \tag{3.12}$$

z začetno vrednostjo $K_0 = a$.

V produktu (3.10) nastopajo samo tisti faktorji, pri katerih je $c_m$ enak torej potrebujemo za izračun največ $n-1$ množenj, kadar so vsi $C_m = 1$, drugače pa manj. Tudi za izračun faktorjev $K_m$ po enačbi (3.12) potrebujemo največ $n-1$ množenj. Kadar je $a$ vnaprej znan, lahko tudi faktorje $K_m$ izračunamo vnaprej, tako da potrebujemo za sprotni izračun samo $n-1$ množenj.

**Primer 3.3**

Za primer vzemimo $N = 29$, $a = 2$ in $C = 20_{10} = 10100_2$. Ker sta samo $c_2$ in $c_4$ različna od nič, potrebujemo za izračun samo faktorja:

$$\begin{aligned} K_2 &= 2^4 \bmod 29 = 16 \\ K_4 &= 2^1 6 \bmod 29 = 25 \end{aligned}$$

$$S = 2^{20} \bmod 29 = 16 \cdot 25 \bmod 29 = 23$$

Če so faktorji $K_m$ izračunani vnaprej, potrebujemo za ta izračun samo dve množenji.

Za izračun inverzne funkcije

$$C = \log_a S \bmod N \tag{3.13}$$

zaenkrat ne poznamo učinkovitega načina. Pri danem $S$ moramo poiskati $C$ za katerega velja enačba (3.8). Najuspešnejši algoritmi potrebujejo za to okrog $\sqrt{N}$ operacij. Za zgornji primer, kjer je $N$ majhen, to sicer ni veliko, pri zelo velikih $N$ pa postane tudi razmerje

$$\frac{\sqrt{N}}{n-1} > \frac{\sqrt{N}}{\log_2 N}$$

to je razmerje med zahtevnostjo izračuna diskretnega logaritma in diskretne eksponentne funkcije, zelo veliko.

### 3.5.2 Eksponentna izmenjava ključev

Diffie in Hellman sta diskretno eksponentno funkcijo uporabila kot enosmerno funkcijo za izmenjavo ključev preko ne-varnega kanala. Pri tej izmenjavi ustvarita Alenka in Boris neko skupno skrivnost, ki jo potem uporabita kot ključ za šifriranje sporočil s simetrično šifro. Postopek izmenjave poteka v naslednjih korakih:

- V prvem koraku se dogovorita za ustrezen $N$ in $a$. To sta dve števili, ki sta lahko tudi javno poznani. Primerne vrednosti $N$ in $a$ so lahko objavljene v javno dostopni bazi podatkov.

- Alenka nato izbere naključno število $X_A$; $1 < X_A < N$ in izračuna

  $$Y_A = a^{X_A} \bmod N$$

  ter ga pošlje Borisu.

- Boris izbere svoje naključno število $X_B$; $1 < X_B < N$ in izračuna

  $$Y_B = a^{X_B} \bmod N$$

  ter ga pošlje Alenki.

- Ker je diskretni eksponent komutativna funkcija lahko Alenka in Boris sedaj izračunata:

$$Y_{AB} = a^{X_A \cdot X_B} \bmod N = Y_A^{X_B} \bmod N = Y_B^{X_A} \bmod N$$

- $Y_{AB}$ lahko sedaj uporabita kot ključ za šifriranje sporočil z neko simetrično šifro, kot na primer z AES.

Alenka mora za komunikacijo z različnimi uporabniki izbrati samo eno število $X_A$, saj lahko to število uporabi tudi za dopisovanje s Cenetom, ki ustvari svoj $X_C$ in nato za šifriranje uporabita $Y_{AC}$. Število $X_A$ zato predstavlja Alenkin privatni ključ, ki ga ne pozna nihče drug. Število $Y_A$ pa je njen javni ključ, ki ga lahko objavi, tako, da je na voljo vsem, ki si želijo z njo tajno dopisovati.

**Primer 3.4**

Naj bo $N = 29$ in $a = 2$.
Alenka izbere naključen ključ $X_A = 23$ in izračuna

$$Y_A = 2^{23} \bmod 29 = 10$$

Boris izbere $X_B = 11$ in izračuna

$$Y_B = 2^{11} \bmod 29 = 18$$

Alenka izračuna $Y_{AB}$ po enačbi

$$Y_{AB} = Y_B^{X_A} \bmod N = 18^{23} \bmod 29 = 2$$

Boris izračuna $Y_{AB}$ po enačbi

$$Y_{AB} = Y_A^{X_B} \bmod N = 10^{11} \bmod 29 = 2$$

Sedaj oba poznata $Y_{AB} = 2$ in to uporabita za ključ simetrične šifre.

### 3.5.3 Asimetrične šifre

Asimetrične šifre uporabljajo dva ključa: ključ za šifriranje $\mathbf{K}_s$ in ključ za dešifriranje $\mathbf{K}_d$. Pomemben pogoj za asimetrične šifrirne postopke je, da se iz ključa za šifriranje $\mathbf{K}_s$ ni mogoče pridobiti ključa za dešifriranje $\mathbf{K}_d$, oziroma mora to predstavljati računsko zahteven problem, ki ni rešljiv v nekem omejenem času oziroma z omejenimi sredstvi.

Ker s ključem za šifriranje $\mathbf{K}_s$ ni mogoče dešifrirati dokumenta, je ta lahko javno objavljen, zato se imenuje tudi javni ključ. Ključ za dešifriranje $\mathbf{K}_d$ pa mora ostati skrit, zato se imenuje tudi tajni ključ.

Za izvedbo asimetrične šifre potrebujemo enosmerno funkcijo s stranskim vhodom (ang. one-way trapdoor function). Enosmerna funkcija s stranskim vhodom $f_z$ je funkcija za katero je

relativno preprosto izračunati $S = f_z(C)$, izračun inverzne funkcije $C = f_z^{-1}(S)$ pa je praktično neizvedljiv, če ne poznamo $z$, če pa $z$ poznamo, pa je ta izračun relativno preprost. Vrednost $z$zato predstavlja stranski vhod, skozi katerega lahko pridemo do šifriranega sporočila.

V svetu klasičnih dokumentov predstavlja enosmerno funkcijo s stranskim vhodom poštni nabiralnik, kot je to ilustrirano na sliki 3.17.


Slika 3.17 – Poštni nabiralnik predstavlja je enosmerna funkcija s stranskim vhodom. Vsakdo lahko vrže pismo v nabiralnik, iz nabiralnika pa ga lahko vzame samo, kdor ima ključ od vrat nabiralnika

Šifriranje na osnovi enosmerne funkcije s stranskimi vrati sta si zamislila že Diffie in Hellman, vendar take funkcije nista poznala niti nista vedela ali takšna funkcija sploh obstaja. Prvo šifro na osnovi enosmerne funkcije s stranskimi vrati so predlagali Ronald Linn Rivest, Adi Shamir in Leonard Max Adleman eta 1978. Po začetnicah njihovih priimkov se ta imenuje šifra RSA.

### 3.5.4 Šifra RSA

Za šifriranje in dešifriranje je pri RSA uporabljena diskretna eksponentna funkcija nad končnim poljem $GF(N)$. Šifriranje poteka po enačbi:

$$S = C^E \bmod N \tag{3.14}$$

kjer sta $E$ in $N$ par števil, ki skupaj tvorita javni ključ:

$$\mathbf{K}_s = (N, E) \tag{3.15}$$

Dešifriranje se izvaja po enačbi:

$$C = S^{K_D} \bmod N \tag{3.16}$$

Če želi Alenka poslati Borisu šifrirano sporočilo, mora Boris najprej ustvariti par ključev $\mathbf{K}_s$ in $K_d$. Ključ $\mathbf{K}_s$ pošlje Alenki, oziroma ga lahko javno objavi za vse, ki bi mu želeli pošiljati šifrirana sporočila. Ključ $K_d$ mora obdrži zase kot svoj privatni ključ, kajti samo s tem ključem je mogoče dešifrirati sporočila šifrirana s ključem $\mathbf{K}_s$. Ključa ustvari po naslednjem postopku:

- izbere dve veliki naključni praštevili $p$ in $q$

- izračuna
$$N = p\,q,$$
- izračuna
$$\phi = (p-1)(q-1),$$
- izbere naravno število $E$ $(1 < E < \phi)$ ki je tuje $\phi$, kar pomeni, da je največji skupni delitelj števil $E$ in $\phi$ enak 1
$$\mathrm{NSD}(E, \phi) = 1,$$
- ustvari javni ključ
$$\mathbf{K}_s = (N, E),$$
- izračuna privatni ključ, $K_d$, ki je enak inverzni vrednosti $E^{-1}$ po modulu $\phi$, tako da reši enačbo
$$K_d\ E \bmod \phi = 1$$

Dokaz, da z dešifriranjem po enačbi (3.16) s ključem $K_d$, ki ga dobimo po zgoraj opisanem načinu, res dobimo nazaj originalen čistopis presega okvire te knjige, bralec pa ga lahko najde v tako v originalnem članku [10] kot tudi v mnogih drugih knjiga o kriptografiji.

Kdor pozna javni ključ $\mathbf{K}_s$, to je $N$ in $E$, lahko s pomočjo enačbe (3.16) sporočilo šifrira, vendar ga ne more dešifrirati, če ne pozna ključa $K_d$. Za izračun ključa $K_d$ bi moral poznati $\phi$, da pa bi lahko izračunal $phi$, bi moral poznati praštevili $p$ in $q$.

Napadalec pozna zgolj $N$, to je produkt števil $p$ in $q$. Da bi dobil $p$ in $q$, bi moral faktorizirati $N$. Faktorizacija produkta dveh velikih praštevil je zahteven računski problem. Kljub temu pa je faktorizacija bistveno manj zahtevna kot preizkušanje vseh ključev pri napadu grobe sile, zato morajo biti ključi pri RSA bistveno daljši kot pri blokovnih šifrah.

### Določanje $p$ in $q$

V prvem koraku pri ustvarjanju ključev za RSA potrebujemo dve veliki praštevili $p$ in $q$.

Če želimo ustvariti 2.048-bitni ključ morata biti praštevili $p$ in $q$ 1.024 bitni. Ker ne poznamo nobenega algoritma, s pomočjo katerega bi lahko izračunali neko naključno praštevilo, moramo ti dve števili poiskati. Iskanja se lotimo tako, da izberemo naključno število ustrezne dolžine in preizkusimo, ali je praštevilo.

Število praštevil, ki so enaka ali manjša številu $n$ je enako $\pi(n)$. Funkcijo $pi(n)$ lahko približno računamo po enačbi:
$$\pi(n) = \frac{n}{\ln n} \tag{3.17}$$
Verjetnost, $p_p(n)$ da bo naključno izbrano število $n$ praštevilo je potem približno enaka:
$$p_p(n) = \frac{\pi(n)}{n} \doteq \frac{1}{\ln(n)} \tag{3.18}$$

Za $m$-bitno število $n$ je manjše $2^m$, zato lahko verjetnost, da je naključno $m$-bitno število tudi praštevilo, ocenimo z neenačbo:

$$p_p(2^m) \geq \frac{1}{\ln 2^m} = \frac{1}{m \ln 2} = \frac{1}{0,693\, m} \tag{3.19}$$

Približno vsako $0,693\, m$ število bo torej praštevilo. Ker vemo, da soda števila niso praštevila, lahko izbiramo samo liha števila. V povprečju bomo za $m$-bitno število potrebovali $0,312m$ poskusov, da najdemo praštevilo, kar je 327 poskusov, za vsako od obeh praštevil 2.048-bitnega ključa.

Ali je naključno izbrano število res praštevilo, lahko uporabimo Fermajev teorem, ki pravi, da za vsako praštevilo $p$ in bazo $1 < b < p$ velja:

$$b^{p-1} \bmod p = 1 \tag{3.20}$$

Ko naključno izberemo število $n$, preizkusimo ali za nek naključno izbran $b$ velja:

$$b^{n-1} \bmod n = 1 \tag{3.21}$$

Če pogoj ni izpolnjen $n$ zagotovo ni praštevilo, če pa je, je to kandidat za praštevilo. Kandidata potem preizkusimo z različnimi bazami $b$. Verjetnost, da bi neko število izpolnjevalo pogoj za $k$ različnih baz je manjša od $2^{-k}$. Če neko število izpolnjuje pogoj za npr. 100 različnih baz, lahko skoraj z gotovostjo trdimo, da je to praštevilo. Poleg tega testa obstajajo še mnogi drugi testi, s katerimi lahko še z vejo gotovostjo potrdimo, da je izbrano število praštevilo.

Če prvo izbrano liho število ne izpolnjuje pogoja, lahko število preprosto povečamo za 2 in ponovimo test. Postopek nadaljujemo, dokler ne najdemo praštevila.

**Primer 3.5**

Za primer si oglejmo, števili 5 in 6.

$$\begin{aligned} 2^4 \bmod 5 &= 16 \bmod 5 = 1 \\ 3^4 \bmod 5 &= 81 \bmod 5 = 1 \\ 4^4 \bmod 5 &= 256 \bmod 5 = 1 \end{aligned}$$

Ker je število 5 je praštevilo, izpolnjuje Fermajev pogoj za vse vrednosti $b$ na intervalu $1 < b < 5$.

$$\begin{aligned} 2^5 \bmod 6 &= 32 \bmod 6 = 2 \\ 3^5 \bmod 6 &= 243 \bmod 6 = 3 \\ 4^5 \bmod 6 &= 1024 \bmod 6 = 4 \\ 5^5 \bmod 6 &= 3125 \bmod 6 = 5 \end{aligned}$$

Število 6 ne izpolnjuje Fermajevega pogoja za noben $b$ na intervalu $1 < b < 6$. Da bi ugotovili, da 6 ni praštevilo, bi zadoščalo že, da pogoja ne izpolnjuje pri eni vrednosti $b$, na primer pri $b = 2$.

**Izbira $E$**

Načeloma je lahko $E$ tudi zelo majhno število, ki je tuje $\phi$, vendar taka izbira ni najprimernejša, saj se lahko zgodi de je $C^E$ manjše od $N$. Tako šifriran tekst lahko potem napadalec dešifrira preprosto tako da izračuna $E$-ti koren šifropisa $S$.

Podobno kot $p$ in $q$ tudi $E$ izberemo naključno in preverimo, ali je izpolnjen pogoj:

$$\mathrm{NSD}(E, \phi) = 1 \tag{3.22}$$

To lahko storimo tako, da izračunamo $\mathrm{NSD}(E, \phi)$ s pomočjo Evklidovega algoritma, v skladu z rekurzivno enačbo:

$$r_{n-2} = q_n \, r_{n-1} + r_n; \tag{3.23}$$

kjer je $q_n$ količnik in $r_n$ ostanek deljenja v $n$-ti iteraciji. Za začetni vrednosti postavimo:

$$\begin{aligned} r_{-1} &= \phi \\ r_0 &= E \end{aligned} \tag{3.24}$$

in konča, ko je ostanek enak nič. Zadnji od nič različni ostanek je največji skupni delitelj števil $\phi$ in $E$, ki mora biti enak 1, da je izbrano število $E$ primerno.

**Primer 3.6**

Naj bo $\phi = 168$ in $E = 63$. Iteracije Evklidovega algoritma so potem:

$$\begin{aligned} 168 &= 2 \cdot 63 + 42 \\ 63 &= 1 \cdot 42 + 21 \\ 42 &= 2 \cdot 21 + 0 \end{aligned}$$

Zadnji nenični ostanek je 21 zato je največji skupni delitelj števil 168 in 63 enak 21, zato izbira $E = 63$ ni primerna pri $\phi = 168$.

**Računanje $K_d$**

Privatni ključ $K_d$ je inverz števila $E$ po modulu $\phi$, kar pomeni, da moramo najti $k_d$, ki izpolnjuje pogoj:

$$K_d \cdot E \bmod \phi = 1 \tag{3.25}$$

Zgornjo enačbo lahko zapišemo tudi v obliki:

$$K_d \cdot E = 1 - a \, \phi \tag{3.26}$$

oziroma

$$K_d \cdot E + a \, \phi = 1 \tag{3.27}$$

kjer je $a$ neko celo število. Zgornjo enačbo lahko rešimo s pomočjo razširjenega Evklidovega algoritma. Hkrati, ko z Evklidovim algoritmom ugotavljamo, ali je za izbrano vrednost $E$ velja:

$$\mathrm{NSD}(E, \phi) = 1$$

lahko z razširjenim Evklidovim algoritmom računamo tudi zaporedne iteracije ključa za dešifriranje $K_{d,n}$, po enačbi:

$$K_{d,n} = (K_{d,n-2} - K_{d,n-1}\ q_{n-2}) \bmod \phi \tag{3.28}$$

z začetnimi vrednostmi:

$$\begin{aligned} K_{d,-1} &= 0 \\ K_{d,0} &= 1 \end{aligned} \tag{3.29}$$

Če je preizkušani $E$ tuj $\phi$, potem pridemo v Evklidovem algoritmu v neki iteraciji $n = k$ do ostanka $r_k = 1$, hkrati pa smo izračunali tudi:

$$K_d = K_{d,k} = E^{-1} \bmod \phi \tag{3.30}$$

**Primer 3.7**

naj bo $\phi = 168$ in $E = 65$. Z razširjenim Evklidovim algoritmom preizkusimo ustreznost izbire $E$ in, če je ta ustrezna, hkrati izračunajmo $K_D$. Zato moramo rekurzivno izvesti enačbi (3.28) in (3.23). Dobimo:

| $n$ | $r_{n-2}$ | = | $q_n$ | $r_{n-1}$ | + | $r_n$ | $K_{d,n}$ | = | ( | $K_{d,n-2}$ | − | $K_{d,n-1}$ | $q_n$ | ) | mod | $\phi$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 168 | = | 2 | 65 | + | 38 | 166 | = | ( | 0 | − | 1 | 2 | ) | mod | 168 |
| 2 | 65 | = | 1 | 38 | + | 27 | 3 | = | ( | 1 | − | 166 | 1 | ) | mod | 168 |
| 3 | 38 | = | 1 | 27 | + | 11 | 163 | = | ( | 166 | − | 3 | 1 | ) | mod | 168 |
| 4 | 27 | = | 2 | 11 | + | 5 | 13 | = | ( | 3 | − | 163 | 2 | ) | mod | 168 |
| 5 | 11 | = | 2 | 5 | + | 1 | 137 | = | ( | 163 | − | 13 | 2 | ) | mod | 168 |

V petem koraku smo dobili ostanek $r_5 = 1$, zato vemo, da je $\mathrm{NSD}(168, 65)$ enak 1 in da je $E = 65$ primerna izbira pri $phi = 168$. Hkrati z izbiro Ustreznega $E$ pa smo izračunali tudi ključ za dešifriranje $K_d = K_{d,5} = 163$.

Problem težavnosti faktorizacije, na katerem sloni varnost RSA, je izpostavil že W. S. Jevons v svoji knjigi "Principi znanosti", ki je izšla leta 1873. V knjigi je zapisal, da je on verjetno edini človek na svetu, ki bo kadarkoli vedel, kateri dve števili je potrebno možiti, da dobimo število 8.616.460.799. Že leta 1907 pa je D.H. Lahmer uspel izvesti faktorizacijo tega števila in objavil števili 8.9681 in 9.6079, ki sta prafaktorja omenjenega števila. Danes lahko vsakdo izvede to faktorizacijo v manj kot eni sekundi na osebnem računalniku.

Domneva, da je faktorizacija zahteven računski problem in, da je za produkt dveh velikih praštevil računsko neizvedljiva, je zelo verjetno pravilna, vendar pa tega še nihče ni uspel dokazati. Zato obstaja možnost, da bo nekdo v prihodnosti odkril bistveno bolj učinkovit postopek faktorizacije in s tem razbil RSA postopek šifriranja. Ravno tako je možno, da se bodo pojavili še bistveno hitrejši računalniki in RSA ključi, ki se uporabljajo danes ne bodo več varni.

Za primer si oglejmo še, kako bi potekal celoten postopek izmenjave ključa s pomočjo RSA šifre.

**Primer 3.8**

Vzemimo, da želi Alenka poslati Borisu šifrirni ključ simetrične šifre $K = 645$, s katerim si bosta potem šifrirala medsebojna sporočila.

Boris mora najprej ustvariti par ključev. Zato najprej izbere dve praštevili $p = 47$ in $q = 59$ in izračuna

$$N = p\,q = 2773$$

in

$$\phi(N) = (p-1)(q-1) = 2668$$

Nato izbere še število $E$, ki je tuje številu $phi$, na primer število $E = 17$, in ustvari javni ključ

$$K_s = (2773, 17)$$

ki ga pošlje Alenki. Alenka sedaj lahko šifrira sporočilo:

$$S = C^E \bmod N = 645^{17} \bmod 2773 = 1802$$

in ga pošlje Borisu. S svojim privatnim ključem za dešifriranje

$$K_d = E^{-1} \bmod \phi(N) = 157$$

lahko Boris dešifrira sprejeto sporočilo

$$C = S^{K_D} \bmod N = 740^{157} \bmod 2773 = 645$$

in s tem pridobi ključ $K = 654$ za simetrično šifro, s pomočjo katere si potem lahko z Alenko izmenjuje zasebna sporočila. Par ključev $K_s$ in $K_d$ lahko uporabi tudi za izmenjavo ključev z drugimi uporabniki. Svoj ključ za šifriranje $K_s$ lahko tudi javno objavi in na ta način vsakomur omogoči, da mu pošlje šifrirano sporočilo.

### 3.5.5 Eliptične krivulje

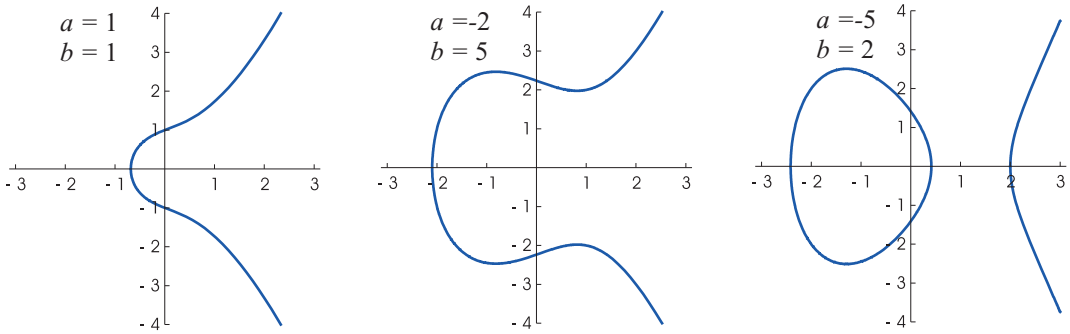
Šifriranje na osnovi eliptičnih krivulj (ECC – Elliptic Curve Cryptography) sta neodvisno predlagala Neal Koblitz in Victor S. Miller že leta 1985, vendar je prišlo v širšo uporabo šele leta 2004, leta 2006 pa ge je NIST sprejel kot standard.

Matematika pri eliptičnih krivuljah je nekoliko zahtevnejša kot matematika nad končnimi polji $GF(N)$, na kateri sloni eksponentna izmenjava ključev kot tudi asimetrična šifra RSA. Zato si bomo tu ogledali zgolj osnove eliptičnih krivulj in se ne bomo spuščali v podrobnosti.

V šifriranju se lahko uporabljajo različne eliptične krivulje. Pogosto se uporabljajo tako imenovane Weierstrassove eliptične krivulje, ki jih bomo obravnavali v nadaljevanju in jih opisuje enačba:

$$y^2 = x^3 + ax + b \tag{3.31}$$

Z izbiro koeficientov $a$ in $b$ dobimo različne krivulje iz družine, kot je to prikazano na sliki 3.18. Ker je na levi strani enačbe $y^2$ so krivulje simetrične okoli osi $x$.




Slika 3.18 – Weierstrassove eliptične krivulje pri različnih izbirah parametrov $a$ in $b$.

Pri šifriranju na osnovi eliptičnih krivulj računamo s točkami na izbrani krivulji. Označimo z $x_n$ in $y_n$ koordinate točke $\mathbf{T}_n$, ki leži na eliptični krivulji.

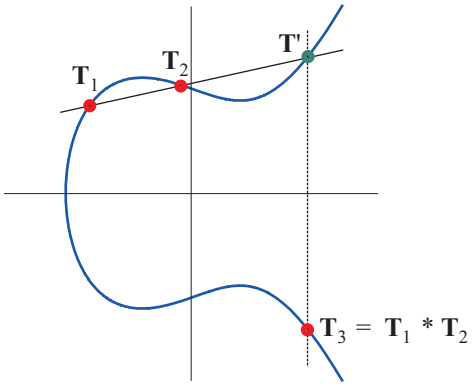
$$\mathbf{T}_n = (x_n, y_n) \tag{3.32}$$

Za namen šifriranja potrebujemo operacijo množenja dveh točk:

$$\mathbf{T}_3 = \mathbf{T}_1 * \mathbf{T}_2 \tag{3.33}$$

Operacijo najlaže predstavimo geometrično. Pri tem upoštevamo dejstvo, da premica, ki seka eliptično krivuljo v dveh točkah $\mathbf{T}_1$ in $\mathbf{T}_2$ nujno seka točko tudi v tretji točki $\mathbf{T}'$. Produkt dveh točk $\mathbf{T}_3$ je definiran kot zrcalna slika točke $\mathbf{T}'$, kot je to prikazano na sliki 3.19.

Kadar točko množimo samo s seboj, ne moremo potegniti premice skozi dve točki, da bi dobili tretjo točko. Vendar, če bi točko $\mathbf{T}_2$ približevali točki $\mathbf{T}_1$, bi se naklon premice, ki gre skozi obe točki približeval tangenti skozi točko $\mathbf{T}_1$. Produkt točke s samo sabo ali tudi funkcijo kvadriranja torej predstavimo na enak način, kot produkt dveh točk, le da je tu točka $\mathbf{T}'$ sečišče tangente z eliptično krivuljo.

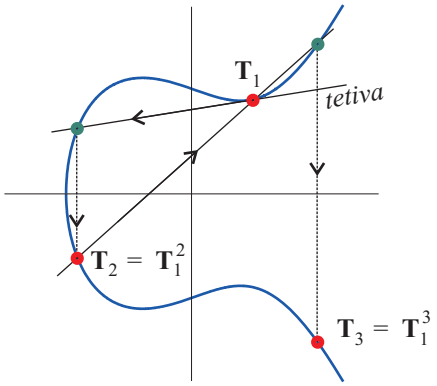



Slika 3.19 – Grafični prikaz produkta $\mathbf{T}_3 = \mathbf{T}_1 * \mathbf{T}_2$. Produkt $\mathbf{T}_3$ je točka, ki je zrcalna točki $\mathbf{T}'$

Na ta način lahko definiramo tudi celo potenco večkratni produkt točke same s sabo. Četrta potenca točke $\mathbf{T}_1$ bi bila torej enaka:

$$\mathbf{T}_3^3 = \mathbf{T}_1^3 = (\mathbf{T}_1 * \mathbf{T}_1) * \mathbf{T}_1 \tag{3.34}$$

kot je to prikazano na sliki 3.20.




Slika 3.20 – Potenciranje točke na eliptični krivulji $\mathbf{T}_3 = \mathbf{T}_1^3$

Množenja izvajamo od leve proti desni tako, da izračunamo vmesne točke:

$$\begin{aligned} \mathbf{T}_2 &= \mathbf{T}_1 * \mathbf{T}_1 \\ \mathbf{T}_3 &= \mathbf{T}_2 * \mathbf{T}_1 \end{aligned} \tag{3.35}$$

Z malo poznavanja geometrije bi sedaj lahko izpeljali tudi enačbe za izračun produkta dveh točk kot tudi enačbe za izračun množenja točke same s seboj oziroma izračun kvadrata točke.

Zaenkrat smo predstavili eliptične krivulje v obsegu realnih števil, ki pa za namen šifriranja niso zanimive. V šifriranju se namreč uporabljajo krivulje nad končnimi polji $GF(N)$, kjer je $N$ praštevilo ali pa cela potenca števila dva ($N = 2^M$).

Vse točke na eliptični krivulji, ki zadoščajo enačbi:

$$y^2 \bmod N = (x^3 + ax + b) \bmod N \tag{3.36}$$

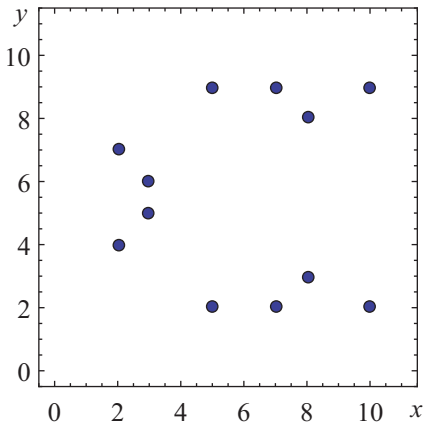
kjer so $y$, $x$, $a$ in $b$ elementi končnega polja, z dodatkom nevtralnega elementa $O$, tvorijo multiplikativno grupo $EC(a, b, N)$, v kateri je operacija množenja definirana na enak način kot na eliptični krivulji, le da se vse operacije izvajajo nad elementi končnega $GF(N)$.

**Primer 3.9**

Za primer eliptične krivulje $a = 1$, $b = 6$ in $N = 11$ je enačba (3.36) izpolnjena za 12 točk, ki jim je potrebno pridružiti še nevtralni element $O$, da tvorijo grupo $EC(1, 6, 11)$:

$$EC(1, 6, 11) = \{(2, 4), (2, 7), (3, 5), (3, 6), (5, 2), (5, 9), (7, 2), (7, 9), (8, 3), (8, 8), (10, 2), (10, 9), O\} \tag{3.37}$$

Točke v grupi $EC(1, 6, 11)$ so prikazane na sliki 3.21.




Slika 3.21 – Točke v multiplikativni grupi $E(1, 5, 11)$ na eliptični krivulji $a = 1$, $b = 5$ nad $GF(11)$.

Eksplicitne enačbe za množenje dveh točk in za množenje točke samo s seboj so enake kot enačbe za eliptične krivulje nad obsegom realnih števil, le da se tu izvajajo nad poljem $GF(N)$. Deljenje se tu izvede kot množenje z inverznim elementom po modulu $N$.

Enačbe za množenje najprej izpeljemo za eliptično krivuljo nad obsegom realnih števil in jih nato zapišemo z operacijami po modulu $N$.

Za izračun produkta $\mathbf{T}_1 = (x_3, y_3)$ točk $\mathbf{T}_1 = (x_1, y_1)$ in $\mathbf{T}_2 = (x_2, y_2)$ moramo najprej izračunati naklon premice, ki gre skoti ti dve točki.

$$\lambda = (y_2 - y_1)(x_2 - x_1)^{-1} \pmod N \tag{3.38}$$

Kadar je $\mathbf{T}_2 = \mathbf{T}_1$ množimo točko $\mathbf{T}_1$ samo s seboj. Naklon enak naklonu tangente na krivuljo skozi točko $\mathbf{T}_1$:

$$\lambda = (3x_1^2 + a)(2y)^{-1} \pmod N \tag{3.39}$$

Nato določimo premico z naklonom $\lambda$, ki gre skozi točko $\mathbf{T}_1$, in poiščemo njeno sečišče z eliptično krivuljo. Dobimo točko $\mathbf{T}'$, ki jo moramo zrcaliti preko osi $x$, da dobimo točko $\mathbf{T}_3 = \mathbf{T}_1 * \mathbf{T}_2$.

Koordinati $x_3$ in $y_3$ izračunamo po enačbi:

$$\begin{aligned} x_3 &= \lambda^2 - x_1 - x_2 \pmod N \\ y_3 &= \lambda(x_1 - x_3) - y_1 \pmod N \end{aligned} \tag{3.40}$$

**Primer 3.10**

Kot primer si oglejmo množenje dveh točk $\mathbf{T}_1 = (2, 4)$ in $\mathbf{T}_2 = (5, 9)$ na eliptični krivulji nad $EC((1, 6, 11)$ iz primera 3.9.

Najprej izračunamo $\lambda$;

$$\lambda = (9 - 4)(5 - 2)^{-1} = 5 \cdot 3^{-1} = 5 \cdot 4 = 9 \pmod{11}$$

in nato še $x_3$ in $y_3$

$$\begin{aligned} x_3 &= 9^2 - 2 - 5 = 8 \pmod{11} \\ y_3 &= 9(2 - 8) - 4 = 9 \cdot 5 - 4 = 8 \pmod{11} \end{aligned}$$

Rezultat je točka $\mathbf{T}_3 = (8, 8)$, ki je tudi točka v $EC((1, 6, 11)$.

**Primer 3.11**

Oglejmo si še četrto potenco točke $(2, 4)$ v $EC(1, 6, 11)$. Najprej točko množimo samo s seboj:

$$(2, 4)^2 = (2, 4) * (2, 4) = (5, 9)$$

Rezultat ponovno množimo s točko $(2, 4)$

$$(2, 4)^3 = (5, 9) * (2, 4) = (8, 8)$$

in končno:

$$(2, 4)^4 = (5, 9) * (8, 8) = (10, 9)$$

Do istega rezultata bi z manj množenji prišli, če bi točko, ki $(5, 9)$, ki smo jo dobili v drugem koraku še enkrat množili samo s seboj:

$$(2, 4)^4 = (5, 9) * (5, 9) = (10, 9)$$

kar je krajši način za računanje potence, saj smo namesto treh množenj potrebovali le dve.

V zadnjem primeru smo pokazali, da je operacija množenja asociativna in lahko diskretno eksponentno funkcijo v $EC(a, b, N)$ računamo po skrajšanem postopku, na enak način, kot smo računali diskretno eksponentno funkcijo nad $GF(N)$. Diskretni logaritem pa je nad $EC(a, b, N)$ še bistveno zahtevnejši problem kot v $GF(N)$.

Diskretna eksponentna funkcija nad eliptičnimi krivuljami predstavlja enosmerno funkcijo. Zato lahko nad eliptičnimi krivuljami z majhnimi spremembami izvedemo vse šifre, ki slone na diskretnem algoritmu nad končnim poljem $GF(N)$. Za ilustracijo si oglejmo zgolj eksponentno izmenjavo ključev.

**Eksponentna izmenjava ključev nad eliptičnimi krivuljami**

Pri tej izmenjavi je uporabljena eliptična krivulja, ki je javno znana. Javno znan je tudi generatorska točka $\mathbf{G} = (x_g, y_g)$. Generatorska točka je primitivni element grupe, kar pomeni, da je možno z množenjem te točke samo s sabo dobiti vse točke v grupi.

Običajno se za izmenjavo uporabljena neka od standardnih krivulj, ki imajo tudi svoja imena, njihovi parametri, vključno z generatorsko točko, pa so javno objavljeni. Standardne krivulje imajo tudi svoja imena, po katerih se lahko nanje tudi sklicujemo.

Izmenjava potem poteka po naslednjem postopku:

- Alenka nato izbere naključno število $X_A$; $1 < X_A < N$. To je njen privatni ključ. Nato izračuna točko
$$\mathbf{Y}_A = \mathbf{G}^{X_A}$$
ki predstavlja njen javni ključ in ga pošlje Borisu.

- Boris izbere svoje privatni ključ $X_B$; $1 < X_B < N$ in nato izračuna svoj javni ključ
$$\mathbf{Y}_B = \mathbf{G}^{X_B}$$
ter ga pošlje Alenki.

- Ker je diskretni eksponent nad eliptičnimi krivuljami komutativna funkcija lahko Alenka in Boris izračunata:
$$\mathbf{Y}_{AB} = \mathbf{G}^{X_A \cdot X_B} = \mathbf{Y}_A^{X_B} = \mathbf{Y}_B^{X_A}$$

- Iz koordinat $\mathbf{Y}_{AB}$ lahko sedaj z neko dogovorjeno funkcijo izračunata ključ $\mathbf{K}$ neke simetrične šifre, kot na primer z AES.

Ker smo v primerih eliptičnih krivulj obravnavali krivulje z zelo majhnimi vrednostmi $a$, $b$ in $N$, si za občutek oglejmo parametre eliptične krivulje p256, ki jo uporablja Goggle za varen dostop do svojih spletnih strani. Pri tej krivulji je $N$ 256-bitno praštevilo, njeni parametri pa so:

$$\begin{aligned}
N &= 115792089210356248762697446949407573530086143415290314195533631308867097853951\\
a &= 115792089210356248762697446949407573530086143415290314195533631308867097853948\\
b &= 41058363725152142129326129780047268409114441015993725554835256314039467401291\\
x_g &= 48439561293906451759052585252797914202762949526041747995844080717082404635286\\
y_g &= 36134250956749795798585127919587881956611106672985015071877198253568414405109
\end{aligned}$$

Omenimo tudi, da obstaja standardna krivulja p512 s 512 bitnimi parametri. Vidimo, da pri taki dolžini parametrov tudi množenja po modulu niso čisto preproste operacije. Množenja po modulu je bistveno laže izvajati, kadar je $N = 2^M$ cela potenca števila dva, saj je pri množenju potrebno izračunati samo $M$ najmanj pomembnih bitov, ki predstavljajo ostanek po modulu $N$.

### 3.5.6 Druge asimetrične šifre

#### Reševanje ugank

Prvo asimetrično šifro je predlagal Ralph Merkle leta 1974. Šifra je zasnovana na reševanju velikega števila ugank. Uganko v tem primeru predstavlja dešifriranje šifropisa, ne da bi poznali ključ.

Alenka pošle Borisu $N$ (okrog en milijon) različnih ključev $K_n$ zapisanih v neki standardni obliki. Vsakega od teh ključev šifrira z drugim, mnogo krajšim ključem $k_n$, to je ključem, ki ima le $N$ možnih vrednosti, tako da omogoča napad grobe sile, to je napad s preizkušanjem vseh možnih ključev. Boris naključno izbere an primer $m$-ti šifropis in ga dešifrira s preizkušanjem vseh možnih vrednosti ključa $k_m$. Ker je ključ $k_m$ relativno kratek, to ne predstavlja neizvedljive naloge. Tako pride do ključa $K_m$, ki ga bo uporabljal za izmenjavo sporočil, vendar mora Alenki prej sporočiti, katerega od prejetih šifropisov je izbral. Zato s ključem $K_m$, s katerim je uspel dešifrirati izbrani šifropis, šifrira neko vnaprej dogovorjeno sporočilo in ga pošlje Alenki. Alenka poskuša sprejeto dešifrirati to sporočilo z vsemi svojimi ključi $K_n$. Ko poskusi s ključem $K_m$, dobi kot rezultat dogovorjeno sporočilo in zato ve, da je Boris izbral ta ključ.

Za izmenjavo ključa morata tako Boris kot Alenka povprečno izvesti približno $N/2$, ker v povprečju prideta do rezultata po polovici preizkušenih ključev. Napadalec, ki ne ve, kateri ključ je izbral Boris in obenem ne pozna vseh $N$ ključev $K_n$, mora za preizkušanje v povprečju opraviti $N^2/4$ dešifriranj. Pri $N = 2 \cdot 10^6$ bi morala Alenka in Boris opraviti vsak po $10^6$ dešifriranj, napadalec ba bi moral opraviti $10^{12}$ dešifriranj, kar je milijonkrat več, kot Boris in Alenka.

Opisani postopek zaradi potrebe po pošiljanju velikega števila šifriranih ključev in velike računske zahtevnosti izmenjave ključa, postopek ni prišel v praktično uporabo, je pa zagotovo zanimiv iz teoretičnega vidika. Razmerje med razmerje med računsko zahtevnostjo, ki jo ima napadalec in računsko zahtevnostjo, ki jo imata Alenka in Boris je tu reda $O(N)$, medtem, ko je pri drugih znanih šifrah reda $O(N/\ln N$, vendar je lahko pri teh $N$ bistveno večji, zato je tudi to razmerje bistveno večje.

#### Nahrbtnik s stranskimi vrati

Tudi to šifro je predlagal Ralph Merkle. Šifra je izpeljana iz problema zlaganja različnih stvari v nahrbtnik, tako, da popolnoma zapolnimo ves prostor v nahrbtniku.

Najpreprostejši primer takega zlaganja, ki ga je uporabil Merkle kot enosmerno funkcijo, je enodimenzionalni problem, to je problem zlaganja palic različnih dolžin v tanko cev. Če imamo na voljo množico palic različnih dolžin, in izberemo neko njihovo podmnožico, je zelo preprosto izračunati njihovo skupno dolžino, tako da njihove dolžine preprosto seštejemo. Če pa poznamo skupno dolžino vseh palic v podmnožici, pa je zelo težko iz celotne množice izbrati tisto podmnožico, ki bo ustrezala tej dolžini.

V šifrirnem postopku, ki ga predlaga Merkle je čistopis $\mathbf{C} = (c_1, c_2, c_3, ..., c_n)$ binarni vektor. Ključ za šifriranje $\mathbf{K}_s = (k_1, k_2, k_3, ..., k_n)$ je vektor vseh možnih dolžin palic. Šifriranje izvedemo tako, da izmed vseh možnih palic izberemo tiste, pri katerih je soležen bit čistopisa enak 1.

Njihova skupna dolžina, ki predstavlja potem šifropis $S$, je enaka skalarnemu produktu ključa in čistopisa:

$$S = \mathbf{K}_s \cdot \mathbf{C} \tag{3.41}$$

kar je zelo preprost postopek. Če poznamo šifropis $S$ in $\mathbf{K}_s$, pa je zelo težko najti binarni vektor $\mathbf{C}$, ki bi zadovoljeval zgornjo enačbo. Skalarni produkt vektorja naravnih števil z binarnim vektorjem je zato enosmerna funkcija, vendar le, če $\mathbf{K}_s$ izpolnjuje določene pogoje.

V posebnem primeru, ko elementi zelo hitro naraščajo, tako, da je naslednji element večji od vsote vseh prejšnjih elementov:

$$k_n > \sum_{m=1}^{n-1} k_m \tag{3.42}$$

je rešitev enačbe (3.41) zelo preprosta. Elemente $\mathbf{C}$ določamo od zadnjega proti prvemu, tako da za padajoče vrednosti $m = n$ do $m = 1$ rekurzivno računamo po postopku:

$O_{n+1} = S$
Za $m = n$ do $m = 1$ s korakom $-1$ ponavljaj
    Če velja $O_{m+1} \geq k_m$ potem
        $c_m = 1$
    drugače
        $c_m = 0$
    $O_m = O_{m+1} - k_m\, c_m$

Na ta način smo pridobili vse elemente čistopisa $\mathbf{C}$. Ključ s tako lastnostjo zagotovo ni primeren za šifriranje. Poleg takega ključa tudi nekateri drugi ključi niso primerni, vendar smo takega omenili zato, ker ge je Merkle uporabil, da je v enosmerno funkcijo vgradil stranska vrata, ki jih potrebujemo za dešifriranje. Šifriranje poteka tako, da $\mathbf{K}_s$ izračunamo iz ključa $\mathbf{K}_d$, ki ima lastnost zelo hitrega naraščanja na tak način, da iz ključa $\mathbf{K}_d$ ni mogoče izračunati ključa $\mathbf{K}_s$, obenem pa je mogoče, če poznamo to preslikavo, šifropis $S$ preslikati v šifropis $S'$, ki bi ga dobili, če bi čistopis $\mathbf{C}$šifrirali s ključem $\mathbf{K}_d$.

Ključ $\mathbf{K}_s$ preslikamo v ključ $\mathbf{K}_d$ v skladu z izrazom.

$$\mathbf{K}_s = (w\, \mathbf{K}_d) \bmod m \tag{3.43}$$

kjer je $m$ naključno izbrano število, večje od vsote vseh elementov ključa $\mathbf{K}_d$ in $w$ naključno število tuje $m$. Potem je mogoče pokazati, da velja

$$S' = (w^{-1} \bmod m)\, S = \mathbf{K}_d \cdot \mathbf{C} \tag{3.44}$$

Ko želi Alenka poslati šifrirano sporočilo Borisu

- Boris najprej ustvari naključen ključ $\mathbf{K}_d$ s hitro naraščajočimi elementi,
- nato izbere števili $m$ in $w$ in
- izračuna $\mathbf{K}_s$.

- Ključ $\mathbf{K}_s$ pošlje Alenki,
- ki šifrira svoje sporočilo $S = \mathbf{K}_s \cdot \mathbf{C}$ in
- ga pošlje Borisu.
- Boris izračuna $S' = (w^{-1} \bmod m)\ S$ in ga po zgoraj opisanem postopku za ključe s hitro naraščajočimi elementi dešifrira.

Kdor ne pozna vrednosti $w$, $m$ in $\mathbf{K}_s$ sporočila ne more dešifrirati, ne da bi rešil enačbo (3.41), kar pa je računsko neizvedljiva naloga.

**Primer 3.12**

- Boris izbere $\mathbf{K}_d = (8, 26, 70, 202, 310, 630, 1.319)$, $m = 8.443$ in $w = 2.550$
- in izračuna $\mathbf{K}_s = (3.514, 7.199, 1.197, 77, 5.301, 2.330, 3.136)$, ki ga pošlje Alenki.
- Alenka šifrira sporočilo svoje sporočilo $\mathbf{C} = 1, 0, 0, 1, 1, 0, 1$ s ključem $\mathbf{K}_s$ in dobi $S = 12.028$, ki ga pošlje Borisu.
- Boris izračuna $w^{-1} = 3.950$ in $S' = 1.839$ in
- po opisanem iterativnem postopku določi elemente čistopisa:

| $m$ | $k_m$ | $c_m$ | $O_m$ |
|---|---|---|---|
| 8 | | | 1.839 |
| 7 | 1.319 | 1 | 520 |
| 6 | 630 | 0 | 520 |
| 5 | 310 | 1 | 210 |
| 4 | 202 | 1 | 8 |
| 3 | 70 | 0 | 8 |
| 2 | 26 | 0 | 8 |
| 1 | 8 | 1 | 0 |

Šifra z nahrbtnikom je bila videti ob svojem nastanku zelo obetavna, vendar je že leta 1982 Shamir odkril način, ki mu je omogočil iz javnega ključa $\mathbf{K}_s$ določiti vrednosti $m'$ in $w'$. Ti vrednosti nista nujno enaki originalnima $m$ in $w$, vendar je mogoče z njima preslikati javni ključ in šifropis tako, da ima ključ hitro naraščajoče elemente in je z njim možno dešifrirati preslikani šifropis. S tem je bila šifra z nahrbtnikom s stranskimi vrati razbita in se ne uporablja več.

### 3.5.7 Varnost ključev

Ker sloni varnost šifrirnih postopkov na tajnosti ključev, je zelo pomembna njihova ustrezna izbira. Ključi morajo biti:

- dovolj dolgi, da onemogočajo napad grobe sile pri simetričnih šifrah oziroma izračun inverzne enosmerne funkcije pri asimetričnih, in
- naključni, s čim bolj enakomerno porazdelitvijo verjetnosti tako, da ne olajšajo napada z ugibanjem bolj verjetnih ključev.

Za izračun inverzne funkcije pri asimetričnih šifrah je računsko manj zahteven kot preizkušanje vseh ključev pri napadu grobe sile na simetrične šifre, zato morajo biti za enako varnost ključi pri asimetričnih šifrah daljši kot pri simetričnih. V spodnji tabeli so podane bitne dolžine ključev, ki pri različnih šifrah zagotavljajo približno enako varnost. Primerjava sloni na primerjavi računske zahtevnosti napada grobe sile pri simetričnih šifrah in izračuna diskretnega logaritma oziroma faktorizacije pri asimetričnih šifrah, ki so zasnovane na enosmerni eksponentni funkciji nad GF(N) oziroma nad eliptičnimi krivuljami.

| simetrične šifre | diskretni eksponent | eliptične krivulje |
|---:|---:|---:|
| 80 | 1.024 | 160 |
| 112 | 2.048 | 224 |
| 128 | 3.072 | 256 |
| 192 | 7.680 | 384 |
| 256 | 15.360 | 521 |

Vidimo, da so dolžine ključev pri asimetričnih šifrah, ki so zasnovane na eliptičnih krivuljah bistveno krajše kot pri RSA, kar omogoča tudi bistveno učinkovitejšo izvedbo. Šifriranje z eliptičnimi krivuljami pri dolžini ključa 256 bitov je približno dvajsetkrat hitrejše, kot RSA šifriranje z 2.048 bitov dolgim ključem, hkrati pa zagotavlja, v skladu z zgornjo tabelo, tudi večjo varnost.

Tako varnost pa zagotavljajo le ključi, ki so resnično naključni, zato je za ustvarjanje ključev izredno pomemben ustrezen generator naključnih števil. Za tvorjenje naključnih števil se običajno uporabljajo računalniški programi, ki ne morejo tvoriti resnično naključnih števil, temveč tvorijo zgolj števila, ki so na videz naključna. To so tako imenovana pseudo-naključna števila. Taki generatorji imajo lahko slabosti, ki napadalcu olajšajo napad. Leta 2013 je New York Times razkril, da je bilo v generatorju naključnih števil, ki ga je standardiziral NIST, namerno vgrajena slabost. Pri generaciji RSA ključev zadošča, da eno od obeh praštevil $p$ in $q$ ni izbrano čisto naključno, da omogoča napadalcu, ki to ve, bistveno lažji napad na RSA.

Le generatorji, ki slone na nekem naravnem naključnem procesu, kot je to na primer termični šum, lahko generirajo resnično naključna števila.

Ker so asimetrične šifre računsko mnogo bolj zahtevne kot simetrične šifre, niso najprimernejše za šifriranje dolgih sporočil. Običajno se uporabljajo zgolj za izmenjavo ključev, ki jih potem

uporabimo za šifriranje s asimetričnimi šiframi. To obenem omogoča tudi pogosto menjavo ključev pri simetričnem šifriranju, kar povečuje varnost simetričnih šifer.

## 3.6 Zgostitvene funkcije

Zgostitvene funkcije (ang. hash functions) so funkcije, ki nek čistopis poljubne dolžine preslikajo v niz bitov točno določene (običajno manjše) dolžine, ki ga pogosto imenujemo tudi izvleček sporočila (ang. message digest).

Izvleček sporočila lahko uporabimo za odkrivanje sprememb sporočila. Spremembe lahko v sporočilu nastanejo nenamerno, zaradi napak pri prenosu oziroma shranjevanju sporočila. Sporočilo pa lahko spremeni tudi napadalec na IKS, da bi s tem dosegel neko korist oziroma, da bi povzročil neko škodo.

Pri prenosu sporočil se izvlečki uporabljajo za odkrivanje in/ali odpravljanje napak, do katerih pride pri prenosu preko neidealnega prenosnega kanala. Pri prenosu sporočil se namesto termina izvleček uporabljajo termini kot kontrolna vsota (ang. checksum), redundanca (ang. redundancy) ali pariteta (ang. parity).

Lastnosti, ki jih morajo imeti zgostitvene funkcije za zaščito pred nenamernimi in namernimi spremembami se razlikujejo. Zaščita pred namernimi spremembami je namreč bistveno zahtevnejša naloga kot zaščita pred nenamernimi spremembami. Napadalec, ki želi namerno spremeniti nek čistopis, se bo namreč potrudil, da bo taka sprememba nezaznavna. Poleg ustrezne zgostitvene funkcije je za to potrebna še vrsta drugih ukrepov, o katerih bo več govora v naslednjih poglavjih, ko bomo govorili o avtentičnosti in verodostojnosti.

### 3.6.1 Zaščita proti nenamernim spremembam

Pri prenosu in shranjevanju podatkov se za njihovo zaščito pred nenamernimi spremembami uporabljajo kode za odkrivanje in kode za odpravljanje napak. S kodami za odkrivanje napak lahko zgolj odkrijemo, da je prišlo do napake, s kodami za odpravljanje napak pa lahko odkrito napako tudi odpravimo.

Za namen odkrivanja napak pri prenosu se pogosto uporabljajo blokovne kode, ki vsakemu bloku podatkov dodajo redundantne podatke (pariteto), ki so s pomočjo neke zgostitvene funkcije pridobljeni iz osnovnih podatkov. Pri sprejemu odkrijemo napako v sprejetem bloku, če z isto zgostitveno funkcijo izračunana pariteta sprejetega bloka, ne ustreza pariteti, ki smo je bila dodana bloku. Podoben način se uporablja tudi pri zapisu in branju podatkov iz nekega pomnilniškega medija, na primer magnetnega diska.

Zgostitvena funkcija, ki jo uporabimo v ta namen mora zagotavljati, da sprememba enega ali več bitov v bloku povzroči tudi spremembo paritete. Verjetnost, da bi ostala pri naključni spremembi podatkov pariteta nespremenjena, mora bit čim manjša. Večinoma te funkcije zagotavljajo, da je zagotovo mogoče odkriti določeno število napak v posameznem bloku sporočila. Če odkrijemo napako pri prenosu, lahko zahtevamo ponovno pošiljanje bloka podatkov, v katerem je bila odkrita napaka.

Za primer si oglejmo kodo CRC (ang. cyclic redundancy check) za preverjanje redundance, ki se pogosto uporablja pri prenosu podatkov, kot na primer v internetnem protokolu TCP v lokalnem omrežju Ethernet, v brezžičnemm omrežju WLAN in drugod. CRC koda je ostanek deljenja polinomov v GF(2), kjer biti v bloku sporočila predstavljajo koeficiente polinoma. Na primer blok sporočila

$$C = 01101010010$$

predstavlja polinom:

$$c(x) = x^9 + x^8 + x^6 + x^4 + x$$

Sporočilo delimo z generatorskim CRC polinomom $g(x)$. Bitni zapis generatorskega polinoma $\mathbf{G}$ je zaporedje bitov, ki predstavljalo koeficiente generatorskega polinoma. Deljenje polinomov se izvaja na enak način kot deljenje dveh števil, saj so števila v številskem sitemu vedno predstavljena v obliki polinomov. Razlika je v tem, da se tu posamezne operacije izvajajo po modulu. Ker računamo CRC v GF(2), se operacije izvajajo po modulu 2. Seštevanje in odštevanje po modulu 2 pa je enako XOR operaciji, kar močno poenostavi računanje.

CRC polinomi so lahko različnih dolžin. Dolžina CRC polinoma določa tudi dolžino ostanka in s tem dolžino CRC kode. Daljše CRC kode omogočajo tudi odkrivanje večjega števila napak. CRC polinomi so standardizirani v različnih standardih za prenos podatkov. Prvi bit v CRC polinomu je vedno postavljen na 1, saj re polinoma določa njegovo dolžino. Pri polinomu $n$-tega reda je število bitov enako $n + 1$.

Postopek za izračun CRC kode v GF(2) izvedemo po naslednjem rekurzivnem postopku:

$$\mathbf{O}_{m+1} = \begin{cases} \mathbf{O}_m \oplus \mathbf{G}_m & ; \quad o_{m,m} = 1 \\ \mathbf{O}_m & ; \quad o_{m,m} = 0 \end{cases} \tag{3.45}$$

$$\mathbf{G}_{m+1} = \mathrm{rd}(\mathbf{G}_m) \tag{3.46}$$

kjer je z $o_{m,m}$ označen $m - ti$ bit ostanka $\mathbf{O}_n$, z $\mathrm{rd}(\mathbf{G}_m)$ pa je označena rotacija $\mathbf{G}_m$ za en bit v desno. Začetna vrednost $\mathbf{O}_0$ je enaka sporočilu $\mathbf{C}$, ki mu na desni dodamo $n$ ničel, začetna vrednost $\mathbf{G}_0$, pa je enaka generatorskemu polinomu $\mathbf{G}$, ki ga na desni dopolnimo z ničlami do dolžine $\mathbf{O}_0$. Zadnji ostanek je poten enak CRC kodi sporočila.

**Primer 3.13**

Za primer si oglejmo CRC kodo sporočila "Hi", ki jo dobimo z uporabo generatorskega polinoma CRC-6, ki se uporablja v radio-frekvenčni identifikaciji (RFID – Radio-Frequency IDentification). V spodnji tabeli so izpuščenih koraki, pri katerih je $o_{m,m} = 0$ in se ostanek ne spremeni. V vsakem izpuščenem koraku se genoratorski polinom zasuče za en bit v desno.

$$g(x) = x^6 + x + 1$$

Temu polinomu ustreza zaporedje bitov 1000011. Sporočilo Hi najprej zapišemo v binarni obliki s 7-bitnimi črkami po ASCII kodni tabeli in mu dodamo 6 ničel na desni strani:

$$\mathbf{C} = 1001000\ 1001001\ 000000$$

Sedaj lahko po opisanem postopku izračunamo CRC.

$$
\begin{array}{rl}
 & 10010001001001000000 \quad \mathbf{O_0} \\
\oplus & 10000110000000000000 \quad \mathbf{G_0} \\
= & 00010111001001000000 \quad \mathbf{O_3} \\
\oplus & 00010000110000000000 \quad \mathbf{G_3} \\
= & 00000111111001000000 \quad \mathbf{O_5} \\
\oplus & 00000100001100000000 \quad \mathbf{G_5} \\
= & 00000011110101000000 \quad \mathbf{O_6} \\
\oplus & 00000010000110000000 \quad \mathbf{G_6} \\
= & 00000001110011000000 \quad \mathbf{O_7} \\
\oplus & 00000001000011000000 \quad \mathbf{G_7} \\
= & 00000000110000000000 \quad \mathbf{O_8} \\
\oplus & 00000000100001100000 \quad \mathbf{G_8} \\
= & 00000000010001100000 \quad \mathbf{O_9} \\
\oplus & 00000000010000110000 \quad \mathbf{G_9} \\
= & 00000000000001010000 \quad \mathbf{O_{13}} \\
\oplus & 00000000000001000011 \quad \mathbf{G_{13}} \\
= & 00000000000000010011 \quad \mathbf{O_{14} = CRC}
\end{array}
$$

CRC koda sporočila "Hi" je enaka zadnjemu $\mathbf{O}_{14}$, kar pomeni $\mathbf{CRC} = 010011$.

CRC koda omogoča odkrivanje nenamernih napak, vendar pa nudi zaščite pred namernimi spremembami sporočila. Zaščite podatkov s CRC kodo torej ne moremo šteti kot del dejanskega varovanja podatkov. V okviru varnosti IKS služi zgolj za zagotavljanje zanesljivega prenosnega kanala na nižje ležečih slojih komunikacijskega protokola. Dejansko zaščito pred spremembami podatkov, tako nenamernimi kot namernimi, moramo zagotoviti na više ležečih slojih. Za zaščito potrebujemo kriptografske zgostitvene funkcije.

### 3.6.2 Kriptografske zgostitvene funkcije

Pri zgostitvenih funkcijah, ki jih rabimo za izračun parite, kot je to CRC, zadošča, da omogočajo odkrivanje določenega števila napak v bloku sporočila. Kriptografske zgostitvene funkcije (ang. cryptographic hash functions), ki jih potrebujemo za zaščito celovitosti podatkov, so mnogo zahtevnejše. Kriptografska zgostitvena funkcija $h(\mathbf{C})$ preslikati poljubno dolg čistopis $\mathbf{C}$ v njegov povzetek, ki ga imenujemo tudi prstni odtis (ang. fingerprint), $\mathbf{P}$ konstantne dolžine:

$$\mathbf{P} = h(\mathbf{C}) \tag{3.47}$$

Tako kot pri ljudeh, kjer niti dva človeka nimata enakega prstenga odtisa, bi želeli tudi, da niti dva različna čistopisa ne bi imela istega prstnega odtisa. Ker je lahko čistopis poljubno dolg je število vseh možnih čistopisov neomejeno. Število vseh možnih podpisov je omejeno z dolžino podpisa, zato je možnih neomejeno število čistopisov, ki imajo enak prstni odtis.

- Verjetnost, da bi imela dva naključna čistopisa enak prstni odtis, mora biti zanemarljivo majhna.

- Podpis mora biti na videz naključen z enakomerno porazdelitvijo verjetnosti. Sprememba enega bita v čistopisu mora spremeniti približno polovico bitov v prstnem odtisu.
- Praktično neizvedljivo mora biti narediti čistopis, ki bi imel vnaprej določen prstni odtis. To pomeni, da je zgostitvena funkcija enosmerna, saj bi drugače lahko z inverzno funkcijo določili čistopis, ki pripada izbranemu prstnemu odtisu.
- Praktično neizvedljivo mora biti spremeniti čistopis ne da bi se spremenil tudi njegov prstni odtis.
- Praktično neizvedljivo mora biti najti dva čistopisa z enakim prstnim odtisom.

Načrtovanje kriptografsko močnih enosmernih funkcij ni preprosta naloga. Za veliko predlaganih zgostitvenih funkcij se je kasneje izkazalo, da imajo določene slabosti, zaradi katerih niso primerne za uporabo pri zaščiti celovitosti podatkov.

Zgostitvene funkcije lahko razdelimo v dve kategoriji: na tiste, ki so zasnovane na simetričnih blokovnih šifrah in na tiste, ki so zasnovane na modularni aritmetiki. V praksi se uporabljajo predvsem prve, saj so računsko manj zahtevne, zato bomo v nadaljevanju obravnavali zgolj te.

### 3.6.3 Zgostitvene funkcije na osnovi simetričnih blokovnih šifer

Kriptografske zgostitvene funkcije lahko izvedemo s pomočjo blokovnih šifrirnih funkcij ali s pomočjo komponent, ki so tem podobne, a so zasnovane namensko za zgostitvene funkcije. Šifrirna funkcija $f_s$ preslika vhodni blok $\mathbf{C}_n$ v izhodni blok $\mathbf{S}_n$ v odvisnosti od ključa $\mathbf{K}$. V razdelku 3.4.1 smo spoznali veriženje blokov. Pr veriženju blokov je čistopisu pred šifriranjem prištejemo šifropis prejšnjega bloka po modulu 2 oziroma z XOR operacijo:

$$\mathbf{S}_{n+1} = f_d(\mathbf{S}_n \oplus \mathbf{C}_{n+1}, \mathbf{K}) \tag{3.48}$$

Zadnji blok šifropisa je tako odvisen od vseh blokov čistopisa in lahko služi kot prstni odtis celega čistopisa. Problem pri taki zgostitveni funkciji je, da ni enosmerna. Kdor pozna ključ $\mathbf{K}$ lahko poljubnemu $M$ blokov dolgemu čistopisu doda $M + 1$ blok, tako da prstni odtis novega čistopisa enak vnaprej izbranemu prstnemu odtisu $\mathbf{P}$. Blok, ki ga mora dodati, izračuna v skladu z izrazom:

$$\mathbf{C}_{M+1} = f_d(\mathbf{P}, \mathbf{K}) \oplus \mathbf{S}_M \tag{3.49}$$

kjer je $\mathbf{S}_M$ zadnji blok šifropisa izbranega $M$ blokov dolgega čistopisa. Pri uporabi take zgostitvene funkcije mora zato ključ ostati tajen, vendar lahko potem preverja spremembe čistopisa le kdor pozna tajni ključ. Zgostitvene funkcije, ki potrebujejo tajni ključ se lahko uporabljajo tudi za avtentikacijo sporočil, zato jih imenujemo tudi kode za avtentikacijo sporočil (MAC – Message Authentication Codes)

Če želimo, da lahko vsakdo preveri, ali je bil čistopis spremenjen, mora biti zgostitvena funkcija enosmerna. Če je funkcija enosmerna, ne potrebujemo tajnega ključa. To dosežemo s posebej načrtovanimi funkcijami nad bloki, ki onemogočajo izračun inverzne funkcije. Na ta

način je izvedenih večina dobro znanih zgostitvenih funkcij, kot so to MD4, MD5, gost, ripmed, haval, SHA-0, SHA-1, SHA-2 in SHA-3 in druge. SHA-3 funkcije so trenutno še v postopku standardizacije. Dokončen standard se pričakuje v prvi polovici leta 2014.

Ker lahko s takimi funkcijami preverjamo spremembe sporočila, jih imenujemo tudi kode za zaznavo sprememb (MDC – Modification Detection Codes). MDC lahko preprosto uporabimo tudi kot MAC tako, da sporočilu pred zgoščanjem dodamo nek tajen ključ kot predpono sporočila.

Pri načrtovanju zgostitvenih funkcij je poleg odpornosti proti napadom pomembna tudi računska učinkovitost in preprosta tako programska kot tudi strojna izvedba. Zato vse omenjene zgostitvene funkcije izvajajo serije operacij v več ciklih, na enak način kot blokovne šifre. Operacije, ki se izvajajo v posameznem ciklu, lahko zajemajo substitucije, permutacije, rotacije, operacije po modulu in logične operacije med posameznimi biti.

Dobre zgostitvene funkcije morajo biti odporne proti vsem znanim napadom. Za zgostitvene funkcije, ki nimajo slabosti, je najuspešnejši napad s kolizijo čistopisov (ang. collision attack). Pri tem napadu skuša napadalec najti par čistopisov z istim prstnim odtisom. Na zgostitvene funkcije MD4, MD5, SHA-0 so že bili izvedeni uspešni napadi. Na funkcijo SHA-1 obstaja napad, ki teoretično omogoča odkriti par čistopisov z istim podpisom, vendar bi bilo zato potrebno izračunati $2^{60}$ podpisov, kar je zaenkrat še neizvedljivo. Kljub temu NIST od leta 2010 ne priporoča več uporabe SHA-1.

Če zgostitvena funkcija nima nobenih slabosti, potem je najboljši napad s kolizijo tako imenovan rojstnodnevni napad (ang. birthday attack). Rojstnodnevni napad je neke vrste napad grobe sile, ki pa je mnogo uspešnejši od iskanja čistopisa, ki ustreza točno določenemu prstnemu odtisu. Ime izvira iz vprašanja, koliko oseb mora biti v prostoru, da imata s vsaj 50% verjetnostjo vsaj dve osebi rojstni dan istega dne v letu. Odgovor na to vprašanje se zdi presenetljiv, saj zadošča že, da je v prostoru 23 oseb. Če je v prostoru 30 oseb, pa je ta verjetnost že približno 70%.

Opisani problem je ekvivalenten problemu iskanja dveh čistopisov $\mathbf{C}_1$ in $\mathbf{C}_2$, ki imata enak prstni odtis, oziroma za katera velja:

$$h(\mathbf{C}_1) = h(\mathbf{C}_2) \tag{3.50}$$

Če zgostitvena funkcija $h(\mathbf{C})$ nima slabosti, potem so pri naključni izbiri čistopisa vsi prstni odtisi enako verjetni. Verjetnost kolizije $P_{kol}(n, N)$, da bi našli v $n$ poskusih našli dva čistopisa z enakim prstnim odtisom je enaka verjetnosti, da bi izmed vseh $N$ možnih podpisov v $n$ poskusih dvakrat izberemo isti podpis. Ta verjetnost je pri velikem $N$ približno enaka:

$$P_{kol}(n, N) \doteq 1 - e^{-n^2/(2N)} \tag{3.51}$$

Povprečno število poskusov, ki jih potrebujemo, da bi imela dva čistopisa enak prstni odtis, pa je približno enako:

$$\overline{n} \doteq 1,25\sqrt{N} = 1,252^{m/2} \tag{3.52}$$

kjer smo z $m$ označili dolžino prstnega odtisa v bitih, tako da je število vseh možnih podpisov enako:

$$N = 2^m \tag{3.53}$$

Pri 128 bitov dolgem prstnem odtisu bi morali tako v povprečju izračunati približno $2,3 \cdot 10^{19}$, pri 256 bitov dolgem prstem odtisu pa že $2,6 \cdot 10^38$ prstnih odtisov, da bi našli dva čistopisa z enakim prstnim odtisom. Trdimo lahko, da so zgostitvene funkcije, ki preslikajo čistopis v prstni odtis ki je dolg vsaj 128 bitov zaenkrat varne pred rojstnodnevnim napadom, če le ne vsebujejo nekih šibkosti, ki bi jih napadalec lahko izkoristil.

### Zgostitvene funkcije SHA

Zgostitvene funkcije z oznako SHA (ang. secure hash algorithm) so funkcije, ki jih je standardiziral NIST, in se najpogosteje uporabljajo v kriptografiji. Hkrati z razvojem računalnikov kot tudi z razvojem različnih algoritmov kriptoanalize, so bile potrebno kriptografsko vedno močnejše funkcije. Tako je NIST izdal standarde SH-0, SH-1 in SH-2. Zadnji je trenutno priporočen za uporabo, v pripravi pa je že nov standard SH-3, ki bo predvsem računsko učinkovitejši od SH-2, saj potrebuje za izračun prstnega odtisa bistveno manj ciklov. V spodnji tabeli so podane osnovne lastnosti posameznih zgostitvenih funkcij iz družine SH.

| $h(\mathbf{C})$ | dolžina prstnega odtisa | št. ciklov | operacije | uspešen napad |
|---|---|---|---|---|
| SHA-0 | 160 | 80 | AND, OR, XOR, ROT, + | da |
| SHA-1 | 160 | 80 | AND, OR, XOR, ROT, + | teoretično $2^{60}$ |
| SHA-2 | 224, 256 / 384, 512 | 64 / 80 | AND, OR, XOR, ROT, SHR, + | ne |
| SHA-3 | 224, 256, 384, 512 | 24 | AND, XOR, NOT, ROT | ne |

Poleg družine SHA je bilo predlaganih še cela vrsta drugih zgostitvenih funkcij. Na mnoge med njimi je bil že izveden uspešen kolizijski napad ali pa vsaj teoretično obstaja napad, ki zahteva manj računanja kot rojstnodnevni napad grobe sile.

Ker je družina funkcij SHA-3 še v postopku standardizacije, je trenutno najbolj aktualna uporaba funkcij iz družine SHA-2, v kateri so štiri zgostitvene funkcije SHA-224, SHA-256, SHA-384 in SHA-512, ki so imenovane po dolžini prstnega odtisa, ki ga dobimo kot rezultat zgoščanja. Vse funkcije iz družine se izvajajo po istem postopku, razlikujejo se zgolj po dolžini besed, začetnih vrednostih in številu ciklov. V vsakem ciklu se izvajajo logične operacije AND, OR in XOR, aritmetična operacija seštevanja po modulu ter rotacija ROT in premik v desno SHR.

### Primer 3.14

Kot primer si oglejmo prstni odtis dveh sporočil, ki se razlikujeta samo v enem znaku, prvo se konča s piko in drugo s klicajem. Prstni odtis je izračunan s funkcijo SHA-256 in je zapisan v heksadecimalni obliki.

SHA-256("Dober dan.")

```
f1749f078f33a2b0f548000c4513ceb228fc48121c4592aecf5f1fb90d15997e
```

SHA-256("Dober dan!")

```
37ef87c9b6df31371f02b9e7688d9802f8cee28c1d5458bcb17d2caa791d583a
```

Iz primera vidimo, da je majhna sprememba čistopisa popolnoma spremenila njegov prstni odtis. Zelo težko bi bilo spremeniti čistopis tako, da se prstni odtis ne bi spremenil. Za SHA-256 zaenkrat še nikomur ni uspelo najti dveh različnih čistopisov z enakim prstnim odtisom.

# 4 Identifikacija

Za varnost vsakega IKSa je izrednega pomena ustrezen način identifikacije uporabnikov. Vsak uporabnik IKSa se mora prijaviti v IKS na tak način, da je nedvomno jasna njegova identiteta. Različni uporabniki imajo namreč v IKSu različne pravice in dostop do različnih podatkov. Na primer, pri prijavi v sistem elektronskega bančništva ima uporabnik pravico dostopa zgolj do svojega računa, nima in tudi ne sme pa imeti dostopa do drugih računov, če za to ni pooblaščen. Dodeljevanje pravic posameznim uporabnikom imenujemo tudi avtorizacija. Kdo, kdaj in na kakšen način sme dostopati do IKSa mora biti določeno v varnostni politiki IKSa. Ravno tako pa mora biti določeno tudi, na kak način se lahko te pravice uveljavljajo oziroma, na kak način se uporabnikom onemogočajo dejanja, do katerih niso opravičeni. Zagotovo pa varnostne politike ni mogoče izvajati brez ustrezne identifikacije.

Ravno tako pa je pomembna tudi identifikacija strežnika, na katerega se nek uporabnik prijavlja. Na ta način se prepreči prijava na lažni strežnik, ki omogoča lažnemu strežniku krajo podatkov o uporabniku, ki jih ta prenese, ker misli, da je prijavljen na pravi strežnik.

V fizičnem svetu se običajno identificiramo z nekim osebnim dokumentom, kot so to osebna izkaznica, potni list, prepustnica in podobno. Ti dokumenti morajo biti opremljeni s sliko in/ali nekimi biometričnimi podtaki, kot so to teža, velikost, spol, prstni odtis in podobno. Na osnovi teh podatkov je namreč možno preveriti, ali je oseba res tista za katero se predstavlja.

Za dostop do IKSa tak način identifikacije ni najprimernejši, zato je potrebno poiskati druge načine. V splošnem lahko rečemo, da se nekdo lahko identificira s tem kar ve (tajno geslo, tajni ključ), z nečem kar ima (na primer pametna kartica, fizični ključ, dovolilnica, ..) ali s tem kar je (biometrični podatki). Najbolj zanesljivi sistemi identifikacije uporabljajo vse tri načine.

Pojem identifikacija se pogosto uporablja kot sinonim za avtentikacijo, čeprav imata pojma nekoliko drugačen pomen. Identifikacija se nanaša na istovetnost oseb ali naprav, medtem ko se nanaša avtentikacija na istovetnost dokumentov in sporočil. Ker se v svetu elektronskih komunikacij identificiramo tak, da preko omrežja pošiljamo elektronska sporočila, je seveda pomembna avtentičnost oziroma istovetnost teh sporočil. Kljub vsesplošni uporabi pojma avtentikacija za namen identifikacije predvsem na angleškem govornem področju, kjer so v splošni rabi termini kot "user authentication, "authentication protocol", "authentication server" in podobno, bomo v nadaljevanju skušali ločevati oba pojma, čeprav bomo pri tem včasih prišli v neskladje z angleško terminologijo.

## 4.1 Identifikacija z geslom

Avtentikacija z geslom je najstarejši in še vedno najpogosteje uporabljan sistem avtentikacije. Pri tem načinu si uporabnik omogoči dostop do IKSa tako, da vpiše svoje uporabniško ime in

geslo. Uporabniško ime ni nujno skrivnost medtem ko mora geslo ostati nujno tajno in ga mora praviloma poznati zgolj uporabnik. Le na ta način je namreč mogoča identifikacija uporabnika na osnovi tega kar ve.

### 4.1.1 Napadi na gesla

Napad na uporabniška gesla je med najlažjimi in najbolj pogostimi napadi na informacijski sistem. Napadalec želi pri takem napadu pridobiti gesla enega ali več uporabnikov, kar mu omogoča nepooblaščen dostop do IKSa z vsemi pravicami, ki pripadajo lastniku gesla. Napadalec skuša pri napadu izkoristiti tako napake uporabnikov pri upravljanju z gesli kot tudi tehnične pomanjkljivosti izvedbe identifikacije z gesli. Za varnost IKSa je zato izredno pomembno, da uporabniki pravilno upravljajo z gesli kot tudi, da je identifikacija na osnovi gesel tehnično pravilno izvedena.

Da bi se lahko zaščitili pred napadi na gesla, moramo poznati vsaj osnovne načine teh napadov.

#### Napad grobe sile

Pri napadu grobe sile napadalec preizkuša vsa možna gesla. Napad grobe sile je lahko uspešen zgolj na šibka gesla, to je gesla, ki imajo relativno malo možnih vrednosti, ker so prekratka.

Napad grobe sile je najuspešnejši, če začne napadalec preizkušati z najšibkejšimi in najbolj verjetnimi gesli. Torej začne z kratkimi gesli in nato dolžino povečuje. Pri vsaki dolžini najprej preizkusi kombinacije samo malih črk, samo velikih črk, malih črk in številk, velikih črk in številk in tako naprej.

#### Ugibanje gesel

Napadalec lahko skuša uganiti geslo uporabnika o katerem ima določene podatke. Lahko uporabnika pozna, ali pa pride do njegovih podatkov preko spletnih imenikov, spletnih strani podjetja, bogov ali in socialnih omrežij, kot so Facebook, Twitter, LinkedIn, Researcher Gate in podobna. Preizkuša lahko imena, rojstne datume in telefonske številke prijateljev, imena hišnih ljubljenčkov, priljubljene filme in podobno. Veliko teh podatkov je namreč javno dostopnih v različnih javno dostopnih bazah podatkov, na spletnih straneh podjetij in v socialnih omrežjih, kot so to Facebook, Tweeter, LinkedIn, Researcher Gate in podobna.

Pri ugibanju gesel s pomočjo nekega programa lahko napadalec uporabi tudi obsežne tabele najpogosteje uporabljanih gesel. V letu 2012 je bilo na primer najpogosteje uporabljeno geslo "password", na drugem mestu pa je bilo geslo "123456". Na našo srečo so take tabele dostopne predvsem za angleško govorno področje.

#### Družbeni inženiring

Do prvega gesla pride napadalec najlaže z družbenim inženiringom, tako da pridobi geslo od uporabnika, ki slabo upravlja z gesli. Napadalec lahko preprosto pokliče uporabnika po telefonu

ali mu pošlje elektronsko pošto. Predstavi se lahko kot administrator IKSa in prosi uporabnika za geslo, ki ga nujno potrebuje za odpravljanje tehničnih težav. Če tehnične težave ne bodo odpravljene, uporabnik ne bo imel več dostopa do svojega računa, ne bo mogel prejemati elektronske pošte, ne bo mogel dvigniti denarja na bankomatu ali kaj podobnega.

#### Opazovanje

Napadalec lahko opazuje uporabnika pri vnosu gesla, predvsem, kadar ta geslo vnaša na javnem mestu, na primer v lokalu z dostopom do Interneta, ali pri vnosu PIN kode na POS terminalu.

Če ima dostop do delovnega okolja uporabnika, lahko najde tudi gesla, ki so zapisana na nezavarovanih mestih, kot je to spodnja stran tipkovnice, zadnja stran zaslona, stranica predala in podobno. Na enak način lahko pride tudi do gesel, ki so nezavarovano zapisana v pametnem telefonu, če uporabnik tak telefon pusti nekje na mizi ali ga izgubi.

#### Lažne vstopne točke

Napadalec lahko preusmeri uporabnika na lažno spletno stran, ki je na videti enako, kot originalna spletna stran, preko katere uporabnik običajno vstopa v nek IKS. Ko uporabnik vpiše svoje uporabniško ime in geslo, ga lahko lažna spletna stran preusmeri na originalno spletno stran, tako, da uporabnik ne opazi prevare.

Na podoben način lahko napadalci pridejo tudi do gesel (PIN kode) plačilnih kartic s postavljanjem lažnih bankomatov. Ko uporabnik vtipka svojo PIN kodo, mu tak bankomat kartico vzame, češ, da je neveljavna ali pa, da je koda nepravilna. Napadalec na ta način pridobi plačilno kartico skupaj s PIN kodo in lahko z njo na pravem bankomatu dviguje denar.

#### Prestrezanje gesel

Napadalec, ki uspe prestreči komunikacijo med delovno postajo uporabnika in omrežjem IKSa, lahko prestreže tudi uporabniško ime in geslo ob prijavi. To velja samo za prijave, kjer se gesla prenašajo po omrežju nezaščiteno. Za tak napad mora napadalec prej uspešno prodreti v neko omrežje, preko katerega se prenašajo sporočila.

#### Zlonamerna programska oprema

Če napadalcu uspe namestiti neko zlonamerno programsko opremo na delovno postajo uporabnika, lahko nadzira vse, kar uporabnik vtipka na svoji tipkovnici oziroma opazuje, kaj je prikazano na njegovem ekranu.

Zlonameren program lahko posreduje na neko izbrano lokacijo tudi vse datoteke z gesli, ki so shranjene na računalniku. Primer takega programa je program iStealer, ki poišče datoteke z gesli različnih programov, kot so to spletni brskalniki (Internet explorer, Firefox, Google, Chrome, Opera, ...), FTP klienti (FileZilla, SmartFtp, ...) in podobno.

#### Fizičen dostop do računalnika

Napadalec, ki ima fizičen dostop do računalnika, lahko uporabi program, ki omogoča ponastavitev gesel. Na ta način pridobi popolno kontrolo nad računalnikom. Primer takega programa je active Password Changer, ki je prvotno namenjen upravičenim uporabnikom, ki so pozabili svoje geslo. Program omogoča spremembo administratorskega gesla in spremembo gesel uporabnikov v okolju Windows.

### 4.1.2 Upravljanje z gesli

Slabo upravljanje z gesli je najšibkejša točka varnosti IKSa. Največ napadov na IKSe je bilo uspešnih zaradi izbire slabih gesel in neprevidnega ravnanja uporabnikov z njimi. To je najpogosteje posledica neosveščenosti uporabnikov in njihovega nepoznavanja nevarnosti, ki jih prinaša slabo upravljanje z gesli. Izobraževanje in ozaveščanje uporabnikov je zato za varnost IKSa ključnega pomena.

#### Napotki za upravljanje z gesli

Identifikacija uporabnikov na osnovi gesla ni najbolj varen način identifikacije. Varnost je mogoče bistveno povečati, če se uporabniki držijo osnovnih napotkov za upravljanje s svojimi gesli.

- Gesla morajo biti dovolj dolga, da onemogočajo napad grobe sile, to je ugibanje vseh možnih gesel. Vseh možnih gesel do dolžine $n$ je

$$N = L^{n+1} - 1L - 1 \doteq L^n \tag{4.1}$$

  kjer je z $L$ označeno število vseh možnih znakov v geslu. Če za geslo uporabimo vse znake na tipkovnici je $L = 94$, če pa uporabimo samo male črke in številke pa je $L = 41$. Če je geslo izbrano popolnoma naključno iz vseh znakov, potem proti napadu grobe sile zadošča geslo dolgo 12 znakov ($N \doteq 4,7 \cdot 10^{23}$). Če uporabimo zgolj male črke in številke, je temu geslu ekvivalentno geslo dolgo 15 črk, če pa uporabimo zgolj male črke, pa mora biti geslo, za enako stopnjo zaščite, dolgo že 17 znakov.

- Za dostop do različnih IKSov moramo nujno uporabljati različna gesla. Predvsem pa je nujno, da za dostop do nekih javnih spletnih strani (spletne trgovine, socialna omrežja, ...) nikoli ne uporabljamo istih gesel kot za dostop do privatnih omrežij oziroma do bančnega računa. Če namreč napadalec odkrije eno tako geslo, ima dostop do vseh IKSov, ki so zavarovani s tem geslom.

- Najmočnejša so gesla, ki so izbrana naključno, s pomočjo nekega preverjenega generatorja naključnih gesel. Na videz naključna gesla lahko generiramo tudi tako, da z neko zgostitveno funkcijo tipa MAC.

- Težava pri naključnih geslih je, da si jih ne moremo zapomniti, če posebej, če je teh gesel veliko. Zato jih moramo shranjevati. Gesla smemo shranjevati zgolj v šifrirani obliki.

Šifrirana gesla morajo biti zavarovana z močnim glavnim geslom, ki pa ne sme biti nikjer zapisano ali shranjeno, torej si ga moramo zapomniti. Če se bojimo, da bi lahko pozabili glavno geslo, ga lahko hranimo na nekem varnem mestu, na primer v sefu. Glavnega gesla ne smemo imeti zapisanega na računalniku, ki je povezan v omrežje, ali na pametnem telefonu.

- Gesla, ki jih želimo zapomnimo (kot je to na primer glavno geslo) morajo biti bistveno daljša od naključnih gesel. Izogibati se moramo gesel, ki jih je preprosto uganiti in tudi pogosto uporabljanih gesel, ki nastopajo v raznih tabelah za ugibanje gesel, kot tudi gesel, ki so povezana z našimi osebnimi podatki.

- Da bi si lahko zapomnili geslo, ki je dovolj dolgo, moramo uporabiti nek način, ki nam pri tem pomaga.

  – Kot geslo lahko izberemo nek stavek, ki ga pišemo brez presledkov in z namenskimi tipkarskimi napakami. Pri tem se izogibamo splošno znanih citatov, kot na primer "To be or not to be". Kot geslo bi na primer lahko izbrali stavek "To rabim vsak dan najmanj 10 krat.", ki ga bi zapisali v obliki "Torabnmusakdannejmen10X!". Dobro je, če v geslu nastopajao tudi številke in drugi znaki in da stavek ni povezan z našimi osebnimi podatki ali z IKSom, do katerega dostopamo z njim.

  – Lahko izberemo nek daljši stavek in kot geslo uporabimo prve črke besed v stavku.Iz prejšnjega stavka bi iz prvih črk dobili geslo "LiNdSiKgUpČbVs". Da dobimo močnejše geslo, lahko vsako drugo ali vsako tretjo črko pišemo z veliko začetnico in dodamo še kakšno ločilo ali številko.

- Svojega gesla nikomur nikoli ne zaupamo, niti sorodnikom ali dobrim prijateljem. Še posebej moramo biti pozorni, da gesla ne zaupamo neznancem in da ga ne sporočamo po telefonu ali elektronski pošti. Če moramo kdaj iz kakršnega koli razloga zaupati nekomu svoje geslo, ga takoj po tem, ko to ni več potrebno, spremenimo. Geslo, ki nam ga je administrator IKSa poslal po elektronski pošti, ali sporočil na nek drug, podoben način, takoj zamenjamo.

- Gesla za dostop do IKSov s pomembnimi podatki pogosto menjamo. Na ta način bistveno povečamo varnost, saj gesla, do katerih se je nek napadalec mogoče dokopal, niso več veljavna.

- Pri vnosu gesel se moramo zavedati okolice. Gesla ne vnašamo, če sumimo, da je v bližini oseba, ki bi nam želela ukrasti geslo. V takem primeru moramo to osebo zaprositi, da se obrne stran, preden vnesemo geslo.

- Izvajamo vse ukrepe, ki preprečujejo vdor na naš računalnik oziroma nameščanje zlonamerne programske opreme na njem. O ustreznih ukrepih bo več govora v naslednjih poglavjih.

### 4.1.3 Tehnična zaščita

Določenim napadom na gesla se lahko izognemo z ustrezno tehnično izvedbo identifikacije na osnovi gesla. S tehničnimi rešitvami se ne moremo izogniti napadom, ki so zasnovani na socialnem inženiringu ali neprevidnemu ravnanju z gesli, lahko pa preprečimo prestrezanje gesel kot tudi izbiro slabih gesel. Bistveno večjo varnost kot preprosta identifikacija na osnovi gesla pa nudijo drugi protokoli identifikacije, o katerih bo govora kasneje.

#### Zaščita proti prestrezanju

Trenutno so, kljub zelo nizki stopnji varnosti, še vedno pogosto v uporabi protokoli za dostop do IKSov, pri katerih se geslo po omrežju prenaša nezaščiteno, v obliki čistopisa. Primeri takih protokolov so protokol HTTP za dostop do spletnih strani, FTP protokol za oddaljen dostop do datotek, POP3 in IMAP protokola za dostop do elektronske pošte in drugi. Vsi ti protokoli imajo sicer tudi možnost varnega prenosa gesla, vendar te niso vedno izkoriščene.

Če bi pri prenosu geslo šifrirali, bi napadalcu sicer preprečili, da bi izvedel geslo, vendar to ne bi pomagalo. Napadalcu namreč zadošča, da ima na voljo šifrirano geslo, saj se z njim lahko, enako kot upravičeni uporabnik, prijavi v IKS.

Da bi napadalcu, ki je lahko prisluškuje komunikaciji preprečili, da bi ukradel geslo, se geslo ne sme prenašati preko omrežja. Identifikacijo brez prenosa ključa preko omrežja omogočajo različni protokoli identifikacije, ki bodo obravnavani v nadaljevanju.

#### Uveljavljanje varnostne politike gesel

Program za preverjanje moči gesel na strežniku lahko zavrne gesla, ki ne zadostujejo minimalnim zahtevam varnostne politike IKSa. Na ta način uveljavlja pravila za gesla, ki so sprejeta z varnostno politiko.

#### Preprečevanje preizkušanja gesel

Preizkušanje gesel je mogoče preprečiti n več načinov. Avtomatsko preizkušanje velikega števila gesel je mogoče preprečiti tako, da mora po neuspešni prijavi preteči nekaj časa, preden se lahko uporabnik spet prijavi v sistem. Ta čas se lahko ob vsaki neuspešni prijavi tudi podaljša.

Po določenem številu neuspešnih prijav nekateri sistemi blokirajo geslo, tako da postane neveljavno. Blokiranje gesel ni najboljša rešitev, ker napadalcu omogoča, da upravičenim uporabnikom onemogoči dostop do sistema, tako da s zaporednim preizkušanjem napačnih gesel blokira njihova gesla.

Drug način proti preizkušanju velikega števila gesel je, da mora uporabnik pri vsaki prijavi v sistem rešiti tudi neko uganko, ki je lahka za človeka, za avtomatsko preizkušanje gesel pa predstavlja problem. Pogosto je v ta namen uporabljena neka slika s tekstom, ki je zapisan na tak način, da bi ga bilo zelo težko prepoznati s programom z avtomatsko prepoznavo črk. Lahko pa je ta uganka pri vsaki drugačnega tipa, kot na primer preprost matematičen račun, preprosto vprašanje, kot kateri dan v tednu je danes, ali kaj podobnega.

#### Shranjevanje gesel na strežniku

Pri postopku identifikacije z geslom mora strežnik poznati gesla vseh uporabnikov, ki so upravičeni do prijave na strežnik. Gesla morajo biti zato na strežniku shranjena. Če bi bila shranjena v obliki čistopisa, bi lahko napadalec, ki bi uspel na nek način vdreti na strežnik, pridobil gesla vseh uporabnikov.

Gesla se morajo zato na strežniku shranjevati tako, da so zaščitena pred razkritjem Gesla so na strežniku lahko šifrirana. Kadar so gesla šifrirana, mora bit nekje na strežniku shranjen tudi ključ za njihovo dešifriranje, kar je šibka točka takega načina shranjevanja.

Boljši način shranjevanja gesel je, da so namesto gesel na strežniku shranjeni zgolj njihovi prstni odtisi. Pri identifikaciji strežnik izračuna prstni odtis gesla, ki mu ga je posredoval uporabnik in ga primerja s shranjenim prstnim odtisom.

Pri takem shranjevanju so problematična šibka gesla. Če se napadalec uspe dokopati do datoteke s prstnimi odtisi, lahko z napadom grobe sile preizkuša gesla, tako da generira gesla in računa njihove prstne odtise. Te odtise potem primerja s shranjenim odtisi na strežniku. Lahko pa tudi preizkuša gesla iz tabele pogosto uporabljanih gesel.

Trenutno so dostopni programi, ki v nekaj minutah odkrijejo vsa do vključno 6 znakov dolga gesla, v nekaj urah pa tudi vsa do vključno 8 znakov dolga gesla.

### 4.2 Protokoli za identifikacijo

Pri osnovni identifikaciji z geslom (PAP – Password Authentication Protocol) se geslo prenaša preko omrežja oziroma ne-varnega kanala v obliki čistopisa, kar omogoča krajo gesel napadalcem, ki lahko prestrezajo sporočila.

Da bi to preprečili, potrebujemo ustrezne protokole za identifikacijo, pri katerih se geslo uporabi samo enkrat ali pa se preko kanala sploh sploh ne prenaša. Za tako identifikacijo uporabljamo ustrezne kriptografske funkcije. Zasnovana je na uporabi simetričnega in/ali asimetričnega šifriranja in/ali zgostitvenih funkcij.

Večinoma se pri kriptografskih funkcijah, ki potrebujejo simetrični ključ, ta na nek način izračuna iz gesla. Običajno se za ključ uporabi kar prstni odtis gesla, ki se izračuna z ustrezno kriptografsko zgostitveno funkcijo.

Ključ za asimetrične šifre se ne more izračunati iz gesla. Kadar se uporablja pri identifikaciji asimetrična šifra, je geslo običajno potrebno samo za dostop do privatnega ključa, ki je šifriran z nekim simetričnim šifrirnim postopkom in shranjen na delovni postaji ali pametni kartici.

#### 4.2.1 Identifikacija z enkratno uporabo gesla

Pri tem protokolu ponudnik storitve ustvari veliko število gesel. Ta gesla mora na nek varen način dostaviti uporabniku (pametna kartica, USB ključ, ....). Tak način omogoča omejeno število prijav, dokler dokler uporabnik ne porabi vseh gesel.

Da ne bi bilo potrebno gesel shranjevati na strežniku, se za tvorjenje gesel uporabi zgostitvena funkcija. Postopek izdelave gesel in identifikacije poteka po naslednjem postopku:

- Strežnik ustvari naključno geslo $\mathbf{G}_0$.
- Iz tega gesla izračuna $N$ prstnih odtisov po rekurzivnem postopku:

$$\mathbf{G}_n = h(\mathbf{G}_{n-1})$$

  kjer je $h$ ustrezna zgostitvena funkcija.
- Uporabnik na nek varen način pridobi prvih $N$ gesel ($\mathbf{G}_0$ do $\mathbf{G}_{N-1}$). Lahko pa pridobi zgolj $\mathbf{G}_0$ in si sam izračuna naslednjih $N-1$ gesel po enakem postopku kot strežnik.
- Strežnik shrani samo geslo $G_N$.
- Za prijave na strežnik uporabnik uporablja gesla po vrsti od zadnjega proti prvemu, od $G_{N-1}$ do $G_0$.
- Strežnik ob prijavi izračuna prstni odtis prejetega gesla in ga primerja s shranjenim geslom (pri prvi prijavi z je to$G_N$). Če sta enaka potrdi identiteto uporabnika, in shranjeno geslo nadomesti z novim geslom, ki ga je dobil od uporabnika.
- Postopek se lahko ponavlja, dokler uporabnik ne uporabi vseh $N$ gesel.

Mož na sredini, ki prestreže neko geslo $\mathbf{G}_n$, iz njega ne more izračunati gesla $\mathbf{G}n-1$ za katerega velja $\mathbf{G}_n = h(\mathbf{G}_{n-1})$, ki bo veljavno za naslednjo prijavo.

Prednost takega načina identifikacije je tudi v tem, da je na strežniku shranjeno samo geslo od prejšnje prijave, ki pa napadalcu, tudi če pride do njega, ne omogoča naslednje prijave. To geslo ima podobno vlogo kot javni ključ v asimetričnih postopkih, torej je lahko vsem poznano. Razlika je to geslo samo za enkratno uporabo in se po vski uporabi zamenja z novim.

Varianta tega postopka je, da gesla ustvari uporabnik in strežniku dostavi samo geslo $G_N$ na tak način, da strežnik zagotovo ve, da to geslo pripada temu uporabniku.

### 4.2.2 Identifikacija z izzivom in odgovorom

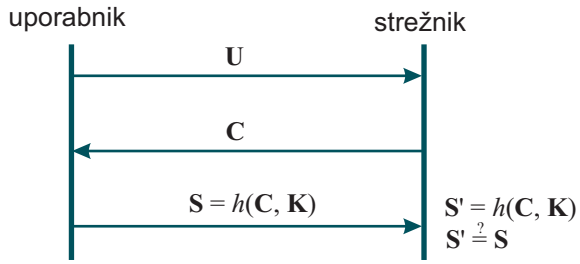
Veliko protokolov za identifikacijo uporablja identifikacijo na osnovi izziva in odgovora (CHAP – CHallenge and Response Protocol). Osnova je pri vseh teh protokolih enaka, razlikujejo se predvsem v formatu sporočil in uporabljenih zgostitvenih funkcijah.

Vzemimo, da se želi uporabnik prijaviti na nek strežnik in se mora zato identificirati. Identifikacija bi potem potekala po naslednjem postopku:

- Uporabnik pošlje strežniku zahtevo za prijavo $\mathbf{U}$, ki vsebuje tudi njegovo uporabniško ime.
- Strežnik, ustvari sporočilo, ki med drugim vsebuje tudi nek naključni čistopis $\mathbf{C}$ in ga pošlje kot izziv uporabniku.
- Uporabnik naredi izvleček tega sporočila z zgostitveno funkcijo tipa MAC $\mathbf{S} = h(\mathbf{C}, \mathbf{K})$, to je zgostitveno funkcijo, ki potrebuje tajni ključ. Kot ključ $\mathbf{K}$ lahko uporabi prstni odtis svojega gesla.

- Strežnik, ki pozna geslo uporabnika oziroma njegov prstni odtis, lahko na isti način izračuna zgostitveno funkcijo sporočila $\mathbf{S}' = h(\mathbf{C}, \mathbf{K})$, ki ga je poslal uporabniku. Rezultat primerja z odgovorom, ki ga je dobil od uporabnika. Če odgovor $\mathbf{S}$ enak $\mathbf{S}'$ , potem je uporabnik nekdo, ki pozna tajno geslo.

Postopek je prikazan na sliki 4.1. Po ne-varnem kanalu se prenese samo zahtevek za prijavo z




Slika 4.1 – Identifikacija na osnovi izziva in odgovora

uporabniškim imenom, naključen čistopis in izvleček tega čistopisa. Iz nobenega med njimi ni mogoče ugotoviti ključa, s katerim je bil ta izvleček narejen. Ravno tako nihče, ki ne pozna ključa $\mathbf{K}$, ne more zgostiti naključnega čisopisa tako, da bi dobil $\mathbf{S}$ enak $\mathbf{S}'$, ki ga je izračunal strežnik.

Na enak način, kot se uporabnik identificira strežniku, se lahko, na zahtevo uporabnika, tudi strežnik identificira uporabniku. S tem si uporabnik zagotovi, da se prijavlja na pravi strežnik.

Problem pri tem načinu identifikacije je, da mora strežnik poznati tajne ključe vseh uporabnikov, zato mora imeti nekje shranjene. Pri tem protokolu ne zadošča, da ima shranjene prstne odtise ključev, ker potrebuje ključ za stiskanje naključnega sporočila. Ključi so sicer lahko nekje šifrirani, vendar mora imeti strežnik dostop do ključa za njihovo dešifriranje, kar pomeni, da mora biti ta ključ shranjen na strežniku, lahko sicer samo v začasnem pomnilniku.

Taka identifikacija pa še ne zagotavlja varnega prenosa sporočil. Človek v sredini (ang. men in the middle), ki lahko prestreza sporočila, sicer ne more odkriti tajnega ključa, lahko pa bere in tudi neopazno spreminja sporočila.

Za varen prenos sporočil morata oba, strežnik in uporabnik šifrirati vsa nadaljnja sporočila. Za to lahko uporabita isti tajni ključ kot za identifikacijo.

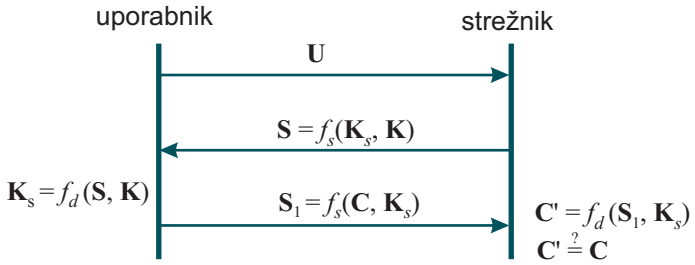
### 4.2.3 Izmenjava sejnega ključa

Pri tem načinu identifikacije, se pri identifikaciji izmenja tudi sejni ključ, to je ključ, ki se uporabi za šifriranje sporočil po uspešni prijavi. Uporaba sejnega ključa, ki se menja z vsako prijavo, je za šifriranje varnejša kot uporaba identifikacijskega ključa. Večkratna uporaba istega ključa namreč povečuje možnosti za njegovo razbijanje. Postopek identifikacije je naslednji:

- Uporabnik pošlje strežniku zahtevo za prijavo $\mathbf{U}$, ki vsebuje tudi njegovo uporabniško ime.
- Strežnik ustvari naključen sejni ključ $\mathbf{K}_s$. Ta ključ šifrira s identifikacijskim ključem uporabnika $\mathbf{K}$. Dobljeni šifropis $\mathbf{S} = f_s(\mathbf{K}_s, \mathbf{K})$ pošlje kot izziv uporabniku.

- Uporabnik s svojim identifikacijskim ključem $\mathbf{K}$, dešifrira sprejeti šifropis in tako pridobi sejni ključ $\mathbf{K}_s = f_d(\mathbf{S}, \mathbf{K})$.
- S sejnim ključem $\mathbf{K}_s$, ki ga je dobil, šifrira neko vnaprej dogovorjeno sporočilo $\mathbf{C}$. Šifropis $\mathbf{S}_1 = f_s(\mathbf{C}, \mathbf{K}_s)$ pošlje na strežnik.
- Strežnik dešifrira sporočilo $\mathbf{S}_1$, ki ga je sprejel od uporabnika, tako, da dobi nazaj čistopis $\mathbf{C}' = f_d(\mathbf{S}_1, \mathbf{K}_s)$. To sporočilo primerja z dogovorjenim sporočilom $\mathbf{C}$. Če sta sporočili enaki, potem je uporabnik avtenticiran.

Potek identifikacije je prikazan na sliki 4.2. Po kanalu se prenaša samo z identifikacijskim ključem




Slika 4.2 – Potek identifikacije z izmenjavo sejnega ključa

šifriran sejni ključ in s sejnim ključem šifrirano sporočilo. Kdor ne pozna tajnega ključa $\mathbf{K}$, ne more priti do sejnega ključa $\mathbf{K_s}$ in zato ne more ustvariti šifropisa $\mathbf{S}_1$, ki bi po dešifriranju s ključem $\mathbf{K_s}$ dalo pravilen čistopis $\mathbf{C}'$.

V neki varianti tega postopka lahko strežnik na začetku skupaj s šifrirnim ključem uporabniku pošlje tudi naključen čistopis (oboje šifrirano s ključem $\mathbf{K}$), ki ga mora potem uporabnik šifrirati s ključem $\mathbf{K}_s$.

Prednost tega postopka je, da vsa nadaljnja komunikacija poteka v šifrirani obliki, šifrirana s sejnim ključem, tako da je nedostopna človeku na sredini. Ostaja pa problem shranjevanja ključev uporabnikov na strežniku.

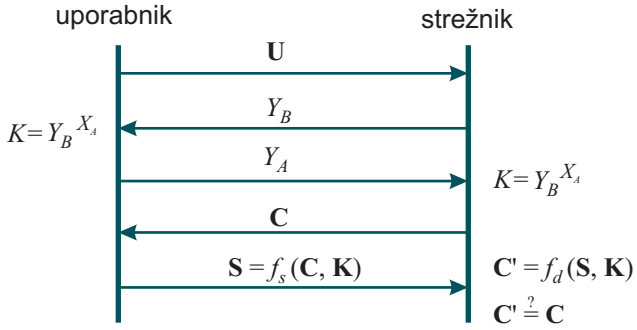
### 4.2.4 Identifikacija z diskretno eksponentno funkcijo

Pri tem postopku zadošča, da strežnik pozna javne ključe uporabnikov. Postopek je pravzaprav enak, kot eksponentna izmenjava ključev, ki je bila opisana v prejšnjem poglavju. Uporabnik ima svoj privatni ključ $X_A$ in javni ključ $Y_A = a^{X_A} mod N$. Podobno ima tudi strežnik Svoj privatni ključ $X_B$ in javni ključ $Y_B = a^{X_A} mod N$. Identifikacija potem poteka po naslednjem postopku:

- Uporabnik pošlje strežniku zahtevo za prijavo $\mathbf{U}$, ki vsebuje njegovo uporabniško ime.
- Uporabnik in strežnik po postopku za eksponentno izmenjavo ključev, ki je opisan v prejšnjem poglavju, ustvarita skupno skrivnost $K = a^{X_A\, X_B}$, ki jo uporabita kot identifikacijski ključ $\mathbf{K}$. Ker se pri eksponentni izmenjavi ta ključ ne prenaša po ne-varnem kanalu, ga poznata samo strežnik in uporabnik.

- Strežnik nato pošlje uporabniku naključen čistopis **C**.
- Uporabnik ta čistopis šifrira z neko simetrično šifro in ključem **K** in šifropis $\mathbf{S} = f_s(\mathbf{C}, \mathbf{K})$ pošlje strežniku.
- Strežnik dešifrira šifropis **S**, ki ga je sprejel od uporabnika. Dobljeni čistopis $\mathbf{C}' = f_d(\mathbf{S}, \mathbf{K})$ primerja s sporočilom **C**. Če sta sporočili enaki, potem je uporabnik avtenticiran.

Postopek je prikazan na sliki 4.3




Slika 4.3 – Identifikacija z diskretno eksponentno funkcijo

Strežniku ni potrebno shranjevati identifikacijskih ključev uporabnikov. Identifikacijski ključ lahko za vsakega uporabnika izračuna sproti, ob prijavi. Vendar pa za to potrebuje svoj privatni ključ, ki pa mora biti shranjen. Če napadalec, ki je uspel vdreti na strežnik, na nek način pride do tega ključa, pa mu to ne pomaga, da bi razkril privatne ključe uporabnikov.

Na strežniku morajo biti shranjeni tudi javni ključi uporabnikov, tako da lahko strežnik z gotovostjo ve, kateremu uporabniku pripada posamezen javni ključ.

Za šifriranje nadaljnjih sporočil se lahko potem uporabi identifikacijski ključ, bolje pa je če si strežnik in uporabnik izmenjata sejni ključ. Za izmenjavo pri tem uporabita identifikacijski ključ.

### 4.2.5 Identifikacija z asimetrično šifro

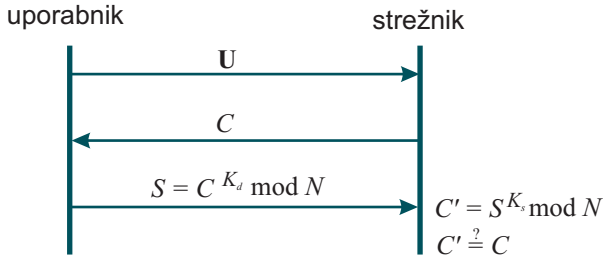
Pri identifikaciji z asimetrično šifro strežniku ni potrebno poznati (imeti shranjenega) nobenega tajnega ključa, da bi izvedel identifikacijo uporabnika. Shranjen mora imeti samo njegov javni ključ.

Postopek identifikacije je tu naslednji:

- Uporabnik pošlje strežniku zahtevo za prijavo **U**, ki vsebuje njegovo uporabniško ime.
- Strežnik pošlje uporabniku naključno število $C$.
- Uporabnik šifrira to sporočilo z neko asimetrično šifro, ki mora biti komutativna, kot na primer RSA. Za šifriranje uporabi svoj privatni ključ $K_d$, ki je pri prenosu šifriranih sporočil namenjen dešifriranju. Dobljeni šifropis $S$ pošlje strežniku.

- Strežnik dešifrira šifropis $S$ z javnim ključem uporabnika $K_s$, ki je pri prenosu šifriranih sporočil namenjen šifriranju. Dobljeno število $C'$ primerja s številom $C$. Če sta sporočili enaki, potem je uporabnik avtenticiran.

Opisani postopek, pri katerem je uporabljena RSA šifra, je prikazan na sliki 4.4.




Slika 4.4 – Identifikacija z asimetrično šifro RSA

Napadalec, ki ne pozna privatnega ključa uporabnika $K_d$ ne more ustvariti šifropisa $S$ tako, da bi ta po dešifriranju z javnim ključem uporabnika $K_s$, dal pravilen čistopis $C'$.

Na strežniku za namen identifikacije uporabnika ni potrebno imeti shranjene nobene tajne informacije. To pa ne velja za primer, ko bi se moral strežnik identificirati uporabniku. V tem primeru mora poznati svoj privatni ključ, saj poteka identifikacija na osnovi nečesa kar strežnik ve. Zato se temu ni mogoče izogniti.

Tudi pri tem postopku identifikacije si lahko za nadaljnjo komunikacijo strežnik in uporabnik izmenjata sejni ključ za nek simetrični šifrirni postopek.

### 4.2.6 Identifikacija s posredovanjem

Kadar se izvaja avtentikacija na osnovi nekega simetričnega postopka pri katerem morata oba udeleženca v komunikaciji poznati tajno geslo, nastopi težava pri distribuciji teh gesel, še posebej, če želimo, da lahko vsak identificira vsakemu, oziroma, da lahko vsak varno komunicira z vsakim. Vsak udeleženec bi moral imeti v tem primeru shranjena gesla vseh ostalih udeležencev. Vseh gesel pri $N$ udeležencih je v tem primeru $N(N-1)/2$.

Težavo lahko rešimo s posredovanjem avtentikacijskega strežnika. Avtentikacijski strežnik mora poznati gesla vseh uporabnikov. Vzemimo, da se želita Alenka in Boris vzajemno identificirati in vzpostaviti varno povezavo s simetričnim šifrirnim postopkom. Kadar komunikacijo začne Alenka, poteka postopek identifikacije in izmenjave ključa na naslednji način:

- Alenka ustvari naključen sejni ključ $\mathbf{K}$ in pošlje strežniku sporočilo $\mathbf{M}_1$ z zahtevo za povezavo z Bobom. V sporočilu je njeno uporabniško ime v čistopisu, ki mu je pripet šifropis uporabniškega imena Borisa, časovni žig $\mathbf{T}_A$ in sejni ključ $\mathbf{K}$. Časovni žig, to je točen čas pošiljanja sporočila, prepreči napadalcu, da bi kdaj kasneje uporabil to sporočilo in poslal zahtevek v imenu Alenke.

$$\mathbf{M}_1 = \{\text{"Alenka"} \ , f_s(\{\text{"Boris"}, \mathbf{T}_A, \mathbf{K}\}, \mathbf{K}_A)\}$$

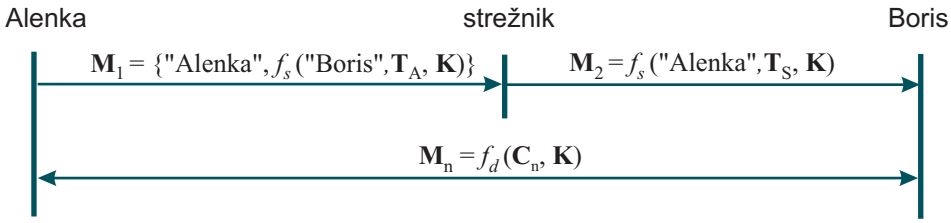
kjer je $f_s$ šifrirna funkcija neke simetrična šifre in $\mathbf{K}_A$ Alenkin tajni ključ.

- Strežnik poišče v svoji zbirki Alenkin tajni ključ $\mathbf{K}_A$ in z njim dešifrira šifrirani del sporočila. Po dešifriranju preveri časovni žig in uporabniško ime. Če je oboje v pravilno, sklepa, da je pošiljatelj zares Alenka, ki edina pozna ključ $K_A$.
- Strežnik nato v svoji zbirki poišče Borisov tajni ključ $K_B$ in mu pošlje sporočilo $\mathbf{M}_2$:

$$\mathbf{M}_2 = f_s(\{\text{"Alenka"} \ , \mathbf{T}_S, \mathbf{K}\}, \mathbf{K}_B)$$

  kjer je s $\mathbf{T}_S$ označen časovni žig strežnika.
- Boris dešifrira sprejeto sporočilo, preveri časovni žig pridobi Alenkino uporabniško ime in sejni ključ $\mathbf{K}$.
- Ker sedaj oba poznata sejni ključ $\mathbf{K}$, lahko vzpostavita šifrirano povezavo.

Potek identifikacije in izmenjave ključev je prikazan na sliki 4.5




Slika 4.5 – Identifikacija in izmenjava sejnega ključa s posredovanjem avtentikacijskega strežnika

Pri identifikaciji s posredovanjem je vzpostavitev komunikacije zelo preprosta. Alenki za to ni potrebno poznati Borisovega tajnega ključa. Komunikacija se vzpostavi že po enem sporočilu, ki ga Alenka pošlje na strežnik. Pri tem protokolu je potrebno, da je strežnik zaupanja vreden, torej marata tako Alenka kot Boris zaupati strežniku.

Ker se sejni ključ prenaša samo v šifrirani obliki, napadalec, ki ne pozna Alenkinega niti Borisovega tajnega ključa, ne more priti do njega. Ravno tako se ne more poslati strežniku zahtevka v Alenkinem imenu, ker ne more šifrirati zahtevka z Alenkinim ključem. Ker strežnik zahtevek zavrne, če ima enak časovni žig kot prejšnji zahtevek, tudi ne more ponovno poslati Alenkinega zahtevka, ki ga je prestregel.

V drugih variantah protokola s posredovanju lahko poteka komunikacija med Alenko in strežnikom na osnovi izziva in odgovora, ki ne potrebuje časovnega žiga, vendar zahteva prenos več sporočil.

Podobno pa se lahko prijava uporabnika na strežnik izvede z enim samim sporočilom, ki vsebuje uporabniško ime ter šifriran časovni žig in sejni ključ.

### 4.2.7 Nekaj standardiziranih protokolov

Standardizirani protokoli za identifikacijo so zasnovani na opisanih postopkih. Standardi predvsem predpisujejo konkretno izvedbo, to je vrsto in natančen format sporočil in uporabljene

šifre in zgostitvene funkcije. Nekateri standardizirani protokoli združujejo identifikacijo tudi z avtorizacijo in obračunavanjem, tako da sama identifikacija predstavlja le del standarda ali priporočila. Omenimo le nekaj standardov, ki so v široki uporabi.

### RADIUS

RADIUS (ang. Remote Authentication Dial In User Service) protokol za oddaljen dostop uporabnikov do omrežja omogoča poleg identifikacije tudi avtorizacijo in zaračunavanje storitev, ki delujejo po načelu odjemalec-strežnik. Omogoča tudi identifikacijo in avtorizacijo pri dostopu do lokalnih omrežij in nudi tudi informacije za avtomatsko nastavljanje parametrov odjemalca.

Identifikacija poteka s posredovanjem Radius avtentikacijskega strežnika. Osnovnemu standardu je podrejenih veliko podrejenih standardom, ki predpisujejo posamezne področja, ki jih pokriva Radius. Zadnja verzija protokola Radius (RFC 2865) je iz leta 2000, zato je že nekoliko zastarel in se večinoma nadomešča z novejšim protokolom Diameter.

### Diameter

Diameter je je novejša različica protokola RADIUS. Ime je dobil na osnovi besedne igre. Premer (ang diameter) kroga je dvakrat večji od njegovega radia (ang. radius). Diameter torej tako kot RADIUS protokol za identifikacijo, avtorizacijo in zaračunavanje, vendar uporablja bolj zanesljiv transportni protokol (TCP namesto UDP), podporo varnemu prenosu na osnovi protokolov IPsec in TLS, boljšo podporo gostovanju v drugih omrežjih, mobilnost v IP omrežjih, možnost dogovarjanja o načinu identifikacije, sporočanje napak in drugo. Zadnja različica protokola Diameter (RFC 6733) je bila izdana oktobra leta 2012. Protokol Diameter sedaj uporablja večina ponudnikov Internetnih storitev (ang. Internet service providers - ISP) kot tudi večina mobilnih operaterjev.

### Kerberos

Kerberos je protokol za identifikacijo v računalniških omrežjih. Ime je dobil po Kerberosu, triglavemu psu iz grške mitologije, ki čuva vrata v podzemlje. Protokol deluje na osnovi vstopnic, ki omogočajo vozliščem v omrežju, da medsebojno dokazujejo svojo identiteto. Identifikacija je zasnovana na izmenjavi sejnih ključev s posredovanjem avtentikacijskega strežnika. Sporočila vsebujejo časovni žig, ki preprečuje napade s ponavljanjem sporočil.

Protokol omogoča prijavo uporabnikov in vzajemno identifikacijo pri storitvah tipa odjemalec - strežnik, kot tudi medsebojno identifikacijo posameznih vozlišč vključenih v omrežje. Kerberos se uporablja za identifikacijo v Microsoft Windows domenskih omrežjih preko storitve aktivnega imenika (ang. active directory).

### HTTPS

HTTPS (ang. hyper text transfer protocol - secure) je varna različica protokola HTTP. Omogoča identifikacijo spletnega strežnika na osnovi overjenega spletnega potrdila. Po overjanju se vz-

postavi varna (šifrirana) povezava med spletnim brskalnikom in strežnikom po protokolu SSL (Secure Socket Layer) oziroma novejši različici TLS (Transport Layer Security). Ko je vzpostavljena varna povezava, ta omogoča preprosto identifikacijo uporabnika z geslom po protokolu PAP.

#### NTLM

NTLM (ang. NT lan manager) je protokol ki se uporablja oziroma se je uporabljal za identifikacijo odjemalcev, avtentikacijo sporočil in varen prenos v omrežjih Microsoft Windows. Identifikacija poteka na osnovi izziva in odgovora. Namesto gesel se uporabljajo prstni odtisi gesel. Ti so shranjeni na strežniku, in so enakovredni geslom. Nekdo ki pozna te prstne odtise, se lahko, ne da bi poznal lažno identificira.

NTLM sloni na kriptografsko šibkih postopkih, kot sta to DES in MD4. Uporaba NTLM identifikacije ni priporočljiva. V widows omrežjih je NTLM sedaj večinoma nadomeščen s protokolom Kerberos, čeprav se še vedno uporablja pri dostopu odjemalcev do strežnikov, ki niso v domeni in kadar dostopa odjemalec do strežnika preko IP naslova.

#### EAP

EAP (ang. extensible authentication protocol) je razširljiv protokol, ki predpisuje format sporočil za različne postopke identifikacije. Različni protokoli za identifikacijo, kot na primer RADIUS in Diameter) lahko vključijo EAP sporočila v svoja sporočila, kot njihov sestavni del, kar imenujemo enkapsulacija. V okviru EAP je vključenih 14 različnih protokolov identifikacije, ki so standardizirani ločeno.

EAP se pogosto uporablja za identifikacijo v brezžičnih omrežjih in je vključen v protokola WPA in WPA2, ki se uporablja v brezžičnem lokalnem omrežju WiFi.

#### SASL

SASL (ang. simple authentication and security layer) je okvir v katerega je vključenih več postopkov za identifikacijo in avtentikacijo v internetnih protokolih. SASL se uporablja v različnih internetnih protokolih, kot je to na primer IMAP za oddaljen dostop do elektronske pošte, SMTP za prenos elektronske pošte, ACPAP protokol za nastavitev aplikacij, LDAP za dostop do imenikov in drugih.

Poleg zgoraj omenjenih standardnih protokolov obstaja še cela vrsta drugih protokolov za identifikacijo, saj skorajda vsaka storitev v elektronskem dostopu in elektronskem poslovanju uporablja svoj specifičen protokol.

### 4.3 Identifikacija z identifikacijsko napravo

Identifikacija z identifikacijsko napravo je identifikacija na osnove tega kar nekdo ima, za razliko od identifikacije z geslom, ki je identifikacija na osnov tega kar nekdo ve.

Kot identifikacijska naprava pogosto služi pametna kartica, lahko pa je to tudi kakšna druga naprava, kot je to na primer namenska identifikacijska kartica s svojo tipkovnico ali pa tudi ključ za daljinsko odpiranje avtomobila.

Načeloma se za identifikacijo z identifikacijsko napravo uporabljajo enaki postopki kot za identifikacijo brez nje. Na identifikacijski napravi so lahko shranjeni sejni ključi za enkratno uporabo ključa, avtentikacijski ključ za simetrične postopke identifikacije ali pa privatni ključ asimetričnih postopkov identifikacije.

Pri dobro načrtovanih identifikacijskih napravah tajni ključ nikoli ne zapusti naprave, tako da tisti, ki naprave nima, ne more izvesti lažne identifikacije. To je velika prednost uporabe take naprave, saj naprave z vdorom v omrežje ali z vdorom na računalnik, ni mogoče ukrasti.

Ker je možno,da tako napravo izgubimo, je dostop do tajne informacije na napravi dodatno zaščiten z nekim geslom ali PIN kodo. V tem primeru potrebujemo za identifikacijo (nekaj kar imamo) poleg tega pa moramo poznati tudi tajno geslo (nekaj kar vemo). Tak postopek identifikacije zato imenujemo tudi hibrif+dna identifikacija, Ta je mnogo bolj varna kot preprosta identifikacija z geslom.

Geslo (PIN kodo) lahko vnesemo na računalniku ali drugi napravi, na primer na bankomatu, na katero priključimo identifikacijsko napravo. Pri napravah z lastno tipkovnico, vnašamo geslo s pomočjo te tipkovnice.

Naprave z lastno tipkovnico nudijo bistveno večjo stopnjo varnosti, kot naprave brez nje. Če vnašamo geslo preko tipkovnice računalnika, lahko nek zlonameren program, ki je nameščen na računalniku, prestreže tajno geslo. Ta problem je še posebej izrazit pri bančnih karticah. Če bančno kartico vstavimo v lažen bankomat in vtipkamo PIN kodo smo napadalcu razkrili PIN kodo. Lažni bankomat kodo zavrne, čeprav je ta pravilna, bančno kartico zadrži. Napadalec tako pridobi kartico in pin kodo, tako da lahko brez težav dviguje denar na pravih bankomatih, dokler lastnik kartice ne prekliče.

Če bi imela bančna kartica svojo tipkovnico, napadalec ne bi mogel pridobiti PIN kode ali daljšega tajnega gesla.

## 4.4 Biometrična identifikacija

Biometrična identifikacija je identifikacija na osnovi tega kar nekdo je. Identifikacija lahko poteka na osnovi prepoznave prstnega odtisa človeka, njegove zenice, njegovega glasu ali nekega drugega značilnega biološkega parametra.

Zgolj biometrična identifikacija ni najbolj zanesljiva. Človeški glas je mogoče posneti, ravno tako je mogoče pridobiti prstni odtis in/ali sliko zenice in to uporabiti na tak način, da se prevara napravo za preverjanje biometričnih značilnosti.

Boljše (dražje) naprave za preverjanje biometričnih značilnosti so sicer narejene tako, da to v čim večji meri onemogočajo. Naprava za prepoznavo prstnega odtisa lahko hkrati meri tudi upornost, kapacitivnost in temperaturo. Naprava za prepoznavo glasu lahko zahteva, da uporabnik izgovori vsakič nek drug stavek, tako da ni mogoče vnaprej izdelati ustreznega pos-

netka glasu. Naprava za prepoznavo zenice lahko spremlja širjenje in oženje zenice in podobno.

Kljub vsem tem dodatnim ukrepom, pa se identifikacija na osnovi biometričnih lastnosti najpogosteje uporablja zgolj za povečanje varnosti identifikacije na osnovi identifikacijske naprave in tajnega gesla.

# 5 Elektronski dokumenti

Že v uvodnem poglavju smo povedali (razdelek 1.1), da so dokumenti podatki s posebnim statusom in določeno pravno formalno veljavo, ki je odvisna od vrste dokumenta. Dokumenti morajo izpolnjevati določene vidike celovitosti podatkov. Za večino dokumentov velja, da morajo biti verodostojni, avtentični, neovrgljivi in časovno opredeljeni. Za določene dokumente pa je zahtevana tudi tajnost in trajnost.

Trajnost dokumentov zagotavljamo z shranjevanjem na zanesljivih pomnilniških medijih in varnostnimi kopijami, ki so po potrebo shranjene na geografsko oddaljenih lokacijah. Pomembnejše lastnosti naprav za shranjevanje podatkov so naštete v razdelku 2.1.3.

Zaupne oziroma tajne dokumente šifriramo z ustreznimi šiframi Zahtevana moč šifre je odvisna od stopnje zahtevane tajnosti. O moči šifer in šifrirnih ključev smo govorili v poglavju o šifriranju. Zahtevana stopnja tajnosti dokumentov mora biti opredeljena v varnostni politiki.

Verodostojnost, avtentičnost in neovrgljivost klasičnih dokumentov lahko zagotavljamo z lastnoročnim podpisom. Za zagotavljanje časovne opredeljenosti pa potrebujemo neko notarsko službo, ki potrjuje čas nastanka dokumenta, oziroma, da je ob določenem času dokument že obstajal. Notarska služba potrjuje dokument z z lastnoročnim podpisom, žigom in pečatom.

Na podoben način lahko verodostojnost in avtentičnost elektronskih dokumentov zagotavljamo digitalnim podpisom, medtem ko za zagotavljanje neovrgljivosti in časovne opredeljenosti tudi tu potrebujemo neke vrste notarsko službo.

## 5.1 Digitalen podpis

Da bi lahko elektronske dokument digitalno podpisovali je najprej potreben splošen dogovor o pomenu digitalnega podpisa. Da pa bi imel digitalni podpis tudi pravno veljavo, mora biti tudi zakonsko opredeljen.

Pomen digitalnega podpisa je enak pomenu lastnoročnega podpisa. S podpisom podpisnik potrjuje, da se strinja z vsebino dokumenta in da prevzema vse odgovornosti, ki iz tega izhajajo.

Digitalni podpis je nek podatek (niz znakov ali bitov), ki ga dodamo elektronskemu dokumentu. Da lahko tak niz služil kot podpis mora imeti naslednje lastnosti:

1. Kakršna koli, tudi minimalna, sprememba dokumenta mora spremeniti tudi digitalni podpis.

2. Praktično nemogoče mora biti spremeniti dokument tako, da ostane podpis nespremenjen.

3. Praktično nemogoče mora biti ustvariti dva dokumenta z enakim podpisom.

4. Le nekdo, ki pozna neko tajno informacijo (tajni ključ podpisnika),lahko podpiše dokument.

5. Skladnost podpisa z dokumentom in identiteto podpisnika je mogoče preveriti.

6. Za preverjanje podpisa ni potrebno poznavanje tajnega ključa.

7. Iz podpisa mora biti razvidna in nedvomna identiteta podpisnika.

Prve tri lastnosti so potrebne za zagotavljanje verodostojnosti dokumenta, zadnje štiri pa za zagotavljanje avtentičnosti.

### 5.1.1 Izvedba digitalnega podpisa

Za izvedbo digitalnega podpisa uporabimo kriptografsko zgostitveno funkcijo, kot na primer SHA-2 in asimetrično šifro, ki morata biti postopka šifriranja in dešifriranja zamenljiva. To pomeni, da lahko čistopise ki jih šifriramo z javnim ključem, dešifriramo s privatnim ključem, čistopise ki jih šifriramo s privatnim ključem, pa lahko dešifriramo z javnim ključem. Taka šifra je na primer RSA.

Vzemimo, da želi Alenka digitalno podpisati dokument, ki je v obliki čistopisa $\mathbf{C}$:

- Alenka najprej izračuna prstni odtis $\mathbf{PO}$ čistopisa $\mathbf{C}$

$$\mathbf{PO} = h(\mathbf{C})$$

- Digitalni podpis $\mathbf{DP}$ dobi tako, da dobljeni prstni odtis šifrira s svojim privatnim ključem $\mathbf{K_p}$.

$$\mathbf{DP} = f_s(\mathbf{PO}, \mathbf{K_p})$$

- Alenka lahko potem podpis doda na konec dokumenta, v nekem predpisanem formatu, tako da je jasno razvidno, kje je konec čistopisa. Digitalni podpis pa lahko da tudi kot prilogo dokumentu, kot na primer prilogo elektronski pošti.

S tem, ko je uporabila kriptografsko zgostitveno funkcijo za izračun prstnega odtisa, je Alenka zadostila prvim trem zahtevanim lastnostim digitalnega podpisa, ki zagotavljajo verodostojnost dokumenta. Ker je za šifriranje uporabila svoj privatni ključ, ki ga pozna le sama, je zadostila tudi četrti zahtevi.

### 5.1.2 Preverjanje digitalnega podpisa

Vzemimo, da želi Boris preveriti digitalni podpis dokumenta, ki ga je podpisala Alenka. Zato mora poznati Alenkin javni ključ $\mathbf{K_j}$ ki je par njenemu privatnemu ključu $K_p$. Obenem mora tudi z gotovostjo vedeti, da javni ključ $\mathbf{K_j}$ res pripada Alenki. Podpis potem preveri po naslednjem postopku:

- Izračuna prstni odtis čistopisa **C**, ki ga je prejel od Alenke.

$$\mathbf{PO} = h(\mathbf{C})$$

- Z Alenkinim javnim ključem $\mathbf{K_j}$ Dešifrira digitalni podpis **DP**:

$$\mathbf{PO'} = f_d(\mathbf{DP}, \mathbf{K_j})$$

- Primerja **PO** z **PO′**. Če sta enaka, potem je podpis preverjen, če nista, pa je podpis napačen, ali pa je bil dokument po podpisu spremenjen.

Ker lahko Boris preveri podpis, ne da bi poznal Alenkin privatni ključ, tako narejen podpis zadošča tudi peti in šesti zahtevi. Ker je javni ključ $\mathbf{K_j}$ par z Alenkinim privatnim ključem $\mathbf{K_p}$, lahko le Alenka, ki pozna svoj privatni ključ, prstni odtis šifrira tako, da dobi Boris po dešifriranju z njenim javnim ključem pravo vrednost. Zato Boris ve, da je dokument podpisala Alenka, s čimer je zadoščeno tudi zadnji zahtevi, da mora biti iz podpisa nedvomna identiteta podpisnika.

### 5.1.3 Primerjava lastnoročnega in digitalnega podpisa

Tako lastnoročni kot digitalni podpis služita zagotavljanju verodostojnosti in avtentičnosti nekega dokumenta. Podpisnik v obeh primerih potrjuje, da se strinja z vsebino dokumenta in da za to prevzema vso odgovornost, ki iz te vsebine izhaja. Vendar pa obstajajo tudi razlike med digitalnim in lastnoročnim podpisom, ki se jih moramo zavedati:

- Z lastnoročnim podpisom podpisan dokument je pogosto mogoče spremeniti tako, da se to ne opazi. To še posebej velja za dokumente, ki niso napisani tako, da bi bile take spremembe otežkočene. Če so v dokumentu prazne vrstice, prazen prostor med podpisom in vsebino dokumenta, prazen prostor na koncu vrstice in podobno, je mogoče na te prostore dopisati tekst, ki spremeni vsebino dokumenta. Če je za pisanje dokumenta uporabljeno črnilo, ki ga je mogoče izbrisati, se lahko tekst nadomesti z novim tekstom. Če je obsega dokument več strani, ki so preprosto spete, podpisana pa je zgolj zadnja stran, je mogoče ostale strani v dokumentu zamenjati ali pa dodati nove strani.

  Dokumenta, ki je podpisan z digitalnim podpisom ni mogoče spreminjati, saj že minimalna sprememba dokumenta zahteva popolnoma drugačen digitalni podpis.

- Lastnoročni podpis je dokaj enostavno ponarediti, medtem ko je pri digitalnem podpisu, če so uporabljeni kriptografsko močni postopki, ponarejanje praktično neizvedljivo.

- Če pa bi nekomu, ki bi našel šibkost v postopku digitalnega podpisovanja, uspelo ponarediti digitalni podpis, na noben način ne bi bilo mogoče ločiti pravega in ponarejenega podpisa.

  Pri ponarejenem lastnoročnem podpisu lahko z natančnejšim grafološkim pregledom tudi kasneje ugotovimo, da gre za ponaredek.

Lastnoročni podpis je torej lahko delno ali slabo ponarejen, medtem ko je digitalni podpis lahko samo popolnoma ponarejen ali pa sploh ni.

- Lastnoročni podpis imamo praviloma en sam. Lahko ga sicer zamenjamo, vendar določene grafološke značilnosti pisave ostajajo in jih je zelo težko spremeniti.

  Digitalnih podpisov imamo lahko več. Za različne namene lahko uporabljamo različne digitalne podpise.

- Lastnoročnega podpisa nam nihče ne more ukrasti in ga tudi ne moremo nekomu odstopiti, tudi če bi to želeli.

  Po drugi strani nam lahko digitalni podpis, oziroma naš privatni ključ, ki ga potrebujemo za podpisovanje, nekdo ukrade, lahko ga izgubimo ali pa ga damo nekomu prostovoljno. Dokumentov, ki jih je nekdo drug podpisal z našim podpisom ni mogoče ločiti od dokumentov, ki smo jih podpisali sami.

Iz primerjave lahko vidimo, da digitalni podpis skoraj v vseh pogledih prekaša lastnoročni podpis, vendar pa ima eno zelo pomembno slabost, to je, da ni neodtuljiv, kar pomeni, da ga lahko nekdo ukrade oziroma, da ga lahko nekomu zaupamo. Z ozirom na to slabost je izrednega pomena izobraževanje uporabnikov. Uporabniki se morajo zavedati vseh nevarnosti, ki jih lahko povzroči neprevidno ravnanje s privatnim ključem, ki ščiti digitalni podpis. Zavedati se morajo, da lahko nekdo drug, ki so mu zaupali podpis, v njihovem imenu naredi vse, kar podpisovanje s tem podpisom omogoča. To je lahko dvig denarja z njihovega računa, podpis škodljive pogodbe ali zadožnica za poljuben znesek, ki se glasi na pravega podpisnika.

V podjetjih se kaj lahko zgodi, da direktor, ki mora rutinsko podpisovati celo vrsto dokumentov, za ta namen zaupa tajni ključ nekemu sodelavcu, na primer svoji tajnici. S tem pa da tajnici tudi možnost, da v njegovem imenu podpiše kakršen koli dokument, brez možnosti, da bi kasneje dokazal, da tega dokumenta ni podpisal on.

Pravilen postopek v takem primeru je, da direktor pooblasti sodelavca za podpisovanje določenih dokumentov. Pri tem ostane nedvomno jasno, kdo je te dokumente dejansko podpisal, oziroma kdo je za to neposredno odgovoren. Direktorju še vedno ostaja objektivna odgovornost, ki izhaja iz pooblastila.

Digitalni podpis izgubimo, ga predamo drugi osebi ali pa nam ga lahko ukradejo. Zato je izrednega pomena, kako ravnamo s privatnim ključem, ki omogoča digitalno podpisovanje. Tega ključa ne smemo nikoli shranjevati v obliki čistopisa, najbolje pa je če je shranjen zgolj na neki idntifikacijski napravi.

## 5.2 Avtentičnost javnega ključa

Ko Boris preverja digitalni podpis Alenke, ga preverja z njenim javnim ključem. Zato mora biti prepričan, da javni ključ resnično pripada Alenki. Ključ mora biti avtentičen. Avtentičnost

ključa je pomembna tako pri preverjanju digitalnega podpisa kot tudi pri izmenjavi sejnih ključev s pomočjo asimetrične šifre.

Boris je lahko prepričan, da javni ključ pripada Alenki

- če mu ga je Alenka osebno izročila,
- če se je o tem prepričal po enem ali več vzporednih kanalov, kot na primer po telefonu in e-pošti,
- če ima Alenka objavljen svoj javni ključ na svoji varni (https) spletni strani
- če je ključ objavljen v nekem verodostojnem imeniku, ali
- če je to potrdil nekdo, ki je vreden zaupanja.

Za preverjanje ključa se običajno uporabi njegov prstni odtis. Alenka lahko prstni odtis svojega javnega ključa pošlje Borisu po e-pošti, obenem pa mu ga sporoči tudi po telefonu. Boris izračuna prstni odtis Alenkinega javnega ključa in ga primerja s prstnim odtisom, ki mu ga je sporočila Alenka. Če sta enaka je ključ avtentičen.

Boris pa lahko zaupa tudi enemu ali več svojih prijateljev, ki potrdijo, da ključ res pripada Alenki. Če Boris že ima javne ključe teh prijateljev, ni potrebno, da mu to potrjujejo osebno. Zadošča, da potrdilo, da ključ pripada Alenki, digitalno podpišejo in ga pošljejo Borisu.

Phil Zimmerman, avtor programa Pretty Good Privacy (PGP), ki je prvi omogočil širši javnosti, da lahko uporabljala šifrirano elektronsko pošto in digitalno podpisovanje, si je za avtentikacijo javnih ključev zamislil mrežo zaupanja. V PGP ima Boris shranjenje javne ključe drugih uporabnikov v tako imenovanem obroču (ang. key ring). Ključe, ki so mu jih uporabniki izročili osebno, podpiše s svojim digitalnim podpisom. Poleg tega pa določi v obroču tudi uporabnike, ki jim zaupa avtentikacijo drugih ključev. Vsakemu določi tudi stopnjo zaupanja. Od stopnje zaupanja je odvisno, koliko teh uporabnikov mora s svojim digitalnim podpisom overiti javni ključ ključ nekega uporabnika, ki mu ni osebno izročil ključa.

Vzemimo, da je Alenka izročila svoj javni ključ Borisu osebno. Boris ga je dodal v svoj obroč in ga digitalno podpisal, ker ve da zagotovo pripada Alenki. Ker pozna Alenko in ve daj je odgovorna oseba, ji je dodelil najvišjo stopnjo zaupanja. Ko želi Boris varno komunicirati z Mihatom, vendar nima njegovega javnega ključa, prosi Alenko, če mu lahko da Mihatov javni ključ. Alenka mu pošlje digitalno podpisan Mihatov ključ. Boris preveri podpis in, če je podpis pravi, doda ključ v svoj obroč, saj Alenki zaupa, da je to res Mihatov ključ. Ker Mihata ne pozna tako dobro, ali pa ve, da ni zelo odgovoren, mu dodeli nižjo stopnjo zaupanja, na primer zahteva, da nov ključ podpišejo vsaj trije s tako stopnjo zaupanja, preden mu bo zaupal tudi sam. Z večanjem števila ključev v obroču Boris vedno laže pridobiva ključe drugih, ki jih sam morda ni nikoli srečal, vendar lahko na osnovi mreže zaupanja, zaupa da so avtentični.

## 5.3 Digitalna potrdila

Da bi imel nek dokument, na primer pogodba, tudi pravno veljavo, pa ne zadošča, da tisti, ki preverja digitalni podpis z gotovostjo ve, da javni ključ, s katerim preverja veljavnost, resnično pripada osebi, ki je dokument podpisala, temveč mora biti to sposoben tudi dokazati.

Alenka, ki je Borisu osebno izročila svoj javni ključ, tako da Boris zagotovo ve, da ta ključ pripada Alenki, lahko kasneje trdi da ta ključ ni njen. Vzemimo, da je z njim podpisala posojilno pogodbo, po kateri dolguje Borisu določeno vsoto denarja. Ker nima denarja, da bi ga vrnila, zatrjuje, da te pogodbe ni nikoli podpisala, in da javni ključ s katerim Boris preverja njen digitalni podpis, ni njen. Boris ne more nikakor dokazati, da je ta ključ resnično pripada Alenki.

Možna rešitev tega problema je, da Alenka, ko izroči ključ Borisu, obenem tudi lastnoročno podpiše nek dokument, s katerega je razvidno, da ta ključ pripada njej. Tak način pa nujno zahteva osebno izmenjavo vseh javnih ključev, kar pa ni praktično.

Mnogo bolj praktično in tudi s pravnega vidika bolj sprejemljivo je, če uvedemo neko zaupanja vredno notarsko službo (TTP – Trusted Third Party), agencijo za overjanje digitalnih potrdil (CA – Certification Authority), ki je zadolžena za overjanje javnih ključev.

Ker morajo vsi uporabniki zaupati CA obenem pa mora, v primeru spora, CA zaupati tudi sodišče, mora biti dejavnost CA določena z zakonom, CA pa lahko opravlja dejavnost samo na osnovi koncesije. Izjema so CA, ki so namenjene interni uporabi znotraj nekega podjetja ali združbe, vendar podpisi, ki jih je overil tak CA praviloma nimajo pravne veljave.

CA mora izdajati digitalna potrdila po nekem predpisanem postopku. Uporabnik se mora praviloma osebno zglasiti na CA in se identificirati z veljavnim osebnim dokumentom. Večinoma CA izdela par ključev (javni in privatni ključ) in ju izroči uporabniku. Hkrati izdela digitalno potrdilo. Digitalno potrdilo vsebuje podatke o izdajatelju, uporabniku, vrsto javnega ključa in javni ključ uporabnika, datuma začetka in konca veljavnosti ter podatke o uporabljenih kriptografskih postopkih in o namenu uporabe. Digitalno potrdilo je dokument, ki ga podpiše CA s svojim privatnim ključem za podpisovanje digitalnih potrdil. Uporabnik mora ob prejemu ključa in digitalnega potrdila lastnoročno podpisati potrdilo o prejemu. CA potem potrdilo objavi na svojem strežniku javnih ključev.

Na osnovi digitalnega potrdila lahko vsakdo, ki ima javni ključ CA oziroma digitalno potrdilo z javnim ključem CA, preveri avtentičnost vsakega javnega ključa, za katerega je CA izdala potrdilo.

Izdaja digitalnih potrdil trenutno pri nobenem od znanih CA ni čisto korektna. Večina CA izdela par ključev sama in jih preda uporabniku. To pomeni, da tudi CA lahko pozna privatni ključ uporabnika, praviloma pa bi moral privatni ključ poznati zgolj uporabnik sam. Poleg tega veliko CA ne zahteva osebne prisotnosti uporabnika, ampak njegovo identiteto ugotavlja na različne druge načine. Tak primer je na primer VeriSign, to je CA, ki je bil ustanovljen prvi in izda tudi največ digitalnih potrdil, predvsem za overjanje spletnih strežnikov

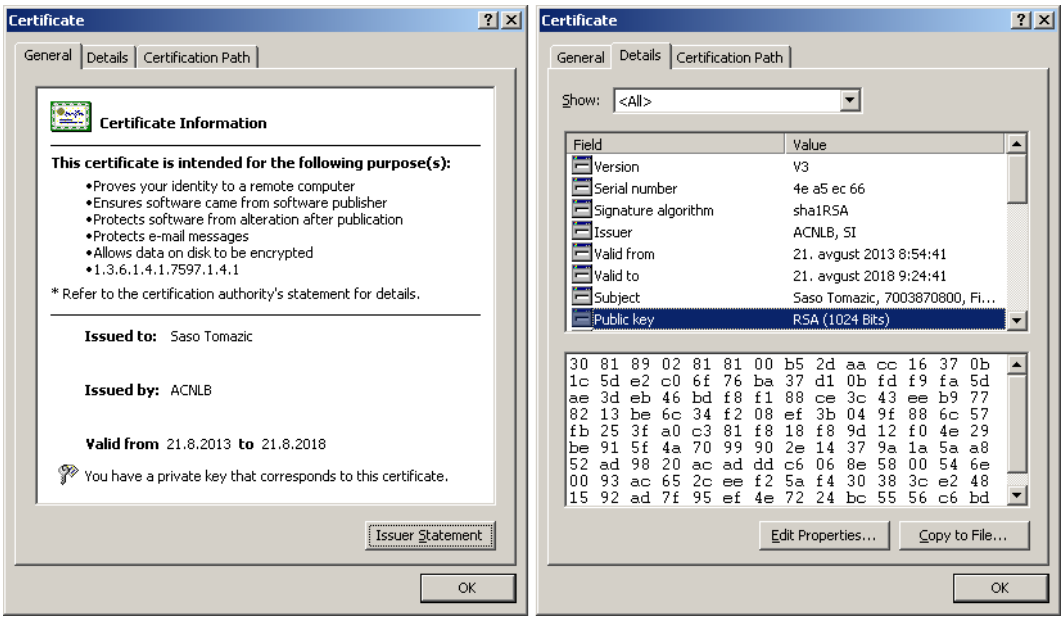
Digitalna potrdila se lahko izdajajo za različne namene, lahko služijo za identifikacijo oseb, avtentikacijo dokumentov, identifikacijo spletnih strežnikov ali podpisovanje in šifriranje elektronske pošte. Posamezno digitalno potrdilo lahko služi enemu ali več namenom.

Digitalna potrdila je mogoče tudi preklicati. Če imetnik digitalnega potrdila sumi, da mu je nekdo ukradel privatni ključ, lahko prekliče potrdilo preko spletnega strežnika. Za preklic potrdila preko spleta mora poznati svoj privatni ključ. Kdor ne pozna privatnega ključa, na ta način ne more preklicati potrdila. Tisti, ki je ukradel ključ, pa nima interesa da bi ga preklical, saj bi s tem uporabniku samo naredil uslugo.

Če uporabnik izgubi svoj privatni ključ, lahko potrdilo prekliče samo tako, da se osebno zglasi pri izdajatelju potrdila.

Digitalno potrdilo se izdaja v standardiziranih formatih. Najbolj pogosto se uporablja format po standardu X.509. Potrdila po standardu X.509 so običajno shranjena v datotekah s končnicami .pem, .cer, .crt, .der, .p7b, p7c, .p12 in .pfx. Te se razlikujejo predvsem v formatu zapisa. Sam format zapisa, je podobno kot vsebina potrdila, tudi standardiziran. Če vsebuje potrdilo tudi privatni ključ, je ta običajno šifriran s simetričnim šifrirnim postopkom.

V datoteki z digitalnim potrdilom je potrdilo zapisano v človeku neberljivi obliki. Da bi lahko videli vsebino, moramo uporabiti nek pregledovalnik. V operacijskem sistemu Windows, je vključen pregledovalnik, s katerim lahko pregledamo vsebino potrdila, hkrati ap lahko preverimo tudi njegovo veljavnost. Primer je prikazan na sliki 5.1.




Slika 5.1 – Primer prikaza digitalnega potrdila v okolju Widows

Za ogled celotnega potrdila v X.509 formatu, moramo potrdilo najprej dekodirati v tekstovno obliko, kot jo prikazuje primer 5.1.

**Primer 5.1**

Primer potrdila X.509 v tekstovni obliki.

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 1319496806 (0x4ea5ec66)
    Signature Algorithm: sha1WithRSAEncryption
        Issuer: C=SI, O=ACNLB
        Validity
            Not Before: Aug 21 07:54:41 2013 GMT
            Not After : Aug 21 08:24:41 2018 GMT
        Subject: C=SI, O=ACNLB, O=NLB, OU=Fizicne osebe/serialNumber=7003870800,
                 CN=Saso Tomazic
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (1024 bit)
                Modulus:
                    00:b5:2d:aa:cc:16:37:0b:1c:5d:e2:c0:6f:76:ba:
                    37:d1:0b:fd:f9:fa:5d:ae:3d:eb:46:bd:f8:f1:88:
                    ce:3c:43:ee:b9:77:82:13:be:6c:34:f2:08:ef:3b:
                    04:9f:88:6c:57:fb:25:3f:a0:c3:81:f8:18:f8:9d:
                    12:f0:4e:29:be:91:5f:4a:70:99:90:2e:14:37:9a:
                    1a:5a:a8:52:ad:98:20:ac:ad:dd:c6:06:8e:58:00:
                    54:6e:00:93:ac:65:2c:ee:f2:5a:f4:30:38:3c:e2:
                    48:15:92:ad:7f:95:ef:4e:72:24:bc:55:56:c6:bd:
                    22:5a:14:08:ae:55:a1:95:55
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Key Usage:
                Digital Signature, Key Encipherment, Data Encipherment
            Authority Information Access:
                CA Issuers - URI:http://www.nlb.si/?doc=5870
            X509v3 Extended Key Usage:
                TLS Web Client Authentication, Code Signing, E-mail Protection,
                IPSec User, Microsoft Encrypted File System,
                Netscape Server Gated Crypto
            Netscape Cert Type:
                SSL Client, S/MIME
            Netscape Base Url:
```

```
            https://ac.nlb.si/fizicne/prevzem/
        Netscape Revocation Url:
            clientcgi.exe?action=checkRevocation&&CRL=cn=CRL614&serial=
        Netscape Comment:
            https://ac.nlb.si/
        X509v3 Certificate Policies:
            Policy: 1.3.6.1.4.1.7597.1.4.1
              CPS: http://www.nlb.si/images/content/_doc/Politika_ACNLB-41.pdf
        X509v3 Subject Alternative Name:
            email:SASO.TOMAZIC@FE.UNI-LJ.SI
        X509v3 CRL Distribution Points:
            Full Name:
              DirName: C = SI, O = ACNLB, CN = CRL614
            Full Name:
              URI:ldap://acldap.nlb.si/o=ACNLB,c=SI?certificateRevocationList
              URI:http://acldap.nlb.si/crl/acnlbcrl.crl
        X509v3 Private Key Usage Period:
            Not Before: Aug 21 07:54:41 2013 GMT,
            Not After: Aug 21 08:24:41 2018 GMT
        X509v3 Authority Key Identifier:
            keyid:CC:BB:BB:86:D6:6F:F8:BE:B4:47:22:77:B3:B6:AD:D7:01:59:96:4D
        X509v3 Subject Key Identifier:
            4A:37:2A:06:D6:E1:07:B5:6F:9C:B4:25:95:80:FE:F7:89:07:90:AE
        X509v3 Basic Constraints:
            CA:FALSE
        1.2.840.113533.7.65.0: 0
..V8.1....
    Signature Algorithm: sha1WithRSAEncryption
         81:32:4e:7c:a3:98:8a:ee:53:84:0f:98:4b:1c:d0:23:f9:98:
         84:75:a7:ce:d7:47:7b:b0:a1:c4:ad:cb:3c:34:49:e4:e6:40:
         36:ca:42:4e:c3:bc:37:96:44:61:e7:f5:92:35:2c:cc:ce:a5:
         a5:75:76:9b:a2:98:00:3c:07:b4:c5:da:a3:ba:41:92:e2:2f:
         8c:d3:db:d1:62:92:c9:a2:58:10:94:c3:2b:81:30:b5:ae:2b:
         10:7e:3e:75:0c:03:6e:7e:c0:46:81:5f:c7:68:ba:35:10:50:
         5b:24:9f:a1:8c:08:b6:d9:42:5f:f9:01:68:9f:2a:49:44:00:
         c9:ec:1d:b9:98:b9:cf:a0:2a:c6:e4:aa:99:c5:84:87:d7:66:
         c1:83:2c:dc:e7:fe:e2:81:7e:45:e4:20:a3:ed:94:18:c9:02:
         31:03:1d:d0:8d:1b:bd:1f:bd:00:16:1f:b9:db:60:31:e9:aa:
         c8:21:cc:71:df:3d:77:94:f5:8a:63:af:45:2c:8f:8f:5c:a4:
         94:a7:52:5a:50:66:ad:cf:0c:7a:4b:e9:d0:d8:51:7c:71:3e:
```

```
dd:61:ff:a9:3c:86:92:08:ae:17:60:82:aa:ac:32:0a:86:d1:
cf:5a:19:c6:70:55:eb:de:30:02:63:fd:49:ef:be:36:9f:15:
bc:21:5a:7b
```

Kot lahko vidimo na tem primeru, vsebuje X.509 digitalno potrdilo poleg samega javnega ključa (`Public-Key`)in njegovega prstnega odtisa (`Subject Key Identifier`) še celo vrsto podatkov o imetniku in izdajatelju potrdila. Potrdilo je namenjeno za RSA 1024 bitno šifro. V razširitvah, ki so v X.509 opcijske, pa vidimo tudi, da je izdano za digitalni podpis, šifriranje podatkov, šifriranje ključev, zaščito programske kode, avtentikacijo spletnega brskalnika, šifriranje in podpisovanje elektronske pošte in vzpostavitev varne povezave preko IPSec tunela.

Potrdilo je veljavno do 8 ure 24 minut in 41 sekund dne 21. avgusta 2018 po GMT času. To pomeni, da so veljavni samo digitalni podpisi, ki so narejeni pred tem, podpisi narejeni kasneje pa nimajo več pravne veljave.

V potrdilu je naveden tudi naslov LDAP imenika, na katerem so objavljeni preklicani javni ključi:

```
ldap://acldap.nlb.si/o=ACNLB,c=SI?certificateRevocationList
```

Za podpis potrdila je bil uporabljen postopek z SHA-1 zgostitveno funkcijo in RSA šifro (`Signature Algorithm: sha1WithRSAEncryption`). Iz potrdila je tudi razvidno, da je politika izdajanja difitalnih potrdil izdajatelja dostopna na spletnem naslovu

```
http://www.nlb.si/images/content/_doc/Politika_ACNLB-41.pdf.
```

Praviloma, bi moral to politiko prebrati vsak uporabnik takega digitalnega potrdila. V politiki je poleg drugega naveden tudi prstni odtis javnega ključa izdajatelja, ki omogoča preverjanje avtentičnost njihovega potrdila. Razvidno je tudi, da je za podpisovanje uporabljen močnejši 2048 bitni RSA ključ, za katerega predvidevajo da bo dovolj močan do leta 2020.

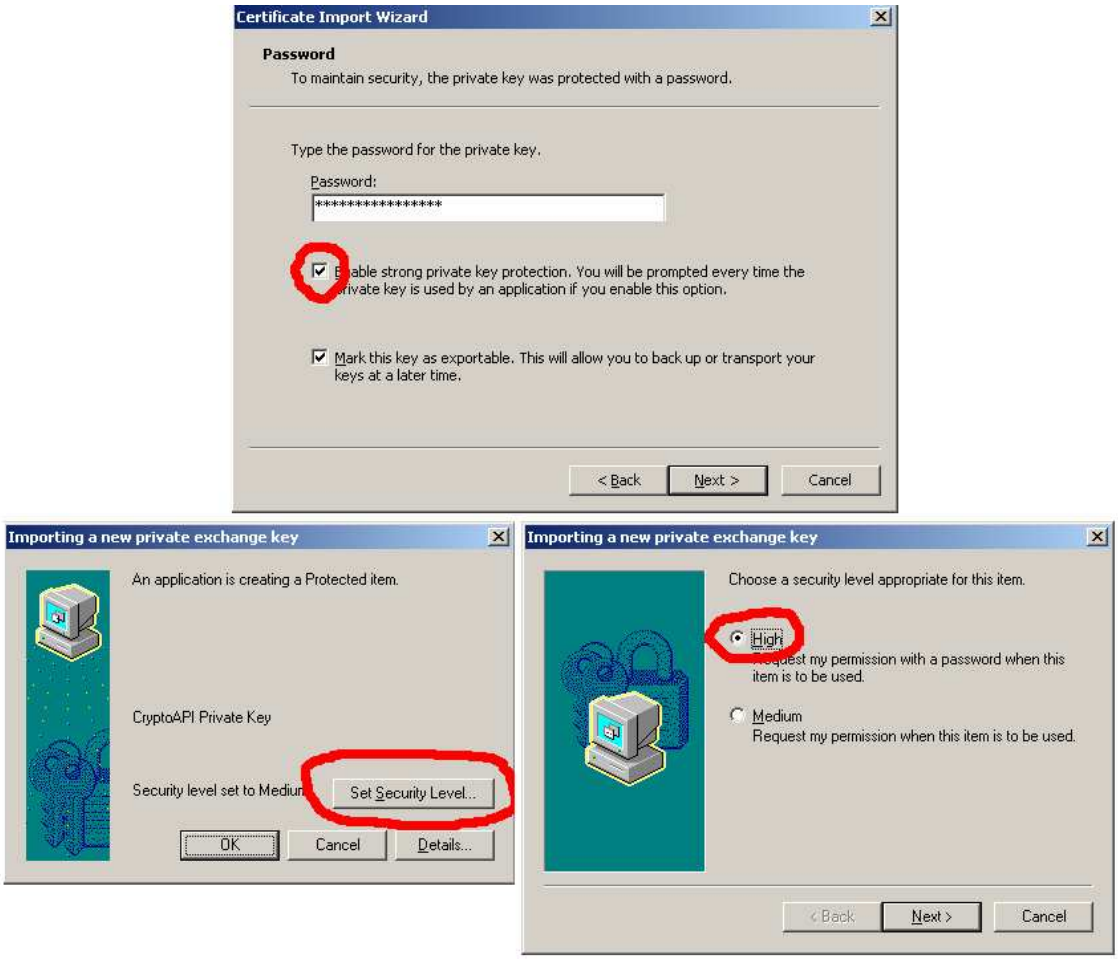
---

V datoteki s potrdili, so poleg potrdil korenskih AC shranjena tudi osebna potrdila. V osebnem potrdilu je poleg javnega ključa ponavadi tudi privatni ključ, ki ga rabimo za identifikacijo in podpisovanje. Privatni ključ mora biti nujno šifriran. V okolju Windows je potrebno biti pri uvozu potrdila zelo previden, saj je pri privzeta možnosti, ki jo ponudi čarovnik za uvoz, privatni ključ dostopen brez gesla. Pri dodajanju potrdila je zato potrebno izbrati visoko stopnjo varnosti, kot je to prikazano na sliki 5.2.

## 5.4 Časovni žig

Pri dokumentih je pomembna tudi njihova časovna opredeljenost, ki pa iz samega digitalnega podpisa ni nedvoumno razvidna. V dokument lahko nekdo vpiše starejši datum kot je dejanski datum podpisa dokumenta. To je še posebej pomembno, če je bil medtem ključ za podpisovanje preklican in dokument, ki je podpisan po preklicu, ni več veljaven.

Za časovno opredelitev dokumenta, mora, podobno kot za overjanje dokumentov, skrbeti neka notarska služba, ki izdela časovni žig dokumenta. Časovno žigosanje digitalno podpisanih




Slika 5.2 – Dodajanje digitalnega potrdila v okolju Windows. Izbrati je potrebno visoko stopnjo zaščite.

dokumentov opravljajo zaupanja vredne (TTP – Trusted Third Party) agencije za časovno žigosanje (TSA – Time Stamping Authorities).

S Časovnim žigom TSA potrjuje, da je v določenem trenutku nek dokument obstajal in, da od takrat ni bil spremenjen. Za časovno žigosanje ni potrebno, da bi TSA poznala vsebino dokumenta. Zadošča, da s svojim digitalnim podpisom podpiše prstni odtis dokumenta s točnim časom podpisa. Časovno žigosanje zato na taki agenciji opravljajo dobro zavarovani strežniki, ki so časovno sinhronizirani s časovnimi strežniki.

Podobno kot CA mora imeti tudi TSA koncesijo za opravljanje te dejavnosti, da bi imel časovni žig pravno veljavo. Za pridobitev koncesije mora izpolnjevati vse varnostne pogoje, ki se zahtevajo za tako poslovanje. Predvsem je potrebno zagotoviti, da nihče ne more dati časovnega žiga z napačnim časom. To more biti onemogočeno tudi vsem uslužbencem agencije.

Podobno kot za overjanje se tudi za časovno žigosanje uporabljajo močnejši kriptografski postopki kot za samo podpisovanje, saj mora časovni žig veljati še po preteku veljavnosti ključa za podpisovanje, s katerim je bil dokument podpisan. Časovno žigosanje na ta način podaljšuje veljavnost dokumenta, tudi, če je bil ta podpisan s šibkim digitalnim podpisom, ki je bil lahko

v vmesnem času že razbit. Ker časovni žig potrjuje, da je bil dokument podpisan že v času, ko tak podpis še ni bil razbit, s tem zagotavlja tudi, da ga ni mogel podpisati nihče, ki ni poznal privatnega ključa podpisnika.

## 5.5 Infrastruktura javnih ključev

Pojem infrastruktura javnih ključev (PKI – Public Key Infrastructure) se nanaša na organizacijo izdajanja digitalnih potrdil, njihovo hrambo, objavo in njihov preklic. PKI se lahko vzpostavlja lokalno, v okvirih nekega podjetja, na nacionalni ravni ali pa tudi na mednarodni ravni. Različne CA, ki imajo enako politiko izdajanja digitalnih potrdil, se lahko medsebojno priznavajo. V celotni PKI lahko poleg CA nastopajo tudi agencije za registracijo (RA – Registration Authorities), ki skrbijo za identifikacijo uporabnikov in agencije za preverjanje (VA - Validation Authorities), ki nudijo uporabnikom storitev preverjanja digitalnih podpisov in digitalnih potrdil. VA in RA so pogosto vključeni v CA, kot njihov sestavni del.

Posamezne CA potrjujejo druge CA, VA in RA z overjanjem njihovih digitalnih potrdi, to je njihovih javnih ključev, ki jih ti potrebujejo za overjanje digitalnih potrdil. To so tako imenovana korenska potrdila (ang. toot certificates). Običajno imajo CA, ki so vključene v PKI, drevesno strukturo. Korenska CA overja vse podrejene CA, ki nadalje zopet overjajo sebi podrejene CA. Da bi imelo to pravno veljavo, mora biti zakonsko opredeljeno, korenski CA pa mora imeti koncesijo za opravljanje te dejavnosti. Natančno mora biti tudi opredeljeno pod kakšnimi pogoji lahko korenski CA overja podrejeni CA. V Evropski skupnosti je že nekaj časa v nastajanju PKI, ki bi imela drevesno strukturo, in bi imela pravno veljavnost v celotni Evropski skupnosti.

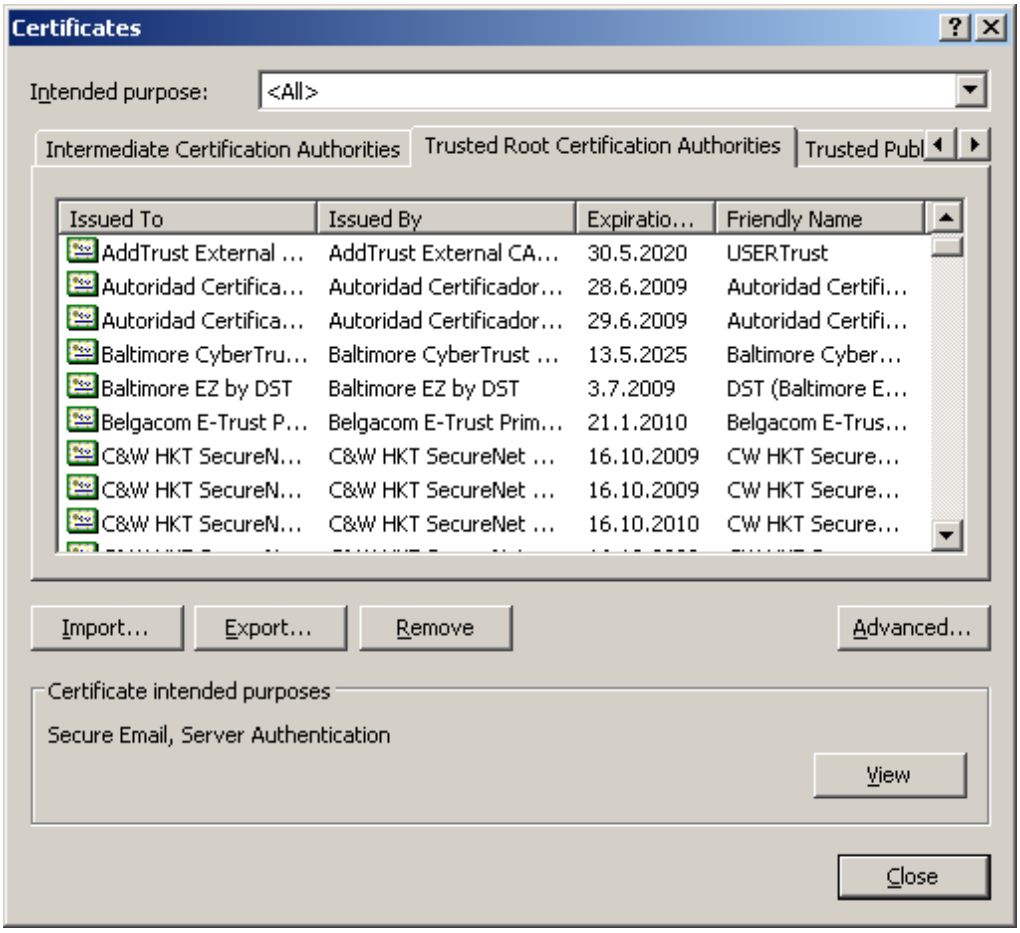
V Slovenji trenutno tudi še ni na ta način narejene drevesne strukture. Koncesijo za izdajo kvalificiranih digitalnih potrdil imajo trenutno SIGEN-CA, Halcom-CA, POŠTA®CA in AC NLB, vendar se med seboj v celoti ne priznavajo. Na primer, za dostop do e-bančništva Nove ljubljanske banke mora imeti uporabnik nujno digitalno potrdilo, ki ga izdaja AC-NLB, medtem, ko so za dostop do e-davkov uporabna digitalna vseh štirih. Za izdajanje digitalnih za poslovanje znotraj državne uprave pa skrbi posebna agencija SIGOV-CA.

Da bi lahko preverili veljavnost digitalnega podpisa ali pa avtentičnost spletnega strežnika, na katerega se prijavljamo, moramo imeti digitalno potrdilo CA, ki je to overil. To potrdilo imamo običajno nameščeno na računalniku. Potrdilom, ki so nameščena na računalniku moramo zaupati, kar pomeni, da jih moramo pridobiti na nek varen način. Preden namestimo neko novo potrdilo, moramo nujno preveriti njegovo avtentičnost.

Če je struktura drevesna, zadošča, da zaupamo samo potrdilu korenske CA, saj so vsa ostala potrdila overjena z njenim podpisom ali pa s podpisom podrejenih CA. Žal drevesna struktura PKI na globalnem nivoju ni vzpostavljena. Ker brskamo po celotnem svetovnem spletu in pogosto želimo, da je je strežnik na katerega se prijavljamo avtentičen (HTTPS), moramo imeti nameščenih na računalniku celo vrsto digitalnih potrdil.

V okolju Windows so ta potrdila shranjena v posebni datoteki, do njih pa lahko dostopamo preko orodja Internet možnosti (ang. internet options) v nadzorni plošči (ang. control panel),

ali pa neposredno iz možnosti (ang. options) spletnega brskalnika. V okolju Windows smo vezani na zaupanje podjetju Microsoft, ki ob namestitvi Windows namesti celo kopico digitalnih potrdil, ki naj bi jim zaupali, kot je to prikazano na sliki 5.3.



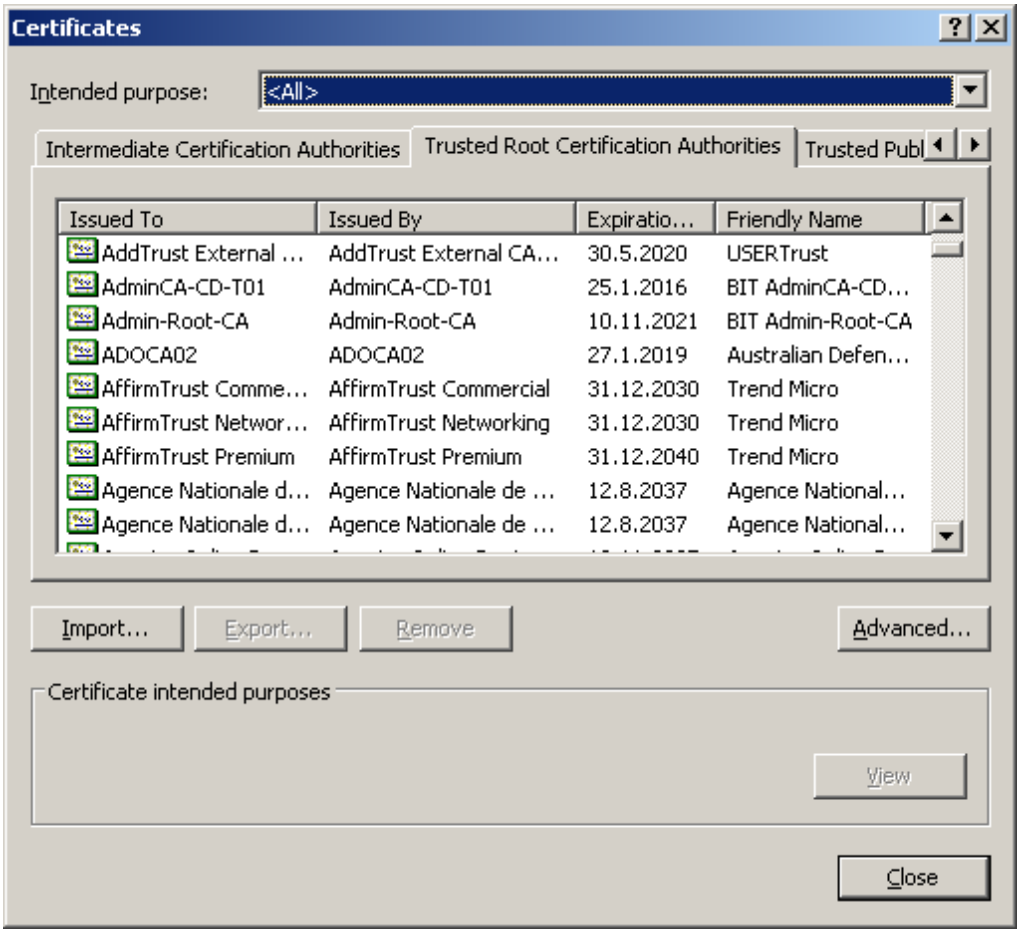

Slika 5.3 – Korenska potrdila ki jih je Microsoft vključil pri namestitvi Widows XP

Ko pregledamo vsa vključena korenska potrdila, lahko opazimo, da med njimi ni nite ene slovenske CA, zato moramo te vključiti sami, tako, da na nek zanesljiv način dobimo njihova potrdila. Na vprašanje, zakaj Microsoft vsaj v slovensko izdajo Windows ne vključi potrdil slovenskih CA, je verjetno potrebno iskati v ceni, ki jo za to zaračuna Microsoft.

Na sliki lahko opazimo lahko tudi, da je je vključenih veliko potrdil, ki jim je že potekla veljavnost, lahko pa se zgodi, da je med njimi tudi kakšno lažno digitalno potrdilo. Zato je potrebno digitalna potrdila vsake toliko časa obnoviti. Žal posodobitve spletnega brskalnika včasih ne posodobijo digitalnih potrdil, ravno tako pa jih ne posodobijo niti avtomatske varnostne posodobitve Windows. Najbolje je, če vsake toliko časa preverimo, ali so na računalniku potrdila, ki jim je potekla veljavnost. Če so, jih posodobimo. Zadnja posodobitev za windows XP je bila ob času pisanja te knjige na voljo na naslovu:

`http://www.microsoft.com/en-us/download/confirmation.aspx?id=41084`

Digitalna potrdila na istem računalniku po posodobitvi so prikazana na sliki 5.4




Slika 5.4 – Korenska potrdila Widows XP po posodobitvi

Po posodobitvi v seznamu ni več potrdil, ki jim je potekla veljavnost.

# 6 Varnost računalnikov

Računalniki niso več samostojne naprave ampak so večinoma vključene v neko omrežje. S tem so lahko izpostavljeni številnim napadom iz omrežja. To velja še posebej za računalnike, ki so neposredno vključeni v javno omrežje Internet. Poleg napadov iz omrežja so računalniki izpostavljeni tudi napadom preko neprevidnega ravnanja uporabnikov: zagon nepreverjenih programov, nameščanje nepreverjene programske opreme in neprevidna uporaba izmenljivih pomnilniških medijev (USB ključi, SSD kartice, USB diski). Ti napadi so še posebej pomembni v računalnikih, ki so vključeni v zaščitena in varna omrežja in je to alternativna pot za napade nanje.

Operacijski sistemi (OS) iz družine Windows pokrivajo 90% tržni delež, zato so računalniki z OS Windows tudi najzanimivejša tarča napadalcev. V nadaljevanju se bomo posvetili predvsem računalnikom (strežnikom in delovnim postajam) z OS Windows, kar pa ne pomeni, da računalniki z drugimi OS (UNIX, Linux, Mac OS, ...) nimajo varnostnih lukenj. Ker je njihov tržni delež manjši, so tudi napadi nanje nekoliko manj zanimivi in nekoliko redkejši. OS UNIX je imel na primer od samega začetka vgrajena stranska vrata, ki so napadalcu, ki je ta stranska vrata poznal, omogočala popolen nadzor nad računalnikom. Minilo je kar nekaj let, preden so ta stranska vrata odkrili in jih odstranili.

Napadalci zelo hitro odkrijejo varnostne luknje v novih različicah Windows, na voljo in javno dostopnih pa je tudi veliko orodij za napade nanje. Ker so danes ta orodja dostopna praktično komer koli, ni več potrebno, da je napadalec vešč programer (ang. hacker), da bi lahko izvajal različne napade.

Razvijalci operacijkega sistema Windows so v začetnem obdobju (Window 3.1, Windows NT, Windows XP,...) spregledali veliko varnostnih lukenj, ki so omogočale napadalcem relativno preprost vdor na računalnik preko omrežja. Predvsem pa je bila privzeta nastavitev sistemskih parametrov zelo ne-varna. Novejši sistemi (od Windows 7 in Windows Server 2008) imajo bistveno varnejšo privzeto nastavitev, ki pa je lahko za uporabnike včasih že moteča. Vedno namreč obstaja kompromis med varnostjo in prijaznostjo uporabe.

Napadi se lahko izvajajo preko neprimernih nastavitev, varnostnih lukenj v programski opremi OS ali varnostnih lukenj v aplikacijah, ki so nameščene na računalniku.

Ob uspešnem napadu lahko pride napadalec do zaupnih informacij ali pa na računalnik namesti neželeno programsko opremo, ki mu omogoča različne aktivnosti na samem računalniku kot tudi napade na druge računalnike v omrežju.

## 6.1 Napadi iz omrežja

Večina napadov je uspešnih predvsem zato, ker uporabniki računalnikov ne nameščajo sproti varnostnih popravkov. Varnostne luknje, ki jih izkoriščajo napadalci, so dokaj hitro, potem ko so odkrite, tudi odpravljene, vendar je potrebno sproti nameščati varnostne popravke, zato je najbolje je, da imamo vključeno avtomatsko nameščanje nadgradenj OS.

Da bi lahko napadalec uspešno izvedel napad iz omrežja, mora na nek način vzpostaviti povezavo z računalnikom. Na računalniku mora biti zato nek program, ki sprejema ukaze, ki jih dobi preko omrežja. To je lahko program, ki teče kot storitev (ang. service) v okviru OS (delitev omrežnih virov, oddaljeni dostop, ..), strežnik (podatkovni strežnik, datotečni strežnik, poštni strežnik, ...), ali pa neka aplikacija, ki se povezuje preko omrežja, kot na primer elektronska pošta, spletni brskalnik in podobno.

Računalniki z OS Windows se v omrežje povezujejo preko internet protokola (IP). Vsak računalnik ima svoj IP naslov (ang. IP address), posamezni programi pa vzpostavljajo povezavo preko svojih vrat (ang. port). Programi, ki uporabnikom nudijo določene storitve, poslušajo na določenih vratih in odgovarjajo na sporočila, ki jih prejmejo preko teh vrat. Za določene pogosto uporabljan storitve so namenjena tako imenovana dobro znana vrata (ang. well known ports). Spletni strežnik (HTTP) na primer posluša na vratih 80, varni spletni strežnik pa na vratih 443. Tu velja omeniti vrata 135, ki so namenjena oddaljenemu klicu procedur (RPC – Remote Procedure Call) ter vrata 139 in 145, ki so namenjena protokolu NetBIOS za komunikacijo v Microsoft Windows omrežju. Ta vrata so namreč najpogostejša tarča napadov na računalnike z OS Windows.

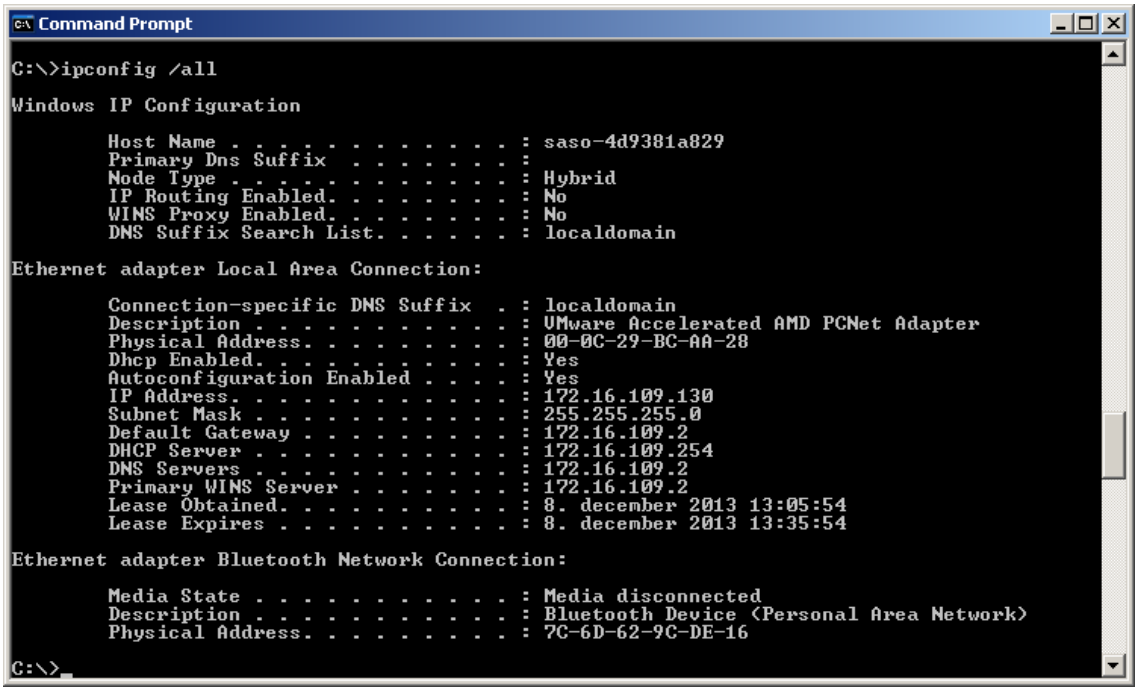
### 6.1.1 Preverjanje varnostnih nastavitev

Za OS Windows je že na voljo nekaj Microsoft orodij, ki so že vgrajena v OS ali pa so na voljo za prenos preko spleta, in omogočajo preverjanje omrežnih nastavitev računalnika in spremljanje prometa med omrežjem in računalnikom.

#### IPconfig

IPconfig je ukaz ki ga zaženemo iz ukazne vrstice in nam pokaže nastavitve vseh omrežnih vmesnikov na računalniku. Najpogosteje ga uporabljamo s stikalom /all, ki pokaže podrobnejše nastavitve, kot je to prikazano na sliki 6.1.

Na sliki vidimo, da sta na računalniku nameščena dva vmesnika: ethernet in bluetooth. Za ethernet vmesnik lahko vidimo njegov MAC (Media Access Control) naslov, IP naslov, naslov privzetega prehoda (Default Gateway), omrežno masko lokalnega omrežja, naslov DHCP (Dynamic Host Configuration Protocol) strežnika in naslov DNS (Domain Name Server) strežnika. Za bluetooth vmesnik je prikazan samo njegov MAC naslov.




Slika 6.1 – Prikaz nastavitev omrežnih vmesnikov z ukazom IPconfig /all

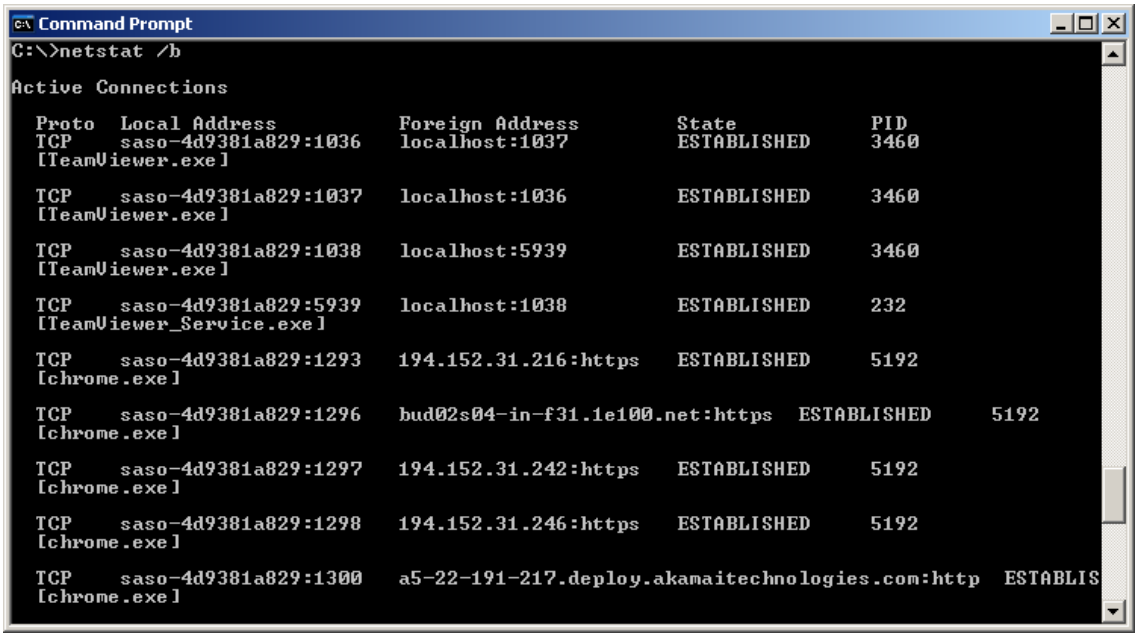
### Nbtstat

Nbtstat je diagnostično orodje za NetBIOS preko TCP/IP omrežja. Zaženemo ga preko ukazne vrstice z različnimi stikali. Omenimo predvsem stikali /n in /s. S stikalom /n izpišemo vsa lokalna NetBIOS imena, medtem ko nam stikalo /s prikaže vse NetBIOS seje, ki trenutno tečejo preko omrežnih vmesnikov računalnika. Vsa stikala, ki so na voljo in njihov pomen dobimo s stikalom /?, to tako, da z ukazne vrstice zaženemo ukaz nbstat /?.

### Netstat

Netstat je program ki ga zaženemo preko ukazne vrstice z različnimi stikali in omogoča prikaz omrežnih povezav, odprtih vrat, programov, ki poslušajo na teh vratih, usmerjevalne tabele in drugo. Vsa stikala in njihov pomen lahko dobimo z ukazom netstat /h. Tu omenimo le stikalo /a, ki prikaže vse vzpostavljene povezave in vsa odprta vrata in stikalo /b, ki prikaže vse procese, ki imajo vzpostavljeno povezavo preko omrežja, in stikalo /v, ki v kombinaciji s stikalom /b pokaže tudi lokacijo vseh programov, ki so vključeni v posamezno povezavo, kot je to prikazano na sliki 6.2.

V prikazu na sliki smo lahko opazili, da ima med poleg programov, za katere vemo, da tečejo na računalniku (chrome.exe in svchost.exe) z omrežjem vzpostavljeno povezavo tudi program jusched.exe. Ko smo preverili preko spleta, čemu je namenjen ta program, smo ugotovi, da skrbi za nameščanje nadgradenj okolja java. Program sicer ne predstavlja varnostne grožnje, vendar pa po nepotrebnem izrablja vire na računalniku, saj ni namenjen varnostnim posodobitvam, zato ga lahko, če želimo, tudi onemogočimo.



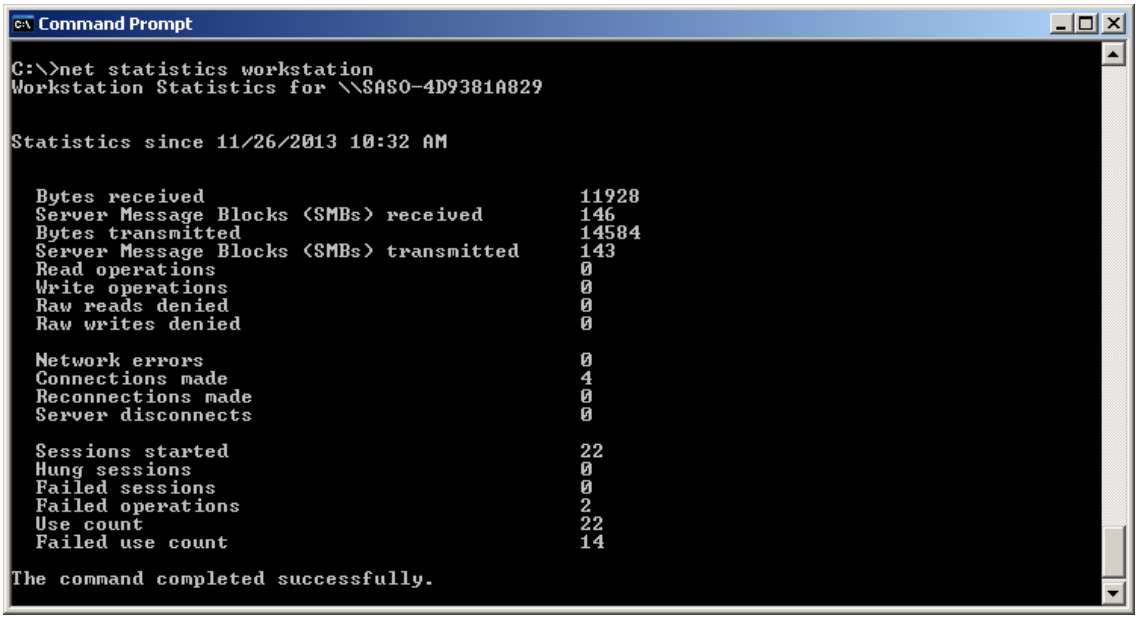

Slika 6.2 – Izpis procesov, ki se povezujejo preko omrežja z ukazom netstat /b /v

### Net

Net je družina ukazov, ki jih zaženemo iz komandne vrstice, in omogočajo vpogled in upravljanje večine omrežnih nastavitev računalnika. Z ukazom net brez dodatnih parametrov izpišemo seznam ukazov v družini:

- Net localgroup omogoča dodajanje, brisanje in upravljanje računalnikov v lokalni skupini.
- **Net pause** zaustavi oziroma da na čakanje določen vir ali storitev v okolju Windows.
- **Net session** prikaže aktivne seje z drugimi računalniki v omrežju in jih lahko tudi prekine.
- **Net share** je namenjen upravljanju virov (diskov, map, tiskalnikov, ..), ki so v skupni rabi.
- **Net start** se uporablja za zagon in pregled omrežnih storitev.
- **Net statistics** izpiše statistiko omrežnih povezav.
- **Net stop** zaustavi določeno omrežno storitev.
- **Net use** prikaže informacije o uporabi virov v skupni rabi in preslikavo diskov.
- **Net user** je namenjen upravljanju uporabniških računov na računalniku.
- **Net view** se uporablja za prikaz računalnikov v delovni skupini (workgroup).

Vse te ukaze je mogoče izvesti tudi preko orodij dostopnih v nadzorni plošči (control pannel) OS Windows. Na sliki 6.3 je prikazan primer izpisa statistike z ukazom net statistics workstation.




Slika 6.3 – Statistika prometa izpisani z ukazom net statistics workstation

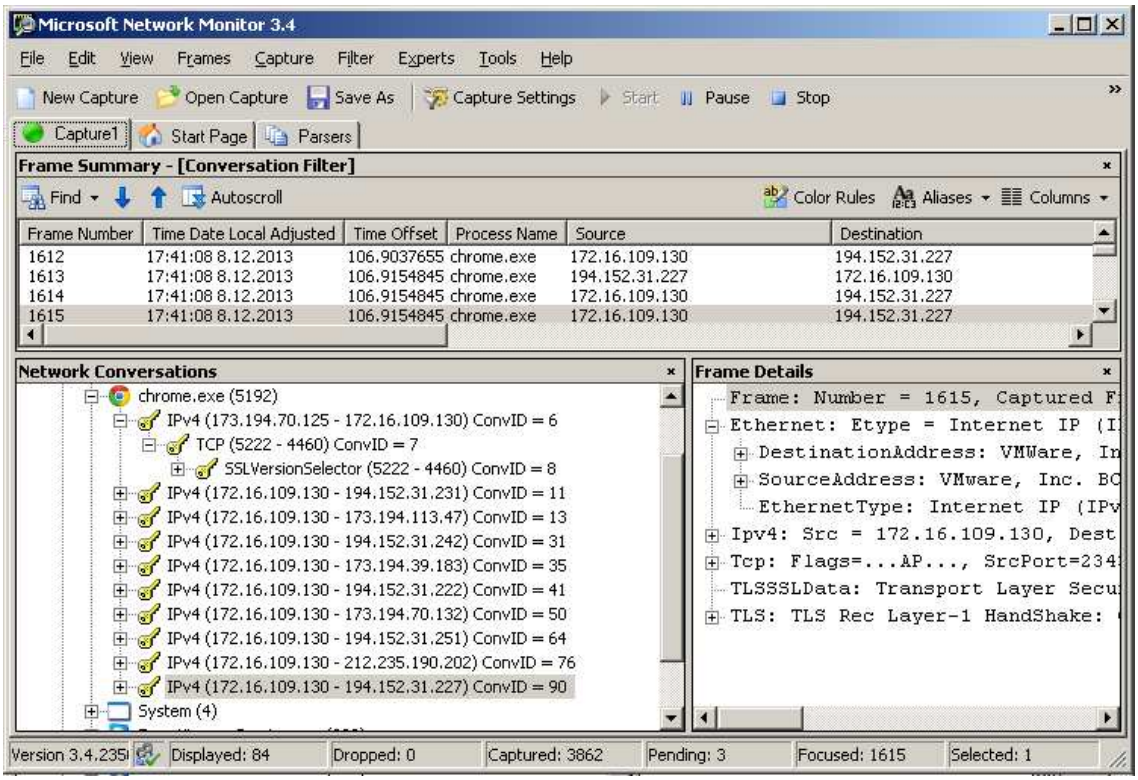
### Microsoft Baseline Security Analyzer


Slika 6.4 – Delni prikaz rezultata pregleda računalnika s programom MBSA

Microsoft Baseline Security Analyzer (MBSA) [19] je orodje, ki omogoča osnovni pregled varnostnih nastavitev OS Windows kot tudi nekatere druge Microsoft programske opreme, kot na primer Microsoft strežnikov in programov iz družine Microsoft Office. Med drugim preverja tudi, ali so na računalniku nameščene zadnje varnostne posodobitve. Za odkrite varnostne luknje ali napačne nastavitve program priporoča ustrezne ukrepe. Primer poročila, ki ga dobimo po zagonu programa je prikazan na sliki 6.4.

### Microsoft Network Monitor

Microsoft Network Monitor [20] je program, ki omogoča spremljanje vsega IP prometa na računalniku in analizo podatkovnih paketov, ki vključuje dekodiranje omrežnih protokolov in prikaz vsebine paketov, ki niso šifrirani. Omogoča filtriranje paketov glede na izvorni naslov, ponorni naslov, omrežni protokol, aplikacijo in drugo. Primer prikaza prometa s programom Microsoft Network Monitor je na sliki 6.5.




Slika 6.5 – Primer prikaza prometa na računalniku s programom Microsoft Network Monitor

Program ponuja zelo veliko različnih možnosti spremljanja prometa, vendar njihov opis presega okvire te knjige. Za natančnejša navodila naj si bralec ogleda pomoč, ki je priložena programu ali navodila, ki jih Microsoft objavlja na spletu. Za resno uporabo programa mora imeti uporabnik vsaj osnovno poznavanje internetnih protokolov.

### 6.1.2 Orodja drugih proizvajalcev

Za preverjanje nastavitev računalnika in odkrivanje in odpravljanje varnostnih lukenj OS Windows obstaja cela vrsta orodij drugih proizvajalcev programske opreme. Najbolj uporabna so orodja, ki omogočajo več različnih varnostnih testiranj združenih v eno orodje. Naštejmo tu samo nekaj boljših med njimi:

- GFI LanGuard Network Security Scanner [21]
- Shadow Security Scanner [22]

- Retina Network Security Scanner Unlimited [23]
- Nessus Vulnerability Scanner [24]
- CORE Impact Pro [25]
- Metasploit [35]
- QuasarsGuard [27]

Večina naštetih programov pomaga odkrivati varnostne luknje na tako OS kot tudi aplikacij, ki so nameščene na računalniku. Za odkrivanje varnostnih lukenj uporabljajo različne baze podatkov z znanimi varnostnimi luknjami. Ker uporabljajo različne baze in različne pristope pri pregledu računalnika, lahko nekateri med njimi odkrijejo varnostne luknje, ki jih drugi ne odkrijejo. Shadow Security Scanner omogoča tudi dostop do funkcionalnosti programa drugim aplikacijam preko vmesnika API (Application Programming Interface). Nekateri med njimi odkrivajo tudi šibka gesel, preverjajo uveljavljanje varnostne politike in sprejete zaščitne ukrepe. Večina preverja tudi, ali so na računalniku nameščene potrebne varnostne posodobitve.

Veliko priročnih orodij za OS Windows je na voljo tudi Microsoft spletni strani Windows Sysinternals [34]. Ta orodja omogočajo različne možnosti upravljanja rčunalnika, zbiranja informacij in spremljanja procesov, ki tečejo na računalniku.

### 6.1.3 Potek varnostnega pregleda računalnikov

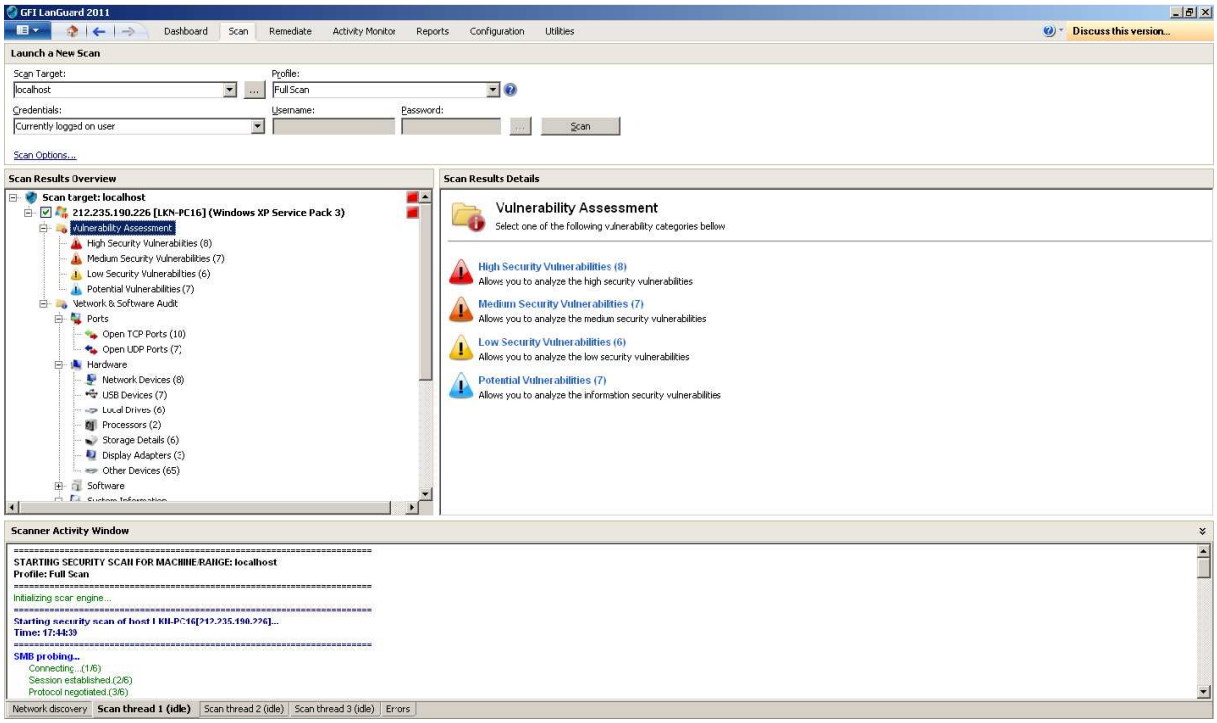
Pregled odpornosti računalnikov na napade iz omrežja začnemo s pregledom vseh računalnikov iz omrežja na enak način, kot bi to storil napadalec na omrežje. Najprej preverimo, na katerih vratih poslušajo računalniki oziroma katera vrata so odprta na posameznem računalniku. Za ta pregled lahko uporabimo eno od splošnih orodij, ki smo jih že omenili ali pa neko namensko orodje kot je to Mcaffe SuprScan [28].

V naslednjem koraku pregledamo, katere verzije OS so nameščene na posameznih računalnikih in kateri viri so dani v skupno rabo. Za to lahko na primer uporabimo program GFI LanGuard Network Security Scanner. Z istim programom lahko sedaj podrobneje pregledamo računalnike z odprtimi vrati. Program vrne podrobno poročilo o ranljivostih posameznih računalnikov. Primer poročila dobljenega s programom LanGuard Network Security Scanner je prikazan na sliki 6.6.

### 6.1.4 Zašita pred napadi iz omrežja

Kot smo že omenili se pred napadi iz omrežja dokaj dobro zaščitimo z rednimi varnostnimi nadgradnjami OS in aplikacij, ki vzpostavljajo povezavo v omrežje. Dodatno zaščito pred napadi nudi osebna požarna pregrada.

Osebna požarna pregrada je program, ki teče na računalniku, in podobno kot Network Monitor, spremlja ves promet iz in v omrežje, vendar določen promet blokira. Osebna požarna pregrada lahko blokira promet glede na IP vrata, IP naslov, aplikacijo, domensko ime, protokol




Slika 6.6 – Rezultat pregleda računalnika s programom LanGuard.

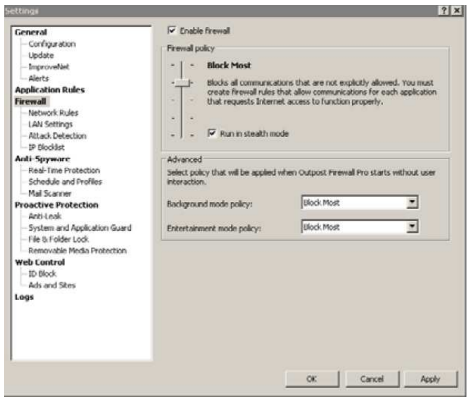
in podobno. Požarne pregrade imajo lahko permisivno politiko, dovoljeno je vse, kar ni eksplicitno prepovedano, ali restriktivno politiko, prepovedano je vse, kar ni eksplicitno dovoljeno. Restriktivna politika je bolj varna ampak tudi bistveno manj prijazna uporabniku, ki mora sam nastaviti vsa dovoljenja. V OS Windows je od verzije XP naprej že vključena požarna pregrada, ki pa ima dokaj permisiven način delovanja. Omejuje predvsem dohodne povezave, ne omejuje pa povezav, ki so bile vzpostavljene s strani računalnika.

Večina osebnih požarnih pregrad drugih proizvajalcev omejuje tako povezave iz računalnika kot tudi povezave na računalnik. Delujejo restriktivno, tako, da mora uporabnik dati dovoljenje vsaki aplikaciji, ki želi dostopati do omrežja. Omejitve so lahko različne pri dostopu do lokalnega omrežja in pri dostopu do javnega omrežja Internet.

Večinoma omogočajo ročno in dinamično nastavljanje dovoljenj in prepovedi. Pri dinamčnem načinu uporabnik preko pojavnega pogovornega okna (pop-up) za vsako aplikacijo določi pravice, ko skuša ta dostopati do omrežja. Lahko popolnoma prepove dostop, lahko omogoči dostop do lokalnega omrežja ali do javnega omrežja. Za aplikacije, ki delujejo kot strežniki (poslušajo) mora uporabnik dati posebno dovoljenje.

Nekatere osebne požarne pregrade omogočajo, da se z ustrezno nastavitvijo sami odločamo za bolj permisivno ali bolj restriktivno politiko. Primer take nastavitve v programu Outpost Pro, je prikazan na sliki 6.7.

Na spletni strani Top Ten Reviews [16], spletnem mestu, kjer je objavljenih najboljših deset ocen za različne produkte, od strojne opreme do programske opreme in storitev, so najbolje ocenjene naslednje osebne požarne pregrade:




Slika 6.7 – Nastavitev restriktivne politike na požarni pregradi Outpost Pro

- Comodo Internet Security Complete [29]
- Agnitum Outpost Pro Firewall [30]
- Kaspersky Internet Security [38]
- ZoneAlarm Pro Firewall [32]
- Total Defense Internet Security Suit [33]

Poleg filtriranja vhodnega in izhodnega prometa glede na vrata, aplikacije in naslove, te aplikacije večinoma nudijo tudi zaščito pred določenimi vrstami neželene programske opreme in proti kraji podatkov z računalnika.

Če nek računalnik ne deli svojih virov, na požarni pregradi zapremo NetBios vrata 137,138, 139 in 445. Namesto da bi blokirali ta vrata, lahko na takih računalnikih tudi izklopimo NetBios.

Na računalnikih, kjer ne teče nobena storitev, kot na primer Exchange poštni strežnik, ki bi uporabljala RPC, nujno zapremo tudi RPC vrata 135, ki so zelo pogosta tarča napadov.

## 6.2 Neželena in zlonamerna programska oprema

Neželena programska oprema na računalniku je programska oprema, ki jo je namestil nekdo drug, ali pa smo jo sami namestili nenamerno, ne da bi se tega zavedali. Taka programska oprema je lahko zgolj moteča, ker upočasnjuje delovanje računalnika, preusmerja uporabnika na določene spletne strani z reklamami ali zbira podatke o navadah uporabnika za usmerjeno oglaševanje. Pogosto pa je zlonamerna, nameščena z namenom kraje podatkov, kraje identitete in povzročanja škode.

Do okužbe računalnika z zlonamerno programsko opremo lahko pride na več načinov:

- Izkoriščanje varnostnih lukenj v spletnih brskalnikih ob dostopu do zlonamernih ali okuženih spletnih strani.
- Škodljive priloge e-poštnih sporočil.

- Prenos in nameščanje programske opreme nepreverjenega izvora (nepreverjene ali lažne spletne strani in P2P omrežja, kot BitTorrnet, eMule, SoulSeek).
- Nameščanje oziroma zagon okužene programske opreme.
- Vdor na računalnik iz omrežja zaradi slabe zaščite računalnika (šibka gesla, luknje v aplikacijah, luknje v strežniških programih).

Varnostno poročilo podjetja Check point [15] razkriva, da so bili v več kot 50% podjetij, ki so bila udeležena v raziskavi, okuženi z zlonamerno programsko opremo vsaj štirje računalniki. V istem poročilu je omenjeno, da je napadalcem v letu 2012 z napadom "Eurograbber", ki je okužil računalnike in mobilne terminale uporabnikov spletnega bančništva, uspelo ukrasti več kot 36 milijonov EUR iz okoli 30 tisoč bančnih računov.

### 6.2.1 Vohunski programi

Vohunski programi (ang. SpyWare) so programi, ki so namenjeni zbiranju podatkov o uporabniku, ne da bi se uporabnik tega zavedal.

Vohunska programska oprema je predvsem štirih tipov:piškotki (ang. cookies), oglaševalci (ang. adware), trojanski konji (ang. Troyan horses) in sistemski monitorji (ang. system monitors). Ti programi so običajno skriti uporabniku, tako, da njihovega delovanja ne opazi.

Vohunska programska oprema lahko smo spremlja obnašanje uporabnika, lahko pa z njeno pomočjo napadalec pridobi katerekoli podatke, od zaupnih informacij nekega podjetja, do gesel, podatkov o kreditnih karticah in bančnih računih in osebnih podatkov uporabnikov.

### 6.2.2 Piškotki

Piškotki so zapisi v lokalni bazi spletnega brskalnika, ki so namenjeni prilagoditvi odziva spletnih strani obiskovalcu. Pogosto pa se uporabljajo za sledenje navad uporabnika na različnih spletnih straneh, ne da bi se uporabnik tega zavedel. Pridobljene informacije se lahko potem lahko uporabijo za uporabniku prilagojene reklamne akcije. Piškotki so najmanj škodljiva oblika neželene programske opreme oziroma natančneje njenega neželenega delovanja.

### 6.2.3 Oglaševalci

Oglaševalci so programi oziroma dodatki spletnega brskalnika, ki prikazujejo razne reklamne oglase. Vsi oglaševalci niso nujno neželena programska oprema. Nekateri programi, najpogosteje na mobilnih napravah, prikazujejo oglase zgolj v brezplačni verziji, če pa program kupimo, se oglasi nehajo prikazovati.

Druge oglaševalce pa lahko, nevedoma ali neprevidno namestimo ob nameščanju brezplačne programske opreme. Ti spremljajo obnašanje uporabnika in lahko v spletnem brskalniku zamenjajo domačo spletno stran, prikazujejo pojavna okna z reklamami, ali preusmerjajo brskalnik na neželene spletne strani.

### 6.2.4 Sistemski monitorji

Sistemski monitorji so izredno nevarna programska oprema. Ko so nameščeni na računalniku, lahko na določen naslov pošiljajo različne podatke, od pritisnjenih tipk na tipkovnici, zaslonskih slik, do datotek shranjenih na računalniku. Lahko vključijo interni mikrofon ali kamero na računalniku in omogočajo prisluškovanje pogovorom in nadzor prostora, v katerem je računalnik.

Sistemski monitor lahko namesti tudi lastnik računalnika v nekem podjetju, da lahko nadzira zaposlene, ali pa lastnik javno dostopnega računalnika, da pridobi določene podatke (na primer uporabniška imena in gesla) od uporabnikov, ki ta računalnik uporabljajo.

### 6.2.5 Trojanski konji

Trojanski konji so škodljivi programi, ki jih uporabnik namesti na svoj računalnik, misleč, da je namestil nek koristen program. Ime izvira iz grške mitologije, kjer so Grki pred vrati troje pustili lesenega konja v katerem so se skrivali grški vojaki. Trojanci so mislili, da je to darilo in so ga sami zvlekli v Trojo, ne da bi vedeli, da so s tem omogočili vstop tudi grškim vojakom.

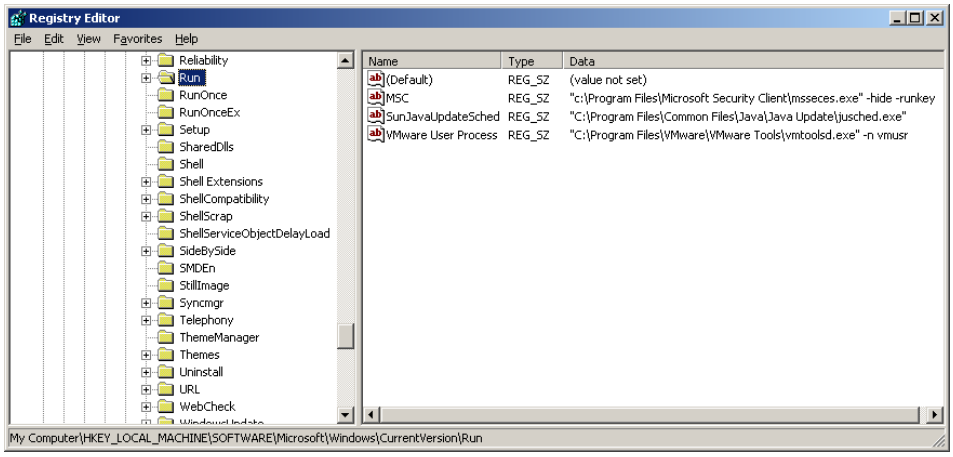
Trojanski konji so načrtovani tako, da izgledajo kot legalna programska oprema z na videz uporabno funkcionalnostjo. Trojanski konj je vsak program, ki izvaja funkcije, ki so uporabniku skrite in jih uporabnik ne želi. To je lahko samostojen program ali program, lahko je spremenjen legalen program ali pa program, ki je pripet na nek legalen program.

Trojanski konji se pogosto uporabljajo kot stranska vrata, ki omogočajo oddaljen dostop do računalnika. Trojanski konj za oddaljen dostop je sestavljen iz dveh komponent: strežnika in odjemalca. Ko uporabnik, ne da bi se tega zavedal, zažene strežnik na svojem računalniku, omogoči napadalcu, da z odjemalcem dostopa do tega strežnika in preko njega opravlja vse funkcije, ki jih ta strežnik omogoča. Napadalec mora seveda poznati IP naslov računalnika, na katerem je nameščen strežnik. Strežnik lahko javi svoj naslov preko e-pošte, preko sistema za pošiljanje sporočil ali pa kar preko povezave na nek centralni strežnik.

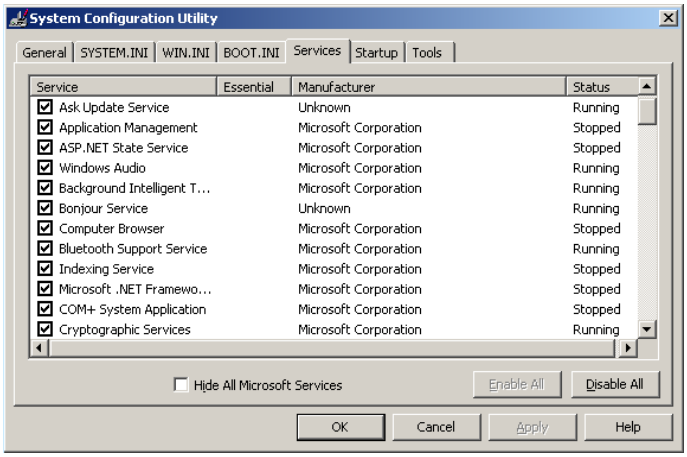
Večina trojanskih konjev se avtomatsko zažene ob zagonu računalnika, tako da spremenijo določene sistemske datoteke na računalniku. OS Windows omogoča veliko različnih načinov za avtomatski zagon programov ob zagonu. Ob zagonu se zaženejo vsi programi, ki so v mapi Startup. Zaženejo se lahko tudi z ustreznimi ukazi v datotekah Win.ini, System.ini, Wininit.ini, Autoexec.bat. Program je mogoče ob zagonu je mogoče zagnati tudi z vpisom ustreznega ključa v sistemski register. Zapis

```
[HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run]
                              "Info"="c:\program files\Explorer.exe"
```

bo ob zagonu računalnika zagnal program "Explorer.exe" v mapi "c:
program files", ki pa je lahko trojanski konj, z istim imenom kot legalni program "Explorer.exe", ki pa je v mapi "c:
Windows" in ne v mapi "c:\program files". Za zagon programa ob zagonu računalnika, se uporabijo naslednji predvsem ključi v sistemskem registru:




Slika 6.8 – Pregled ključa "Run" z urejevalnikom sistemskega registra



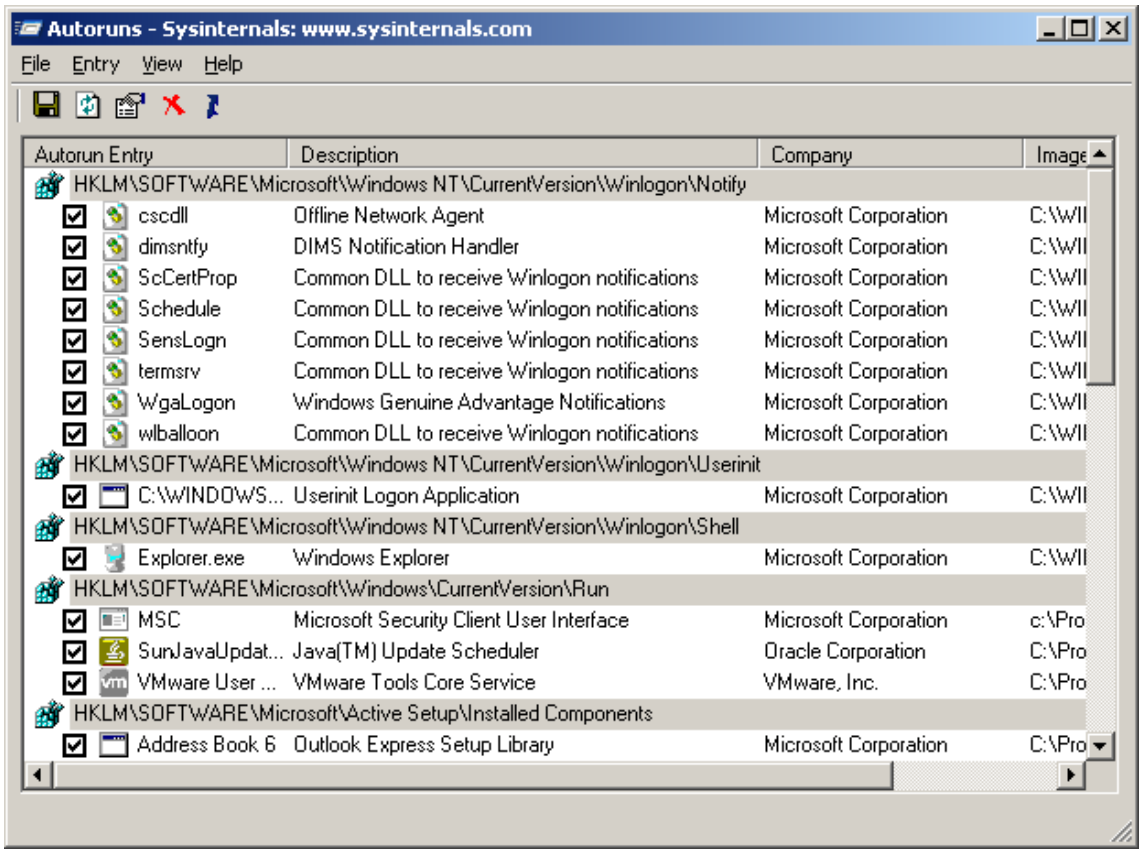

Slika 6.9 – Pregled programov, ki se zaženejo ob zagonu Windows, s programom System Configuration Utility.

- HKEY_ LOCAL_ MACHINE\Software\Microsoft\Windows\CurrentVersion\Run
- HKEY_ LOCAL_ MACHINE\Software\Microsoft\Windows\CurrentVersion\RunOnce
- HKEY_ LOCAL_ MACHINE\Software\Microsoft\Windows\CurrentVersion\RunOnceEx
- HKEY_ CURRENT_ USER\Software\Microsoft\Windows\CurrentVersion\Run
- HKEY_ CURRENT_ USER\Software\Microsoft\Windows\CurrentVersion\RunOnce
- HKEY_ CURRENT_ USER\Software\Microsoft\Windows\CurrentVersion\RunOnceEx

Te ključe je smiselno preveriti z urejevalnikom sistemskega registra (regedit.exe), ki je priložen OS Windows, da ugotovimo, ali se na začetku poganjajo neželeni programi. Primer pregleda registra s programom regedit je prikazan na sliki 6.8.

Ker obstaja, poleg omenjenih, še veliko drugih načinov za zagon programov na začetku, je za pregled teh smiselno uporabiti neko orodje, kot na primer System Configuration Utility (msconfig.exe) prikazan na sliki 6.9, ki je vključen v OS Windows.

V ta namen je priročen tudi program Autoruns (autoruns.exe), ki je eno od orodij v skupini SysInternals [34], kot je to prikazano na sliki 6.10.




Slika 6.10 – Pregled programov, ki se zaženejo ob zagonu Windows, s programom Autoruns. Te programe lahko onemogočimo tako, da odstranimo kljukice pred njimi.

Poleg o oddaljenega nadzora lahko trojanski konji opravljajo tudi druge neželene funkcije, kot so to

- kraja uporabniških imen in gesel,
- sledenje vnosa na tipkovnici in slik na zaslonu,
- brisanje datotek na disku ob določenem dnevu in uri,
- izvajanje napada onemogočanja storitev (DoS – Denial of Service) na nek strežnik,
- spreminjanje računalnika v strežnik (npr. FTP strežnik ali posredniški strežnik), dostopen vsakomur ali pa smo napadalcu, ki pozna geslo in
- onemogočanje programov za zaščito računalnika, kot so to osebna požarna pregrada, proti virusni program, programi za detekcijo neželene programske opreme in podobno.

### 6.2.6 Računalniški virusi

Računalniški virusi so še vedno najpogostejša neželena programska oprema na računalnikih. Virusi so programi, ki so se sposobni sami replicirati, tako, da se priklopijo nekemu legalnemu programu. Kadar koli uporabnik zažene tak program, zažene tudi virus, ki se lahko ponovno replicira ali pa izvaja neke druge neželene funkcije. Virusi lahko izvajajo vse funkcije, ki jih izvajajo trojanski konji, le da se trojanski konji niso sposobni sami replicirati.

### 6.2.7 Računalniški črvi

Računalniški črvi so programi, ki so se sposobni, podobno kot virusi, sami replicirati. To so lahko samostojni programi ali pa programi, ki se pripnejo na druge programe. Za razliko od virusov, ki okužijo zgolj programe na računalniku, na katerem jih je uporabnik zagnal, pa črvi izkoriščajo varnostne luknje drugih računalnikov v omrežju in se razširjajo po celotnem omrežju. Za svoje razširjanje lahko uporabljajo različne mehanizme, od prilog k elektronski pošti, ki jo pošiljajo na elektronske naslove v imeniku uporabnika, do izkoriščanja varnostnih lukenj v različnih strežnikih in aplikacijah. Večino črvov tudi spremeni sistemski register tako, da se zaženejo ob ponovnem zagonu računalnika.

### 6.2.8 Skriti programi – rootkits

Skriti programi so programi, ki se skušajo skriti pred OS, tako, da običajna orodja za odkrivanje zlonamerne programske opreme ne odkrijejo njihove prisotnosti.

Najbolj pogost način za skrivanje programov je, da dodamo novo storitev (ang. service) v OS Windows. Orodja, ki omogočajo skrivati programe pred OS so znana pod skupnim imenom "Rootkits". Beseda je zloženka dveh angleških besed "root", ki je običajno uporabniško ime administratorja na sistemu UNIX in "kit", ki pomeni skupek orodij.

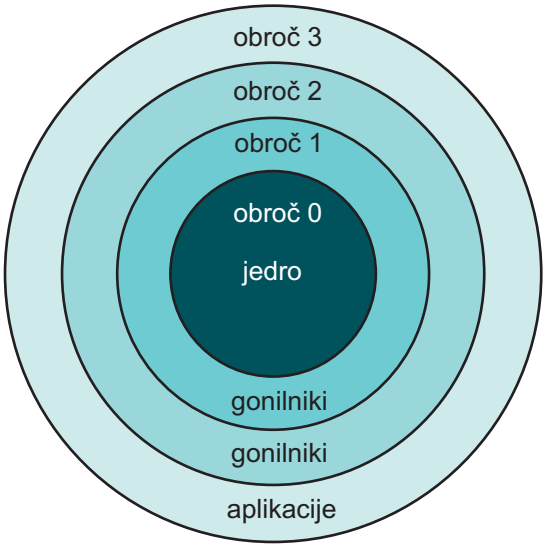
Rootkit se lahko namesti avtomatsko, s pomočjo nekega trojanskega konja ali virusa, lahko pa ga namesti napadalec sam, ko enkrat uspe dobiti nadzor nad računalnikom z neko drugo vrsto napada.

Rootkiti delujejo na zelo nizkem sloju OS in lahko onemogočijo delovanje programom, ki so namenjeni njihovemu odkrivanju. Rootkit se lahko vključi v samo jedro (ang. kernel) OS ali pa celo v sistemsko opremo (ang. firmware) samega računalnika.

Običajno rootkiti sami ne omogočajo dostopa do računalnika ali drugih škodljivih operacij. Uporabljeni so za to, da take programe skrijejo pred OS in drugimi programi za njihovo odkrivanje.

Rootkiti niso nujno zlonamerni, saj jih lahko namesti legalni uporabnik računalnika za odkrivanje napadov na računalnik. Nekateri programi, ki so namenjeni zaščiti računalnika namreč uporabljajo enake tehnike, da se sami zaščitijo pred napadi.

Rootkiti se ločijo po tem, v katerem obroču zaščite OS delujejo. Zaščita OS je namreč razdeljena na obroče, kot je to prikazano na sliki 6.11.




Slika 6.11 – Obroči zaščite OS. Programi, ki tečejo v obroču 0 (jederni način) imajo najvišje pravice, nato pa programom, ki tečejo v obročih 1, 2 in 3 (uporabniški način) pravice padajo z naraščanjem številke obroča. Najnižje pravice imajo tako uporabniške aplikacije.

Zunanji obroči lahko dostopajo notranje obroče samo preko nekaterih vnaprej določenih vrat. To lahko prepreči nekemu vohunskemu programu, ki teče kot aplikacija v obroču 3, da bi vklopil mikrofon ali spletno kamero, ne da bi bil uporabnik o tem obveščen. Upravljanje strojne opreme namreč teče v obroču 1.

Rootkiti, ki delujejo v jedernem načinu se imenujejo tudi bootkiti (angleška beseda boot označuje proces nalaganja OS). Bootkiti se naložijo hkrati ali celo pred samim OS in na ta način zaobidejo vse zaščite, ki so vključene v OS. Ko so enkrat aktivirani, jih je praktično nemogoče zaznati ali odstraniti s programi, ki tečejo v istem OS. Veliko uspešnejša zaščita je, da preprečujemo njihovo namestitev. Nekatere bootkite je možno odkriti le tako, da računalnik ugasnemo in ga zaženemo iz zaupanja vrednega pomnilniškega medija (CD ali DVD) na katerem je nameščen tudi program za odkrivanje rootkitov.

Rootkiti, ki delujejo na sloju sistemske programske opreme so lahko prisotni predvsem na napravah, kjer se ne nalaga ločeno operacijski sistem, temveč delujejo samo s sistemsko programsko opremo. To so lahko razne obrobne naprave (tiskalniki, skenerji) in omrežne naprave (stikala, usmerjevalniki, ...). Nekateri rootkiti pa se lahko skrijejo tudi v BIOS (Basic Input Output) računalnika. BIOS predstavlja sistemsko programsko opremo samega računalnika in skrbi med drugim za nalaganje OS ob zagonu računalnika.

### 6.2.9 Boti in zombiji

Boti, skrajšano od roboti, so računalniški programi, ki so sposobni opravljati določena ponavljajoča opravila na ukaz. Kadar gre za škodljivo programsko opremo, se ti programi običajno s pomočjo rootkitov skrijejo, tako, da uporabnik ne opazi njihove dejavnosti. Računalniki, ki so okuženi z boti se imenujejo zombiji. Zombiji so mitološka bitja brez lastne volje, ki so pod

popolnim nadzorom nekoga drugega.

Napadalci lahko ustvarijo celo omrežje zombijev (ang. bootnet), to je računalnikov, ki so okuženi z boti in čakajo na ukaze. Najpogosteje komunicirajo preko nenadzorovanih IRC (Internet Relay Chat) kanalov. Omrežje zombijev lahko nato napadalec izkoristi za obsežen napad akcije.

Omrežje zombijev se lahko izkoristi za pošiljanje neželene elektronske pošt (spam), napade onemogočanja storitev DoS, pridobivanje informacij o uporabnikih, skrivanje resničnih povezav ali postavljanje lažnih strežnikov.

Uporabniki lahko zombije še naprej uporabljajo, ne da bi se zavedali, da poleg želenih funkcij opravljajo še neke druge dejavnosti, kot na primer pošiljajo neželeno elektronsko pošto.

Leta 2011 je policija v sodelovanju z industrijo odkrila in onemogočila omrežje zombijev imenovano Rustock, ki je vključeval okoli milijon okuženih računalnikov in je bil sposoben poslati 30 milijard neželenih poštnih sporočil dnevno. Ko so ga onemogočili, je količina neželene pošte upadla za 30%.

Po poročilu [15] je bil v letu 2012 z boti okuženi vsaj en računalnik v 63% vseh podjetij. Obenem ocenjujejo, da je približno četrtina vseh osebnih računalnikov priključenih v internet del vsaj enega omrežja zombijev.

Trenutno je aktivnih na tisoče omrežij zombijev, glede na poročilo [15] pa so bili v letu 2013 najbolj aktivna:

- **Zeus** je bot, ki omogoča napadalcu oddaljen dostop do računalnika. Trenutno se Zeus pojavlja v velikem številu različic in verzij. Te lahko vsak ustvari z uporabo orodja Žeus toolkit", ki je tudi naprodaj. Predstavlja veliko družino trojanskih konj namenjenih za vdore v e-bančne sisteme.

- **Zwangi** je ogalševalec. Na okuženem računalniku je prijavljen kot dodatek spletnemu brskalniku. V brskalniku lahko ustvari novo orodno vrstico, zamenja privzeti iskalnik in prikazuje neželene reklame v pojavnih oknih. Običajno se namesti na računalniku hkrati z neko drugo, želeno programsko opremo.

- **Sality** je virus, ki se širi po omrežju tako, da okuži druge programe in se kopira na izmenljive pomnilniške medije in diske v skupni rabi.

- **Kuluoz** se namesti, ko odpremo zlonamerno prilogo neželene pošte. Sprjema ukaze iz oddaljenega računalnika in lahko izvrši škodljive programe na okuženem računalniku.

- **Juasek** omogoča napadalcu oddaljen dostop in popolen nadzor okuženega računalnika. Na računalnik se namestim kot storitev (service) tako, da preživi ponovni zagon računalnika.

- **Papras** omogoča oddaljeni nadzor računalnika in spremljanje celotnega internet prometa z namenom kraje predvsem bančnih informacij.

Boti predstavljajo danes bistveno večjo nevarnost kot klasični virusi, ker so vključeni v veliko omrežje, ki se je sproti sposobno prilagajati na nove obrambne mehanizme in ga je tako skoraj nemogoče popolnoma zatreti.

## 6.3 Zaščita pred zlonamerno programsko opremo

Kot pri boleznih je tudi tu boljša preventiva kot kurativa. Bolje preprečiti kot zdraviti. Večino ukrepov za preprečevanje okužbe smo že omenili, ko smo govorili o zaščiti računalnika pred napadi iz omrežja. Dodatno lahko okužbo preprečujejo tudi zaščitni ukrepi v omrežju, o katerih bomo govorili v naslednjem poglavju.

Poleg omenjenih ukrepov pa igrajo pomembno vlogo pri preučevanju in/ali odkrivanju in odstranjevanju okužbe različni programi in orodja za odkrivanje zlonamerne programske opreme in vdorov v računalnik:

- proti virusni programi,
- programi za odkrivanje in preprečevanje okužbe z boti,
- programi za zaščito pred vdori (IPS - Intrusion Prevent System),
- programi za nadzor in filtriranje obiskanih spletnih mest in
- programi za nadzor procesov, ki tečejo na računalniku.

### 6.3.1 Varnostni mehanizmi OS Windows

V OS Windows je bilo z Windows 7 uvedenih precej varnostnih izboljšav glede na prejšnje verzije tega OS. Naštejmo nekaj najpomembnejših:

- Omogočeno je šifriranje podatkov na izmenljivih medijih.
- Izboljšan je sistem nadzora nad uporabniškimi računi (EUAC – Enhanced User Account Control), ki ima štiri nivoje zaščite. Navadnim uporabniki nimajo administratorskih pravic, kar onemogoča uporabnikom kot tudi zlonamerni programski opremi izvajanje administratorskih funkcij.
- Pri namestitvi OS se namesti tudi Windows defender, ki ščiti računalnik pred vohunsko programsko opremo.
- Požarna pregrada, ki je vključena v OS, omogoča tudi filtriranje izhodnega prometa.
- Storitve, ki tečejo v okviru OS imajo minimalne pravice, ki so potrebne za njihovo delovanje, kar zmanjšuje možnosti napadalcu, ki odkrije neko varnostno luknjo v storitvi.
- Omogočen je nadzor dostopa do omrežnih virov preko NAP (Network Access Protection).

- Dodana je funkcija AppLocker, ki omogoča omejevanje dostopa do aplikacij različnim uporabnikom.
- Omogočena je identifikacija na osnovi biometričnega prstnega odtisa.

Kljub vsem varnostnim izboljšavam pa zaščita, ki jo nudi OS Windows, ne zagotavlja popolne varnosti. Pomembna pomankljivost privzetih nastavitev je, da so skrite končnice izvedljivih programov. Če uporabnik prejme po elektronski pošti neko prilogo, kot na primer 'Ponudba.txt.exe' ne vidi končnice ".exe"in sklepa, da gre za neškodljivo tekstovno datoteko. To nastavitev je zato priporočljivo spremeniti takoj po namestitvi OS.

Poleg tega je znanih še veliko število različnih varnostnih lukenj OS Windows. Seznami vseh znanih ranljivosti različnih OS se sproti dopolnjujejo na spletnem portalu CVE Details [18].

### 6.3.2 Orodja drugih proizvajalcev

Na voljo je zelo veliko različnih orodij in programov za zaščito pred nezaželeno programsko opremo, kot so to programi za odkrivanje varnostnih lukenj, programi za spremljanje in nadzor procesov, protivirusni programi, programi za odkrivanje botov, programi za odkrivanje oglaševalcev in drugi.

Za odkrivanje varnostnih lukenj v OS Windows je na voljo orodje Metasploit [35], ki uspešno odkrije vse znane varnostne luknje OS Windows in predlaga ustrezne nastavitve.

Zelo dobro zaščito pred neželeno programsko opremo nudi program Process Guard, ki pa ga je proizvajalec Dymond Computer System žal prenehal nadgrajevati, vendar je zadnja verzija še vedno na voljo na raznih spletnih portalih za prenos programske opreme, kot na primer na portalu Softpedia [36].

Process Guard deluje nekoliko drugače od drugih programov za zaščito računalnika. Deluje na zelo nizkem nivoju OS in nadzira zagon vsakega procesa. Nobenemu procesu ne dovoli izvajanja brez dovoljenja, kar preprečuje zlonamerni programski opremi, da bi se izvedla.

Ker v okviru OS teče zelo veliko število procesov, lahko deluje Process Guard v načinu učenja, ko sam dodeli dovoljenje vsem procesom, ki se izvajajo. V proces učenja ga lahko preklopimo le, kadar smo popolnoma prepričani, da računalnik ni okužen z zlonamerno programsko opremo, najbolje takoj po sveži namestitvi OS oziroma sveži namestitvi nove, zanesljive programske opreme. Preden ga preklopimo v način učenja, je najbolje, da izključimo računalnik iz omrežja. Proces učenja pa je lahko tudi velika varnostna luknja tega programa, saj se kaj lahko zgodi, da ga damo v proces učenja, ko nameščamo neko programsko opremo, ki je nismo prej dovolj dobro preverili in je lahko sama zlonamerna ali pa ima dodane druge nezaželene programe.

Za zaščito računalnika je namesto posameznih orodij smiselno izbrati orodja oziroma programe, ki nudijo celovito zaščito proti različnim oblikam neželene programske opreme. Naštejmo tukaj le pet v letu 2013 najbolje ocenjenih na spletnem portalu Top Ten Reviews [16]:

- Bitdefender Total Security [37]
- Kaspersky Antivirus [38]

- F-Secure Anti-Virus [39]
- Norton Antivirus Plus [40]
- Avast Pro Antivirus [41]

Vsi našteti programi sprotno zaščito pred zagonom okuženega programa, periodično pregledovanje celotnega računalnika, pregledovanje na zahtevo, varen zagon računalnika iz CD-ja, shranjevanje sumljivih programov v karanteno in belo listo programov , ki jih označimo za neškodljive. Vsi nudijo zaščito proti vsem različnim oblikam škodljive programske opreme, od trojanskih konjev in virusov do rootkitov in botov. Omogočajo tudi sprotno preverjanje elektronske pošte in datotek na izmenljivih pomnilniških medijih.

Bitdefender omogoča poleg tega tudi iskanje zastarelih verzij programov, ki vsebujejo varnostne luknje.

### 6.3.3 Izolirano okolje

Da bi preprečili okužbe računalnika z neželeno in zlonamerno programsko opremo, lahko določene aplikacije v izoliranem in zaščitenem okolju. Zaščiteno okolje ima omejen dostop tako do virov računalnika kot tudi do funkcionalnosti OS, ki teče na računalniku.

Nekatere aplikacije, kot na primer spletni brskalniki, sami po sebi tečejo v izoliranem okolju in naj ne bi imeli dostopa do funkcij OS, ki bi bile lahko nevarne. Poleg tega naj bi imeli izredno omejen dostop do diska računalnika. Zaradi boljše uporabniške izkušnje imajo sodobni brskalniki izredno veliko funkcionalnost. Skriptni jezik javascript, ki je namenjen programiranju uporabniškega vmesnika, ki ga vidi uporabnik na določeni spletni strani, je izredno bogat. Poleg tega omogočajo spletni brskalniki tudi nameščanje tako imenovanih razširitev (ang. extension) in vtičev (ang. plugins). Vse to pa ima za posledico, da so v izvedbi spletnih brskalnikov določene varnostne luknje, ki napadalcu, ki jih pozna, omogočijo, da izvaja na računalniku funkcije tudi izven izoliranega okolja.

Večina aplikacij ne deluje v izoliranem okolju Tem aplikacijam so dostopni vsi viri, ki so dostopni uporabniku, ki jih zaganja. To daje zlonamernim programom, kot so to trojanski konji, možnost dostopa do vseh virov računalnika.

Aplikacije, ki jim ne zaupamo, ker smo jih dobili iz nezanesljivega izvora ali pa so od proizvajalca, ki mu popolnoma ne zaupamo ali pa ga ne poznamo, je smiselno izvajati v posebej ustvarjenem izoliranem okolju. V takem okolju je smiselno zaganjati tudi spletni brskalnik, kadar obiskujemo nepoznane strani na spletu, ki bi lahko izkoristile varnostne luknje v brskalniku.

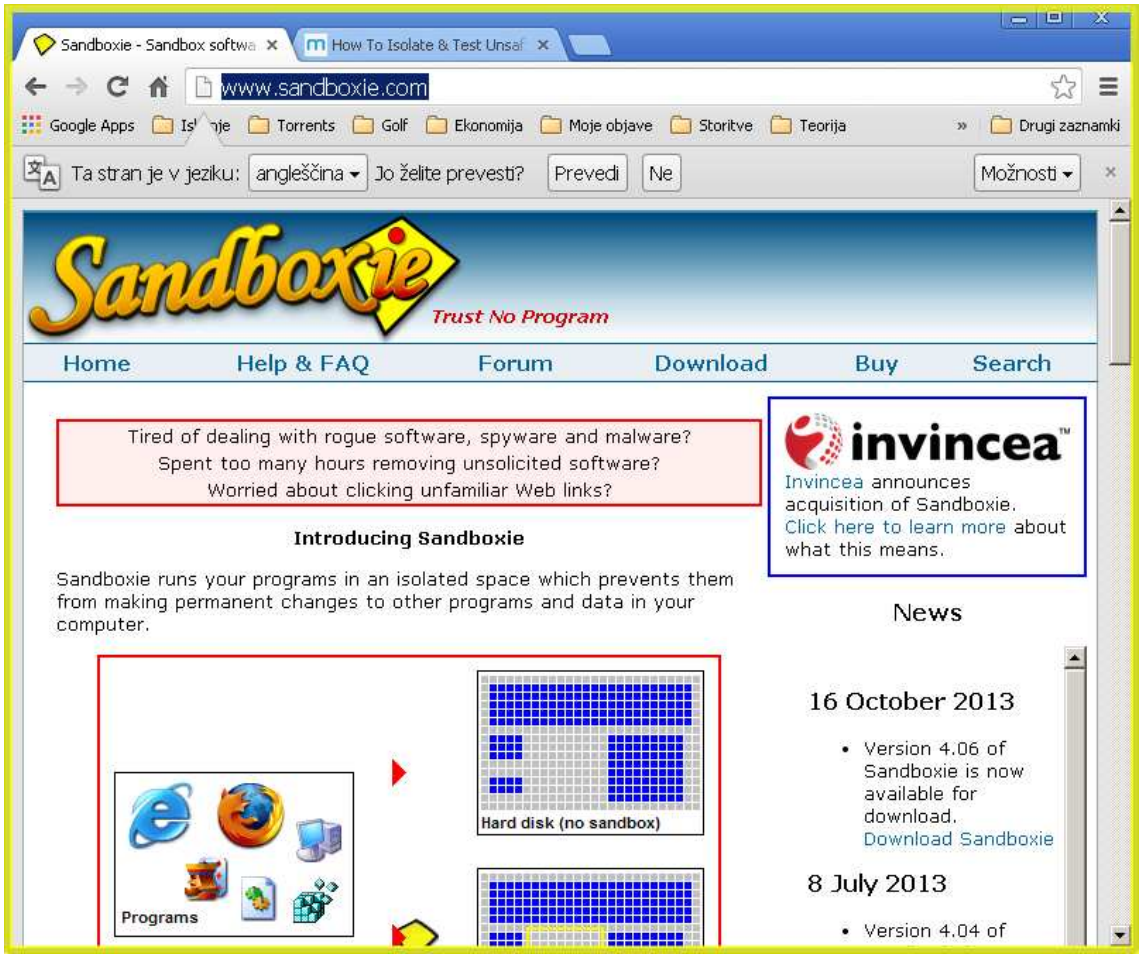
Poznamo dve osnovni različici zaščitenih okolij: peskovnik (ang. sandbox) in navidezen računalnik (ang. virtual machine).

#### Peskovnik

Peskovnik je navidezno okolje, v katerem teče določen uporabniški program. Navidezno okolje je preslikava realnega okolja. Program ima pri tem dostop samo do navideznih virov, ki pripadajo

temu okolju, tako, da ne more narediti nobenih trajnih sprememb na računalniku. Po uporabi programa, ki ga zaženemo v peskovniku, lahko razveljavimo vse spremembe, ki jih je program naredil v peskovniku, tako da ostane tudi peskovnik čist.

Na voljo je več programov, ki omogočajo zagon uporabniških programov v izoliranem okolju. Med najbolj popularnimi je Sandboxie [42], ki ustvari vmesno plast med računalnikom in aplikacijo. Vse spremembe, ki jih naredi program se shranjujejo na izoliranem področju diska. Sandboxie omogoča zaganjanje spletnega brskalnika in odjemalca pošte v peskovniku z enim klikom miške. Edina razlika, ki jo lahko vidimo, če zaženemo aplikacijo v peskovniku je okvir, ki ga Sandboxie naredi okrog okna aplikacije, ko zapeljemo kazalec miške čez glavo okna, kot je to prikazano na sliki 6.12.




Slika 6.12 – Spletni brskalnik, ki je zagnan v peskovniku s programom Sandboxie, deluje enako kot spletni brskalnik zagnan v realnem okolju. Ločimo ga lahko po rumenem okvirju, ki ga Sandboxie naredi okrog okna aplikacije.

Drug način zagona aplikacij v peskovniku pa so tako imenovane prenosne aplikacije (ang. Portable applications). Prenosna aplikacija ima v sebi vključeno svoje navidezno okolje, v katerem teče in ne posega izven tega okolja. Vse spremembe, ki jih naredi aplikacija, se izvedejo izključno v tem okolju. Prenosno aplikacijo imamo tako lahko shranjeno na prenosnem disku

ali USB ključu in jo zaženemo na katerem koli računalniku z ustreznim OS. Ko aplikacijo zapustimo, ta na računalniku ne pusti nobenih sledi. Aplikacijo naredimo prenosno tako, da jo namestimo s pomočjo nekega programa za izdelavo prenosnih aplikacij. Med bolj znanimi programi za izdelavo prenosnih aplikacij ThinApp [43] proizvajalca Vmware, ki je sicer bolj znan po navideznih računalnikih, seveda pa obstaja še cela vrsta drugih programov za izdelavo prenosnih aplikacij.

**Navidezen računalnik**

Za razliko od peskovnika, kjer teče v navideznem okolju posamezna aplikacija, je navidezen računalnik navidezno okolje, v katerem teče celoten OS. Navidezen računalnik emulira delovanje celotnega računalnika. OS, ki teče na navideznem računalniku ni nujno enak kot OS gostitelja, to je računalnika na katerem teče program navideznega računalnika. OS navideznega računalnika je popolnoma izoliran od OS gostitelja.

Navidezni računalniki se uporabljajo za različne namene. Lahko jih uporabljamo za boljše izkoriščanje strojne opreme, tako da različne strežnike namestimo na navidezne računalnika, ki pa so vsi na istem fizičnem računalniku. Lahko jih uporabimo za testiranje pri razvoju programske opreme, ko preverjamo njeno delovanje na različnih OS.

Navidezen računalnik pa lahko uporabimo tudi za namen zaščite osnovnega računalnika pred zlonamerno programsko opremo. Vsa potencialno nevarna opravila, kot sta to brskanje po spletu in preizkušanje novih, potencialno nevarnih aplikacij, opravljamo na posebej za ta namen nameščenemu navideznemu računalniku. Navidezen računalnik lahko po tem vedno vrnemo na začetno stanje, tako, da izničimo vse spremembe, ki so pri teh opravilih nastale.

Če uporabljamo navidezen računalnik zato, da bi zaščitili svoj računalnik pred zlonamerno programsko opremo, moramo paziti, da ne uporabljamo istega navideznega računalnika tudi za dejavnosti, ki jih želimo zaščititi. Torej do svojega bančnega računa ne dostopamo na navideznem računalniku, ki ga uporabljamo za brskanje po spletu in preizkušanje nove programske opreme. Lahko pa se odločimo, da opravilom, ki zahtevajo posebno varnost, kot je to dostop do bančnega računa, namenimo ločen navidezen računalnik, ki ga uporabljamo izključno v ta namen.

Kot smo že omenili, je med najbolj znanimi proizvajalci navideznih računalnikov podjetje Vmware, ki nudi celo vrsto rešitev za virtualizacijo, to je ustvarjanje navideznih okolij.

# 7 Varnost omrežja

Večina računalnikov danes ni priključena neposredno v javno omrežje Internet temveč je povezana v neko zasebno omrežje. Zasebno omrežje je lahko lokalno ali pa je geografsko distribuirano na več lokacijah. Zasebno omrežje je lahko v eni ali več točkah povezano v javno omrežje.

V zasebnem omrežju lahko med računalniki poteka komunikacija, ki v javnem omrežju ni potrebna. Računalniki vključeni v zasebno omrežje si lahko med seboj delijo skupne vire, kot so to diski, tiskalniki, različni strežniki in podobno. Računalniki zato ne morejo biti tako zaščiteni pred vdori iz zasebnega omrežja, kot bi želeli, da so zaščiteni pred vdori iz javnega omrežja. Da bi lahko zaščitili računalnike v zasebnem omrežju , moramo zaščititi celotno zasebno omrežje.

Za zaščito zasebnega omrežja je najbolj priporočljiva tako imenovana obramba v globino (ang. defense in depth), ki je osnovana na večslojni zaščiti. Osnovna ideja take zaščite je, da notranji sloji še vedno ščitijo omrežje, če napadalec predre zunanji sloj. To lahko napad onemogoči ali pa vsaj upočasni, tako da ga lahko zaznamo, preden napadalec predre vse sloje zaščite.

Notranje sloje zaščite predstavljajo ukrepi za zaščito računalnika, ki smo jih obravnavali v prejšnjem poglavju in vključujejo osebno požarno pregrado, programe za zaščito pred neželeno programsko opremo, šifriranje pomembnih datotek, predvsem datotek sključi, ter zaganjanje nepreverjenih aplikacij in brskanje po spletu v izoliranem okolju.

## 7.1 Požarna pregrada


Slika 7.1 – Požarna pregrada

Požarna pregrada je naprava, ki nadzoruje promet med dvemi omrežji. To je lahko samostojna naprava, lahko je del usmerjevalnika ali pa teče kot programska oprema na računalniku. Požarna pregrada predstavlja oviro med zaupanja vrednim zasebnim in ne-varnim omrežjem, na

primer javnim omrežjem Internet.

Požarne pregrade lahko delujejo na različnih slojih protokolnega sklada: na omrežnem sloju (IP), na transportnem sloju (TCP, UDP, ...) ali na aplikacijskem sloju (SMTP, HTTP, ...).

Prva generacija požarnih pregrad deluje zgolj na osnovi pregledovanja glave IP paketov. Izloča oziroma prepušča pakete na osnovi izvornega IP naslova, ponornega IP naslova in vrat. Ker poteka večina prometa po omrežju po protokolih TCP in UDP, ki uporabljajo tako imenovana dobro znana vrata, lahko na ta način omejimo določene vrste prometa. Ker so te požarne pregrade delovale predvsem na omrežnem sloju in samo delno na transportnem sloju (številka vrat), so obravnavale vsak paket ločeno od ostalih paketov, ne glede na vzpostavljeno povezavo na transportnem sloju. Taka požarna pregrada lahko prepuščala tudi pakete, ki ne uporabljajo dobro znanih vrat za neko storitev, čeprav te storitve naj ne bi prepuščala.

Druga generacija požarnih pregrad deluje tudi na transportnem sloju, tako da spremlja stanje povezave. Določene pakete prepušča le, če je bila predhodno vzpostavljena povezava med izvornim in ponornim naslovom.

Tretja generacija požarnih pregrad deluje na aplikacijskem sloju. V paketih pregleduje vsebino paketa, ki mora ustrezati določenemu dovoljenemu aplikacijskemu protokolu (SMTP, FTP, HTTP). Pakete, ki nimajo ustrezne vsebine preprosto zavrže, ne da bi to javila pošiljatelju paketa. S tako požarno pregrado se lahko prepreči uporaba določenih potencialno nevarnih aplikacijskih protokolov za povezavo v javno omrežje. Prepreči se lahko tudi komunikacijo različnim nestandardnim protokolom, ki bi jih lahko uporabila neželena ali zlonamerna programska oprema za posredovanje različnih informacij napadalcu. Nastavitev parametrov take požarne pregrade pa vedno predstavlja nek kompromis med udobnostjo in varnostjo.

## 7.2 Prevajanje omrežnih naslovov

V okviru zasebnega omrežja se običajno uporabljajo IP naslovi, ki so namenjeni zgolj za uporabo v zasebnih omrežjih. Ti naslovi niso neposredno dostopni iz javnega omrežja, saj isti naslovi nastopajo v velikem številu zasebnih omrežij.

Pri povezavi računalnika v zasebnem omrežju v javno omrežje se zato izvaja prevajanje omrežnih naslovov (NAT – Network Address Translation). NAT je lahko izveden v usmerjevalniku, lahko je del požarne pregrade ali pa programska oprema na računalniku. Zasebna IP naslov in vrata se prevedeta v javni IP naslov in vrata. Tako prevedbo inicializira računalnik znotraj zasebnega omrežja, ki zahteva povezavo z zunanjim omrežjem. Za zunanje omrežje so zasebni naslovi skriti in zato so tudi naprave znotraj zasebnega omrežja iz zunanjega omrežja nedostopne.

Če želimo, da je nek strežnik v zasebnem omrežju dostopen iz zunanjega omrežja, je potrebno na NAT nastaviti stalno preslikavo zasebnega naslova tega strežnika v javni naslov. Čeprav je tak strežnik dostopen iz zunanjega omrežja pa ostaja njegov zasebni naslov še vedno skrit. Čeprav je bil NAT na začetku uveden predvsem za reševanje problema pomanjkanja IPv4 naslovov, pa danes predstavlja pomemben člen v zaščiti zasebnih omrežij, saj poleg požarne pregrade

preprečuje pregledovanje in odkrivanje varnostnih lukenj na računalnikih v zasebnem omrežju.

## 7.3 Posrednik

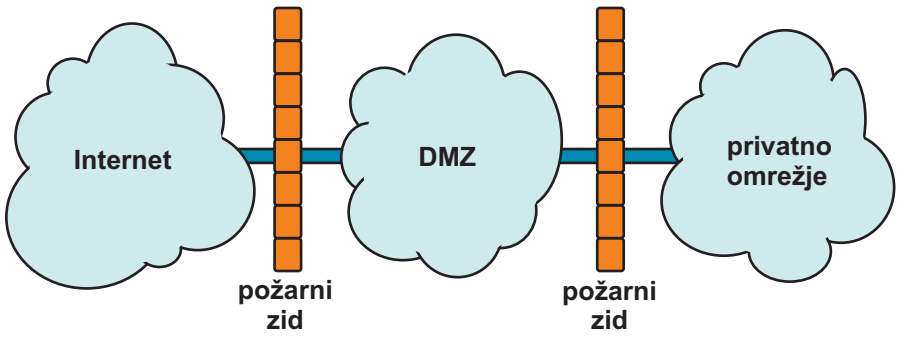
Posrednik (ang. proxy) je strežnik, ki v imenu računalnikov v zasebnem omrežju, posreduje vse v javno omrežje in tudi v njihovem imenu sprejema odgovore. Računalnik se iz zasebnega omrežja se na ta način nikoli ne povezuje neposredno v javno omrežje.

Posredniki lahko služijo različnim namenom. To so predvsem:

- Varnost. Računalniki za posrednikom niso neposredno dostopni iz javnega omrežja. Ves promet med javnim in zasebnim omrežjem se odvija preko posrednika, na katerem se lahko opravi varnosten pregled vsega prometa. Pregleduje lahko tudi vsebino vhodnega in izhodnega prometa in omejuje prenos vsebin, ki niso v skladu z varnostno ali drugo politiko podjetja.
- Uravnoteženje prometa (ang load balancing). Posrednik lahko zahteve, ki so namenjene nekemu strežniku porazdeli med več strežnikov.
- Beleženje. Posrednik lahko beleži zgodovino prometa med javnim in zasebnim omrežjem.
- Pospeševanje dostopa in zmanjšanje prometa po hrbteničnem omrežju. Posrednik lahko deluje kot vmesni pomnilnik (ang. cache) in shranjuje pogosto dostopane statične spletne strani, tako da jih ni potrebno vsakič znova prenašati z nekega oddaljenega strežnika.

Posredniki se, poleg legalnih, uporabljajo tudi za nelegalne namene, kot je to prestrezanje in pregledovanje prometa, vključno z gesli, skrivanje identitete pri dostopu do določenih in izogibanje omejitvam, ki jih določa varnostna politika.

## 7.4 Demitalizirana cona




Slika 7.2 – Demitalizirana cona je s požarnima zidovoma ločena od privatnega in javnega omrežja.

Demitalizirana cona (DMZ – DeMilitarized Zone) predstavlja dodaten sloj pri obrambi v globino. DMZ je ločeno (logično ali fizično) omrežje, ki ga postavimo med zasebno in javno

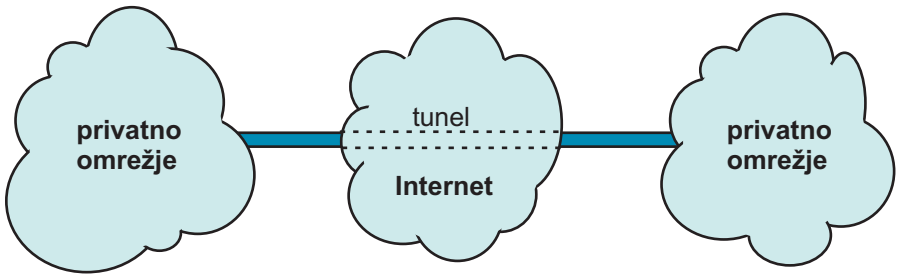
omrežje. V tem omrežju so vsi strežniki, ki so neposredno dostopni iz javnega omrežja, kot so to spletni strežnik, poštni strežnik in podobno. Ti strežniki imajo zelo omejen dostop do virov v zasebnem omrežju. Dostop je omejen samo na tiste vire, ki jih nujno potrebujejo za svoje delovanje. Hkrati pa je omejena tudi komunikacija med strežniki v DMZ in javnim omrežjem, kot je to prikazano na sliki 7.2.

DMZ nudi dodatno zaščito pred napadi iz javnega omrežja. Če se napadalcu uspe prebiti do strežnikov v DMZ, še vedno nima dostopa do strežnikov z občutljivimi podatki, ki so v privatnem omrežju. Strežniki v DMZ lahko omogočajo tudi zaznavanje napadov, in tako skrbnika omrežja opozorijo, da sprejme ustrezne ukrepe, preden uspe napadalcu prebiti naslednji požarni zid.

## 7.5 Navidezno zasebno omrežje

Zasebno omrežje lahko sestavlja eno ali več med seboj povezanih lokalnih omrežij (LAN – Local Area Network). Lokalna omrežja so lahko povezana med seboj preko zasebnih fizičnih prenosnih poti ali pa preko javnega omrežja. Kadar so povezana preko javnega omrežja, govorimo o navideznim zasebnem omrežju (VPN – Virtual Private Network).

Povezava med dvemi lokalnimi omrežji preko javnega omrežja poteka tako, da se med njima ustvari tako imenovan tunel, kot je to prikazano na sliki 7.3. Tunel predstavlja šifrirano




Slika 7.3 – Navidezno privatno omrežje. Dve lokalni omrežji sta povezani preko tunela skozi javnega omrežje Internet. Podatki v tunelu so šifrirani.

povezavao med robnima usmerjevalnikoma obeh omrežji, tako, da so podatki pri prenosu zaščiteni.

Navidezna zasebna omrežja se ločujejo po tem, kje se zaključi tunel. Tunel se lahko zaključi na robu ponudnika internetnih storitev (ISP - Internet Service Provider) ali pa na robu lokalnega omrežja. V prvem primeru mora ISP ponujati storitev VPN, v drugem primeru pa lahko vzpostavimo VPN sami.

VPN, ki jih ponuja ISP, imajo določene prednosti pred VPN, ki jih ustvarimo sami. Njihova izvedba je lahko učinkovitejša, poleg tega pa so lahko napadalcu poleg podatkov skrita tudi izvorni in ponorni naslov tunela. Paketi se lahko namreč v tem primeru šifrirajo vključno z glavo paketa, ki vsebuje izvorni in ponorni naslov. Da pa bi lahko usmerjevalniki usmerjali te pakete, se morajo na vsakem usmerjevalniku tudi dešifrirati, kar pomeni, da mora ISP poznati šifrirni ključ. Zato je ta način primeren le, če je ISP zaupanja vreden.

VPN se razlikujejo tudi po protokolu, ki je uporabljen za vspostavljanje tunela. Najpogosteje je za ustvarjanje tunela uporabljen protokol IPsec (IP security). IPsec izpolnjuje

večino varnostnih zahtev, to je zaupnost, avtentičnost in verodostojnost podatkov, ki se preko njega prenašajo. IP paketi se preko javnega omrežja prenašajo šifrirano zaviti v IPsec pakete. Šifrirajo in dešifrirajo se na začetku in koncu IPsec tunela.

Poleg IPsec obstaja se za vzpostavljanje VPN lahko uporabljajo tudi drugi protokoli, kot na primer SSL (Secure Socket Layer) in TLS (transport Layer Security) , ki sta uporabljena v OpenVPN, ali pa DTLS (datagram Transport Layer Security), ki ga uporablja proizvajalec omrežne opreme Cisco v svojem AnyConnect VPN.

VPN je mogoče vzpostaviti tudi z nešifrirano povezavo preko omrežja zaupanja vrednega ISP. V tem primeru se lahko vzpostavi tunel preko MPLS (Multi-Protocol Label Switching). Usmerjanje paketov znotraj MPLS omrežja ne poteka na osnovi IP naslovov temveč se paketi usmerjajo na MPLS stikalih na osnovi MPLS label. Preko MPLS se lahko vzpostavi navidezna linija med dvema točkama, lahko se poveže dve lokalni omrežji na povezavnem sloju , tako da delujeta kot eno samo lokalno omrežje (na primer Ethernet), ali pa se povežeta dve omrežji na omrežnem sloju, tako, da se paketi med njima usmerjajo po IP protokolu preko robnih usmerjevalnikov MPLS omrežja.

## 7.6 Varnost brezžičnih lokalnih omrežij

V brezžičnem lokalnem omrežju (WLAN - Wireless Local Area Network) poteka brezžična komunikacija preko radijskih valov. Radijsko valovanje lahko sega izven prostorov, v katerih so naprave vključene v lokalno omrežje, zato je taka komunikacija izpostavljena prisluškovanju in napadom iz bližnje okolice. Če želimo lokalno omrežje zaščititi pred napadi, mora biti komunikacija v WLAN šifrirana.

Večina WLAN deluje na osnovi standarda WiFi (IEEE 802.11). V prvi verziji tega standarda je bil za šifriranje uporabljen postopek WEP (Wired Equivalent Privacy). Čeprav večina naprav, ki omogoča WiFi povezavo, še vedno podpira WEP, pa njegova uporaba ni priporočljiva. Zelo hitro je bila namreč odkrita šibkost v tem postopku. Danes je mogoče zaščito WEP razbiti prej kot v eni minuti.

Postopek WEP je kasneje, v standardu (IEEE 802.11i) nadomestil postopek WPA (WiFi Protected Access) in nato še nekoliko izboljšan WPA2. WPA2 je danes edini, ki je priporočljiv za uporabo, ker pa WPA2 ni združljiv s starejšo opremo, se še vedno uporablja tudi WPA.

WPA2 je napreden postopek za avtentikacijo in šifriranje. Ima dva načina delovanja "WPA Personal", pri katerem je geslo za vse uporabnike enako in "WPA Enterprise", kjer ima vsak uporabnik svoje geslo. Pri WPA Enterprise se izvaja vzajemna avtentikacija, to je avtentikacija uporabnika in avtentikacija dostopovne točke. Za avtentikacijo se uporabljajo digitalna potrdila, avtentikacija pa poteka v povezavi s strežnikom Radius. Pri WPA Personal, ki je namenjen predvsem domači uporabi pa se izvaja zgolj avtentikacija uporabnika na osnovi gesla. WPA2 uporablja za šifriranje CCMP (Counter Mode Cipher Block Chaining Message Authentication Code Protocol) pri katerem so podatki šifrirani z AES šifro.

# Viri in literatura


[1] C. E. Shannon, Theory of Secrecy Systems, Bell Systems Technical Journal, zvezek 28, 1948.

[2] A.Menezes, P. van Oorschot, and S. Vanstone, Handbook of Applied Cryptography, CRC Press, 1996.

[3] Bruce Schneier, Applied Cryptography, Second Edition: Protocols, Algorthms, and Source Code in C,Wiley Computer Publishing, John Wiley & Sons, Inc, 1995.

[4] Susan Landau, Communications Security for the Twenty-first Century: The Advanced Encryption Standard, Notices of the AMS, zvezek 47, številka 4.

[5] Advanced encryption standard (AES), Federal Information Processing Standards Publication 197. 2001.

[6] Martin E. Hellman, An overview of public key criptography, IEEE Communications Magazine, zvezek 16, številka 6, 1978.

[7] Gustavus J. Simmons, urednik, Contemporary Criptology – The science of information integrity, IEEE Press, 1992.

[8] Tone Vidmar, Informacijsko komunikacijski sistem, Založba Pasadena, 2002.

[9] W. Diffie, M. Hellman, New directions in cryptography, IEEE Transactions on information theory, zvezek 22, 1967.

[10] R.L. Rivest, A. Shamir, and L. Adleman A Method for Obtaining Digital Signatures and Public-Key Cryptosystems, Communications of the ACM, zvezek 21 številka 2, 1978.

[11] Sašo Tomažič, Secure communications over the Internet, The European journal of teleworking, zvezek 4, številka 1, 1996.

[12] Sašo Tomažič, Varnost v telekomunikacijah in kako jo zagotovit, zbornik Varnost v telekomunikacijskih sistemih, Elektrotehniška zveza Slovenije, 2003.

[13] Goran Bobojević, Pregled sodobnih šifrirnih postopkov, magistrsko delo, Univerza v Ljubljani, Fakulteta za elektrotehniko, 2005.

[14] Aleš Zavec, Napadi na informacijsko-komunikacijski sistem in zaščita pred njimi, magistrsko delo, Univerza v Ljubljani, Fakulteta za elektrotehniko, 2013.

[15] Check point 2013 security report, Check point software technologies, 2013.

[16] Top Ten Reviews, *http://www.toptenreviews.com/*, dostopano 9. decembra 2013

[17] Aleš Zavec, Sašo Tomažič, Malware in Windows environment: what is it and how to prevent it, zbornik YUINFO 2006, Kopaonik, 2006.

[18] CVE details, *http://www.cvedetails.com/*, dostopano 16. decembra 2013.

## Programska oprema

[19] **Microsoft Baseline Security Analyzer**,
*http://www.microsoft.com/en-us/download/details.aspx?id=7558*,
dostopano 9. decembra 2013.

[20] **Microsoft Network Monitor**,
*http://www.microsoft.com/en-us/download/details.aspx?id=4865*,
dostopano 9. decembra 2013.

[21] **GFI LanGuard Network Security Scanner**,
*http://www.gfi.com/products-and-solutions/network-security-solutions/gfi-languard*,
dostopano 9. decembra 2013.

[22] **Shadow Security Scanner**,
*http://www.safety-lab.com/en/products/securityscanner.htm*,
dostopano 9. decembra 2013.

[23] **Retina Network Security Scanner Unlimited**,
*http://www.beyondtrust.com/Products/RetinaNetworkSecurityScanner/*,
dostopano 9. decembra 2013.

[24] **Nessus Vulnerability Scanner**,
*http://www.tenable.com/products/nessus*
dostopano 9. decembra 2013.

[25] **CORE Impact Pro**,
*http://www.coresecurity.com/core-impact-pro*,
dostopano 9. decembra 2013.

[26] **Metasploit**,
*http://www.rapid7.com/products/metasploit/*,
dostopano 9. decembra 2013.

[27] **QuasarsGuard**,
*http://www.qualys.com/enterprises/qualysguard/*,
dostopano 9. decembra 2013.

[28] **Mcaffe SuprScan**,
*http://www.mcafee.com/us/downloads/free-tools/superscan.aspx*,
dostopano 12. decembra 2013.

[29] **Comodo Internet Security Complete**
*http://www.comodo.com/home/internet-security/internet-security-complete.php*,
dostopano 14. decembra 2013.

[30] **Agnitum Outpost Pro Firewall**,
*http://www.agnitum.com/products/outpost/*,
dostopano 14. decembra 2013.

[31] **Kaspersky Internet Security**,
*http://www.kaspersky.com/internet-security*,
dostopano 14. decembra 2013.

[32] **ZoneAlarm Pro Firewall**,
*http://www.zonealarm.com/security/en-us/zonealarm-pro-firewall-anti-spyware.htm*,
dostopano 14. decembra 2013.

[33] **Total Defense Internet Security Suit**,
*http://store.totaldefense.com/ca/products/internetsecurity/internetsecurity_suite.asp*,
dostopano 14. decembra 2013.

[34] **Windows Sysnternals**,
*http://technet.microsoft.com/en-US/sysinternals*,
dostopano 14. decembra 2013.

[35] **Metasploit**,
*http://www.metasploit.com/download/*,
dostopano 16. decembra 2013.

[36] **Process Guard**,
*http://www.softpedia.com/get/Security/Security-Related/Process-Guard.shtml*,
dostopano 16. decembra 2013.

[37] **Bitdefender Total Security**,
*http://www.bitdefender.com/*,
dostopano 16. decembra 2013.

[38] **Kaspersky Antivirus**,
*http://www.kaspersky.com/*,
dostopano 16. decembra 2013.

[39] **F-Secure Anti-Virus**,
*http://www.f-secure.com*,
dostopano 16. decembra 2013.

[40] **Norton Antivirus Plus**,
*http://www.symantec-norton.com/*,
dostopano 16. decembra 2013.

[41] **Avast Pro Antivirus**,
*http://www.avast.com/en-eu/index*,
dostopano 16. decembra 2013.

[42] **Sandboxie**,
*http://www.sandboxie.com/*,
dostopano 22. decembra 2013.

[43] **ThinApp**,
http://www.vmware.com/products/thinapp/,
dostopano 22. decembra 2013.