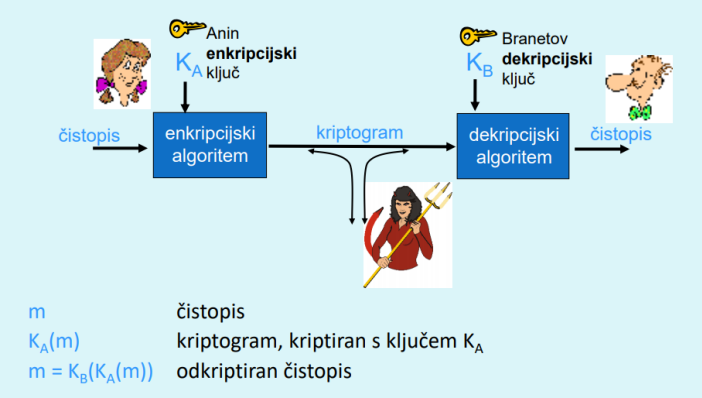
Kriptografija

Omrežna varnost

- varnosti izzivi pri elektronski komunikaciji:
    - **zaupnost**: samo pošiljatelj in prejemnik naj bi lahko brala sporočilo
        - preprečevanje **prisluškovanja (eavesdropping)**: prestrezanje sporočil (pasivni napad)
    - **integriteta (integrity control):** pošiljatelj in prejemnik želita zagotoviti, da sporočilo med prenosom ni bilo spremenjeno
        - prečevanje dodajanja, brisanja in **spreminjanje** sporočil (aktivni napad)
    - **identifikacija** in **avtentikacija:** pošiljatelj in prejemnik želita preveriti in potrditi medsebojni identiteti
        - preprečevanje **kraje identitete (impersonation):** napadalec lahko ponaredi izvorni naslov v paketu
        - preprečevanje **ugrabitve seje (hijacking):** napadalec odstrani pošiljatelja in prevzame njegovo vlogo
        - omogočanje **avtorizacije:** katere aktivnosti lahko uporabnik izvaja v sistemu
        - preprečevanje **zanikanja komunikacije** - non repudiation, ne moreš reči, da nisi ti po avtentikaciji(avtentikacija = dokazovanje identitete)
- primeri aplikacij: elektronske transakcije preko spleta (npr. nakupi), elektronsko bančništvo, DNS strežniki, usmerjevalniki...
- varnost v praksi (operativna varnost):
    - naprave: **požarni zidovi**, sistemi za **zaznavanje/preprečevanje vdorov**(Intrusion Detection System/Intrusion Prevention System),
    - dodaten cilj: preprečevanje onemogočanja storitev (denial of service): onemogoči uporabo storitev (npr. s preobremenitvijo virov)
    - varnost je **potrebna na vseh plasteh:** na aplikacijski, transportni, omrežni in povezavni plasti
    - **fizična plast:** kriptiranje povezave
    - **omrežna plast:** filtriranje paketov, opazovanje prometa
    - **transportna plast:** kriptiranje povezav med dvema procesoma
    - **aplikacijska plast:** avtentikacija na podlagi preverjanja identitete

Terminologija



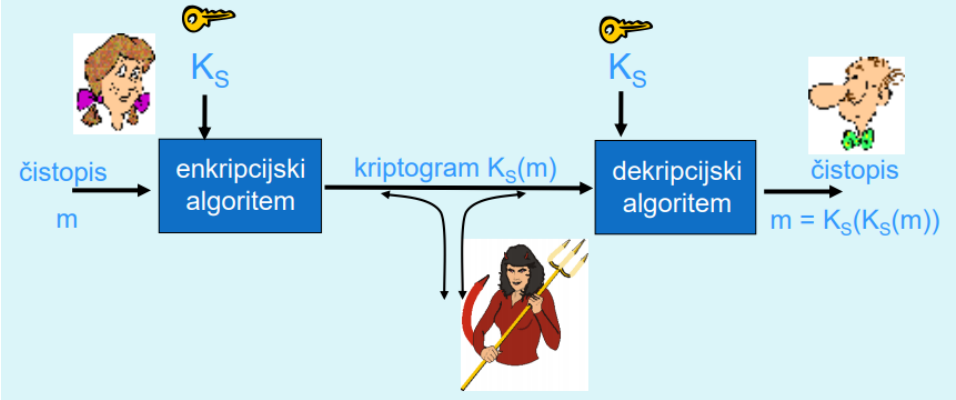
m ... message
c ... cyphertext

Kriptografija

- običajno se uporablja
  - **algoritem(postopek)**, ki je **znan** vsem
  - **tajni** so samo **ključi**
- **vrste kriptografije glede na ključe:**
  - **simetrični ključi:** enkripcijski in dekripcijski ključ sta enaka. Hitra
  - **javni ključi(asimetrična):** uporaba dveh ključev, enega javnega in drugega tajnega. Počasna
- zgoščevalne funkcije (hash functions)
  - so tudi enkripcijski algoritem, vendar ne uporablja ključev

Enkripcijski algoritem parametriziramo z enkripcijskim ključem.

Kriptografija s simetričnim ključem



- pošiljatelj in prejemnik uporabljata isti (simetričen) ključ
- primer ključa: dogovor o zamenjavi znakov v sporočilu (substitucijski vzorec)
- kako si pošiljatelj in prejemnik varno izmenjata ključ? Gre za problem distribucije ključev, ena rešitev so javni ključi.

Metode kriptiranja

- glede na način kriptiranja (algoritem)
  - **substitucija** (zamenjava znakov z drugimi)
    - Cezarjev kriptogram
    - Vigenèr-jev kriptogram
    - Porterjev kriptogram
  - **transpozicija** (zamenjava vrstnega reda znakov)
  - **sodobne (kombinirane) metode**
- glede na velikost sporočila (podatkov)
  - znakovna
  - bločna (kriptiramo zaporedja znakov)
- glede na uporabljene ključe
  - **simetrična** kriptografija (oba udeleženca uporabljata enak ključ)
    - znakovna (bit po bit)
    - bločna (sporočilo razbijemo na bloke, vsakega kriptiramo neodvisno)
  - **kriptografija z javnimi ključi** (ključa za enkripcijo in dekripcijo sta različna)

Substitucijske metode

Cezar

- Cezarjev kriptogram: substitucija z zamikom za k črk
- monoalfabetna substitucija
- ključ je k (velikost zamika)
- imamo 25 možnih ključev
- kriptogram razbijemo v največ 25 poskusih

Primer

- k = 21
- kriptogram JULUA = čistopis NAPAD

a	b	c	č	d	e	f	g	h	i	j	k	l	m	n	o	p	r	s	š	t	u	v	z	ž
u	v	z	ž	a	b	c	č	d	e	f	g	h	i	j	k	l	m	n	o	p	r	s	š	t

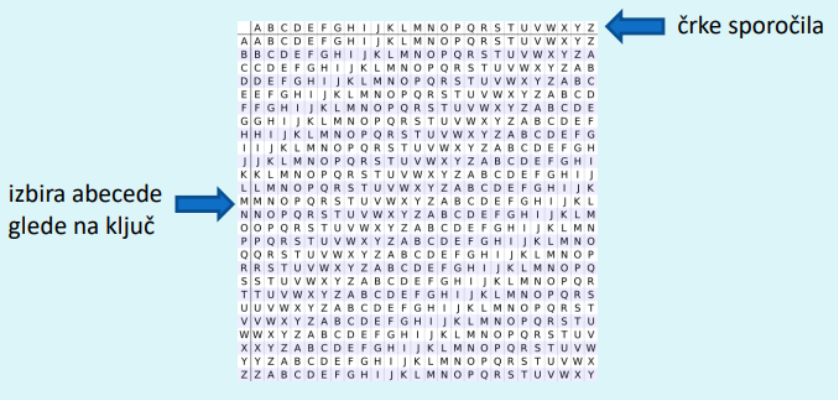
Kriptoanaliza substitucijskih kriptogramov

Kriptoanaliza (razbijanje kriptogramov) na osnovi:

- **poznanega besedila** (npr. “please login” ali "HTTP/1.1")
  - zato kriptiramo le vsebino, ne cele komunikacije
- **statistike jezika** (črke, besede, dvo- ali tročrkovni sklopi – potrebno je daljše besedilo).
- **poznavanja vsebine** (semantika) olajša razbijanje – iščemo pričakovane korene besed ipd.

Vigenèr-jev kriptogram – večabecedno kriptiranje

- Viegenerjeva matrika: vse Cezarjeve abecede.
- ključ = niz D črk, vsaki pripada ena vrstica (enaka 1. črka).
- z abecedo n-te črke gesla kriptiramo n-to, n+D-to, n+2D-to … črko sporočila.
- preprost ključ
- statistika jezika in semantika postaneta nemočni

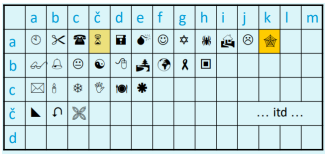


Primer

r	a	č	u	n	a	l	n	i	š	k	e	k	o	m	u	n	i	k	a	c	i	j	e
j	u	n	i	j	a	v	s	i	i	z	p	i	t	i	n	a	ž	a	l	o	s	t	o
d	p	a	d	e	j	o	r	a	z	e	n	p	r	i	e	k	o	n	o	m	i	k	i
S	e	p	t	e	m	b	r	a	p	a	b	o	s	p	e	t	v	s	e	p	o	s	t
a	r	e	m																				

- Geslo(ključ): računalniške komunikacije
- Sporočilo: Junija vsi izpiti na žalost odpadejo, razen pri ekonomiki, septembra pa bo spet vse po starem
- dolžina ključa: D=24
- Prvi stolpec črk (torej 1., 25. (1+24), 49. (1+48) črko) kriptiramo z 18. abecedo (r), itd.

Porterjev kriptogram



- Kriptiramo po 2 znaka hkrati.
- Simboli so v tabeli – vrstica za en, stolpec za drugi znak.

Kodiranje

- cel znak ali besedo nadomestimo z drugo
- ni splošnega pravila za zamenjave
- ključ predstavlja cela kodna tabela

Transpozicijske metode

Transpozicijski kriptogram

- spremenimo zaporedje znakov ali delov besedila
- uporabimo ključ, oštevilčimo črke po abecedi
- zapišemo stolpce glede na oštevilčenje črk

k	o	p	r	i	v	a
3	4	5	6	2	7	1
J	u	n	i	j	a	n
a	ž	a	l	o	s	t
v	s	i	i	z	p	i
t	i	o	d	p	a	d
e	j	o	b	l	a	b

← Ključ

← Katera po abecedi je črka ključa

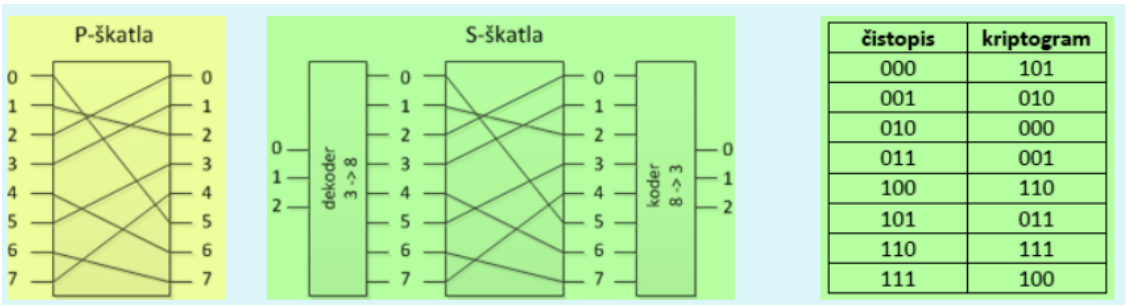
Sodobna simetrična kriptografija

- bločna kriptografija: sporočilo m kriptiramo tako, da obdelamo posamezne bloke k bitov (npr. bloke po 64 bitov)
- preslikava med bloki sporočila in kriptograma je bijektivna (enolična)
- primer, če k=3

vhod	izhod
000	110
001	111
010	101
011	100
100	011
101	010
110	000
111	001

Bločna kriptografija

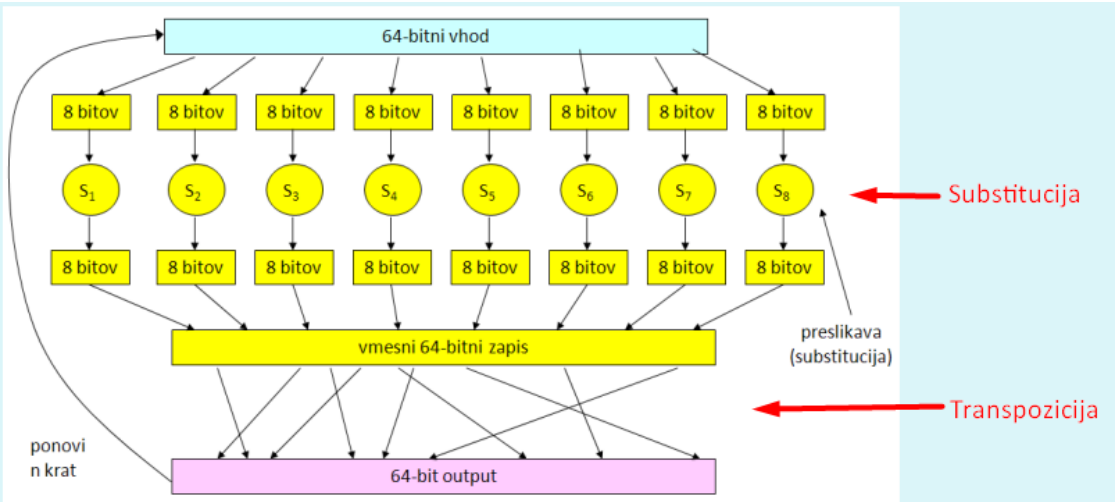
- permutacijska škatla (s ključem 23157046)
- substitucijska škatla (dekoder-permutacija-koder) oz. (decoder + P-box + encoder)



čistopis	kriptogram
000	101
001	010
010	000
011	001
100	110
101	011
110	111
111	100

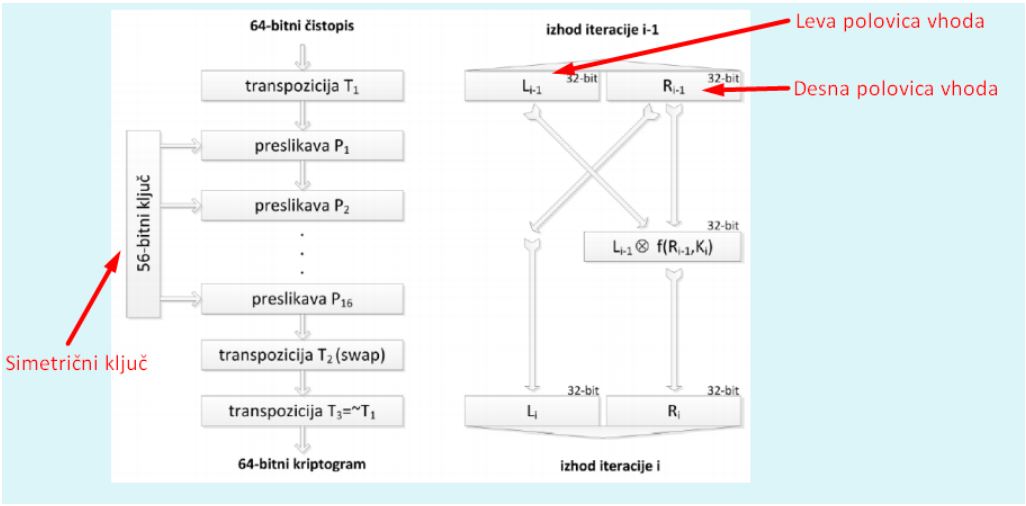
- možno je kombiniranje škatel v preslikovalno kaskado za poenostavitev logike

- problem:
  - če k=3, imamo 40320 (=2k!=8!) permutacij vseh možnih vhodov (možno razvozlati kodo na domačem PCračunalniku)
  - če k=64, je permutacij ogromno (=264!) in je težko hraniti tabelo permutacij (težka tudi izmenjava ključev)
- rešitev: uporabimo preprosto funkcijo za kriptiranje, ki pa simulira veliko tabelo



- **primeri algoritmov** za bločno kriptografijo
  - DES (Data Encryption Standard)
  - 3DES (3-kratni DES s 3 različnimi ključi)
  - AES (Advanced Encryption Standard)
- zgornji algoritmi uporabljajo KLJUČE
  - DES 64-bitne bloke in 56-bitni ključ;
  - AES 128-bitne bloke in 128/192/256-bitne ključe

DES (Data Encryption Standard)



- DES je kombinacija transpozicijskih in substitucijskih metod
- transpozicija -> 16 preslikav (ključ!) -> SWAP -> transpozicija

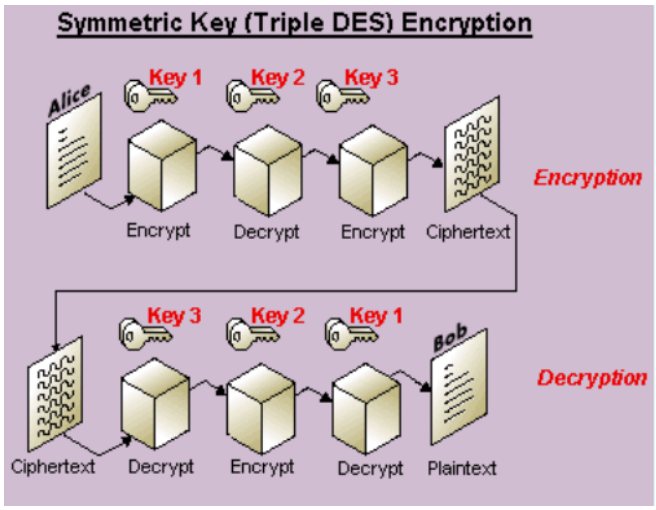
SWAP: zamenja zgornjo in spodnjo polovico bitov v številu (npr. 1100 -> 0011)

Zadnja transpozicija je inverz prve.

3DES (trojni DES)

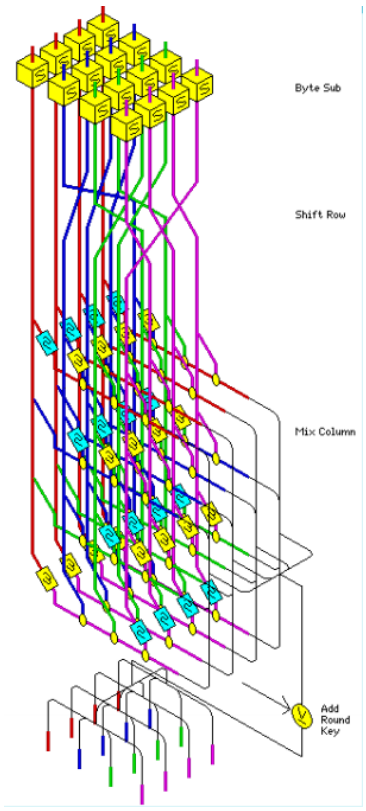
- 3 x kriptiranje (kriptiranje(K1), dekriptiranje(K2), kriptiranje(K3))
- 3 x počasnejši
- 2^(2*56)(zato ker sta še dva dodatna 56 bitna ključa) krat varnejši za napad z grobo silo
- združljivost z DES (če K2 == K3)

Dekriptiranje se izvede v obratni smeri od kriptiranja, zato se ti dve nasprotni fazi izničita, če imata enaka ključa (K2 == K3)



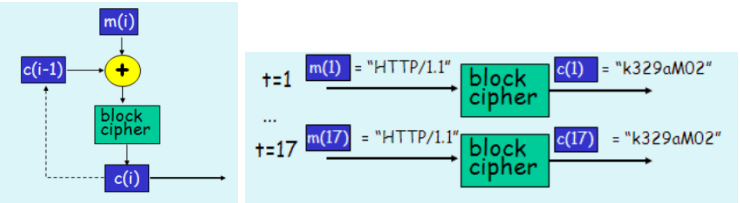
AES

- **daljši bloki**
- **lahko izbiramo dolžino ključa**
- bloki 128 bitov, ključ 128/192/256 bitov
- dober računalnik bi potreboval 1010 let za razbijanje
- različne AES operacije: zamikanje vrstic, zamenjave stolpcev, izpeljave ključev



Verižno kriptiranje blokov

- bločna kriptografija ni praktična za pošiljanje velikih sporočil, ker dajejo ponavljajoči se bloki (npr. HTTP/1.1) iste kriptograme (možno razbijanje sporočila)
- **rešitev: verižno kriptiranje**
  - pošiljatelj s prvim sporočilom pošlje neko naključno vrednost (inicializacijski vektor) c(0)
  - pošiljatelj kriptira vsako sporočilo v kriptogram: **$c(i) = K_S(m(i) \text{ XOR } c(i-1))$**
  - prejemnik odkriptira sporočilo: **$m(i) = K_S(c(i)) \text{ XOR } c(i-1)$**
- ni potrebno, da je IV tajen
- omogoča, da se isto sporočilo kriptira v različne kriprograme!



Primer

m=010010010, IV=c(0)=001, k=3

pošiljatelj:

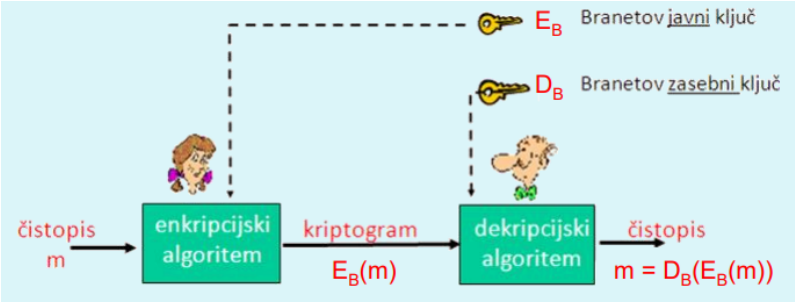
- $c(1) = K_S(m(1) \text{ XOR } c(0)) = K_S(010 \text{ XOR } 001) = K_S(011) = 100$
- $c(2) = K_S(m(2) \text{ XOR } c(1)) = K_S(010 \text{ XOR } 100) = K_S(110) = 000$
- $c(3) = K_S(m(3) \text{ XOR } c(2)) = K_S(010 \text{ XOR } 000) = K_S(010) = 101$

Razbijanje bločnih kriptosistemov z grobo silo(ocene iz 2013)

Ključ	Oseba	Mala skupina	Razisk. omrežje	Veliko podjetje	Vojska
40	sekunde	milisekunde	milisekunde	mikrosekunde	nanosekunde
56	dan	ure	ure	sekunde	mikrosekunde
64	tedni	tedni	dnevi	ure	milisekunde
80	tisočletja	tisočletja	stoletja	leta	minute
128	∞	∞	∞	∞	∞

Asimetrična kriptografija

- enkripcijski (E) in dekripcijski (D) ključ sta lahko različna
- ključa sta si "inverzna" - operacija nad enim izniči učinek drugega
- E je lahko javen, D mora biti tajen. Zahteve:
  - D(E(m)) = m in E(D(m)) = m
  - iz m in E(m) je nemogoče ugotoviti D
  - iz E je nemogoče ugotoviti D
- primer: algoritem RSA (Rivest, Shamir, Adleman)



Algoritem RSA

- izberemo p,q: veliki praštevili (1024 bitov)
  - n = pq
  - z = (p-1)(q-1)
- izberemo e tako, da nima skupnih deliteljev z z.
- izberemo d tako, da ed mod z = 1
- definiramo:
  - javni ključ: E = (n,e)
  - zasebni ključ: D = (n,d)

$$m \rightarrow c = m^e \bmod n \quad \textit{kriptiranje}$$
$$c \rightarrow m = c^d \bmod n \quad \textit{dekriptiranje}$$

RSA je **počasen** zato se ga navadno **uporablja** le **za začetek komunikacije**, da varno izmenjamo simetrične ključe nato pa nadaljujemo z eno od simetričnih metod. To **reši problemm distribucije simetričnih ključev**.

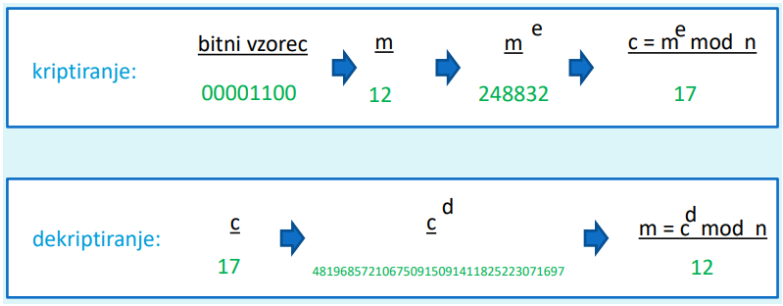
Zakaj je RSA varen?

- Denimo, da poznamo javni ključ (n,e). RSA je varen, ker je v praksi težko poiskati delitelja p in q števila n, saj za to ne obstaja učinkovit algoritem.
  - iskanje deliteljev 500-mestnega števila bi trajalo 10^25 let

RSA primer

Brane izbere: p=7, q=5

- sledi n=35, z=24
- izberemo e=5 (e in z sta si tuji števili) in d=29 (da je ed-1 deljivo z z)



Principi varne komunikacije (avtentikacija, integriteta, elektronski podpis)

Avtentikacija udeležencev

- Avtentikacija je dokazovanje identitete
- želimo imeti komunikacijsko okolje, v katerem lahko udeleženci preverijo, da:
- je pošiljatelj dejansko oseba, za katero se predstavlja
- je prejemnik zagotovo prava oseba in ne prisluškovalec

mehanizmi za avtentikacijo z RSA:

- imamo pošiljatelja A in prejemnika B
- avtentikacija **pošiljatelja**:
  - sporočilo m kriptiramo v $D_A(m)$, prejemnik odkriptira v $E_A(D_A(m)) = m$
- avtentikacija **prejemnika**:
  - sporočilo m kriptiramo v $E_B(m)$, prejemnik odkriptira v $D_B(E_B(m)) = m$
- **vzajemna** avtentikacija:
  - sporočilo m kriptiramo v $D_A(E_B(m))$, prejemnik odkriptira v $D_B(E_A(D_A(E_B(m)))) = m$

Pri RSA ni pomembno v katerem vrstnem redu izvedemo EA, DB itd., saj dokler uporabimo vse 4 ključe, bomo vedno lahko zakriptirali/odkriptirali sporočilo ne glede na vrstni red uporabe ključev.

Integriteta komunikacije

- želimo imeti komunikacijsko okolje, v katerem lahko udeleženci preverijo, da:
  - sporočilo ni bilo spremenjeno,
  - je sporočilo "sveže" (ni ponovljeno posneto staro sporočilo - replay attack oz. ponovitev komunikacije),

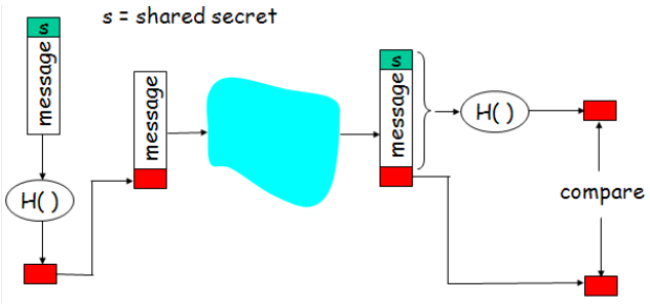
Tako se izognemo aktivnemu napadu(prisluškovanje, prestrezanje, spreminjanje in posredovanje spremenjenega sporočila).

- uporabljamo zgoščevalne kriptografske funkcije:
  - preprosto izračunljive,
  - iz H(m) ne moremo ugotoviti m
  - težko je najti m in m', da velja H(m)=H(m') (birthday attack oz. rojstnodnevni napad)
  - izgleda kot naključen niz
  - primeri: MD5, SHA-1

Birthday attack: sporočilo m spremenimo v sporočilo m', tako, da bo zgoščevalna funkcija vrnila enako zgoščeno vrednost. Tako lahko spremenimo sporočilo brez da bi nas odkrili.

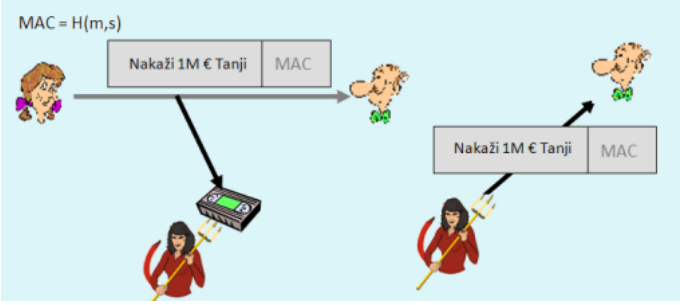
Zgoščena vrednost sporočila

- s sporočilom m pošljemo tudi H(m), prejemnik preveri, ali se ujemata (dokaz o integriteti sporočila, ne pa o avtentikaciji)
- za preverjanje avtentikacije uporabimo še avtentikacijski ključ (shared secret). Pošljemo m in H(m+s) = MAC (message authentication code)
- shared secret poznata le prejemnik in pošiljatelj (pri vzpostavitvi si ga izmenjata na varen način, recimo z RSA). Shared secret preprečuje spreminjanje sporočila vsem, ki ne poznajo shared secret-a, zato ker ne morejo doseči enak hash ob spreminjanju sporočila.

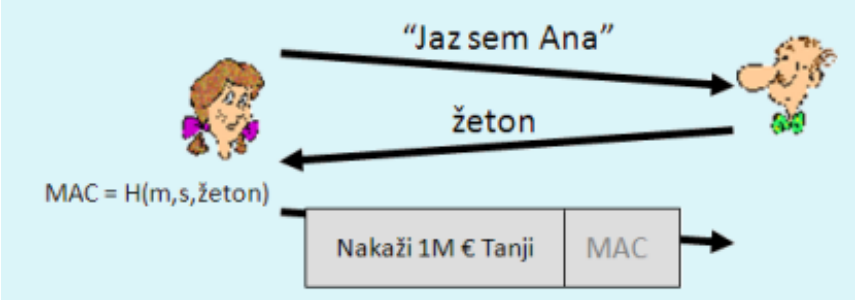


Napad s ponavljanjem komunikacije

- napadalec lahko shrani celotno (avtenticirano!) komunikacijo in jo kasneje ponovno izvede



- rešitev: uporabimo enkraten podatek (nonce, žeton, sol), ki ga zgostimo skupaj s sporočilom in ključem: H(m, s, žeton)



Tudi če napadalec pošlje identitčno sporočilo, strežnik ve, da se gre za neveljavne podatke saj je žeton že bil uporabljen in bi pri ponovitvi sporočila morali imeti novi žeton. Žetone pa izdaja strežnik.

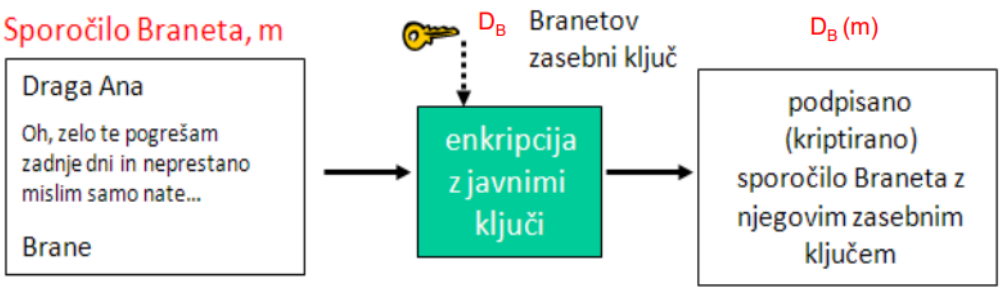
Digitalni podpis

- način za zamenjavo osebnega podpisa (informacija, ki enolično potrjuje identiteto posameznika)
- avtentikacija z MAC ni enolična, uporabimo kriptografijo z javnimi ključi

NAČIN 1

EB = Branetov javni ključ DB = Branetov zasebni ključ

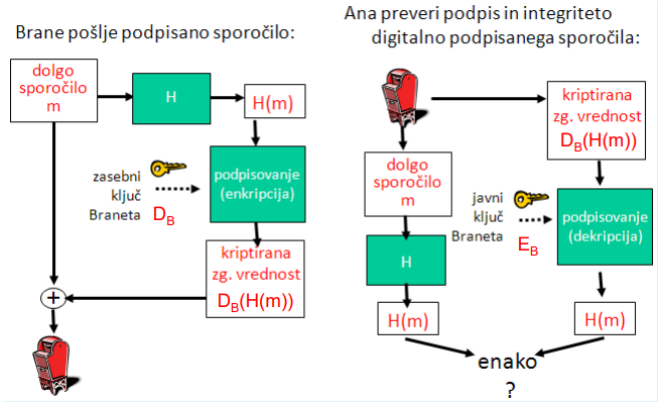
- pošiljatelj B lahko izračuna DB(m), kar lahko predstavlja podpisan dokument
- prejemnik izračuna EB(DB(m)) = m
- časovno zahtevno pri dolgih sporočilih



Avtenticiramo le pošiljatelja. Poslano sporočilo (DB(m)) lahko dekriptira in prebere kdorkoli, saj je potreben ključ za dekripcijo(EB) javno dostopen.

NAČIN 2 (ta se uporablja v praksi)

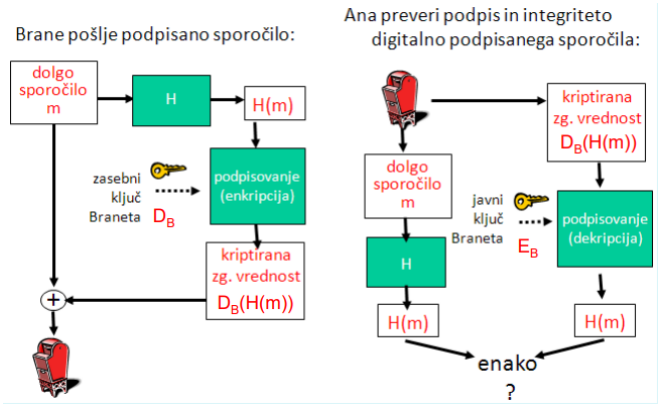
- pošiljatelj B uporabi zgoščevalno funkcijo H in podpiše le zgoščeno vrednost
- skupaj z originalnim (nekriptiranim) sporočilom pošlje tudi kriptirano zgoščeno vrednost



Pri **javnih ključih** se pojavi problem **kraje identitete**. Nekdo lahko reče, da je njegov ključ od nekoga drugega. Nimamo dokazil, da je javni ključ res od tistega, ki ga je poslal.

Certifikacijska agencija

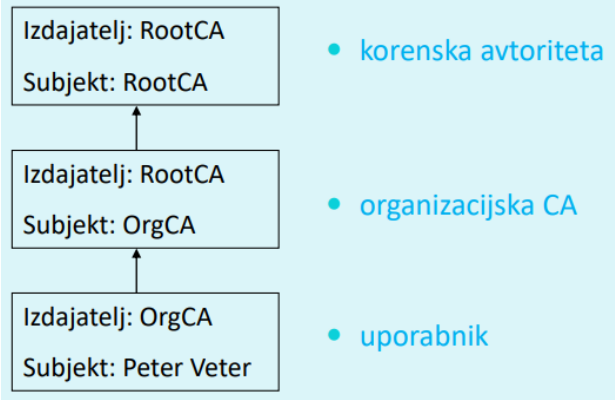
- pri digitalnem podpisovanju lahko vdiralec podpiše sporočilo s svojim zasebnim ključem in se pretvarja, da je to ključ od nekoga drugega
- **rešitev**:
  - certifikacijske agencije (certification authority) preverjajo povezavo med javnim ključem in identiteto osebe.
  - certifikacijska agencija shrani povezavo med ključem in identiteto v CERTIFIKAT (tega agencija podpiše s svojim zasebnim ključem, kot dokazilo, da je certifikat res od te agencije)
  - npr. pri SIGEN-CA smo morali pokazati osebno izkaznico in ostale podatke, po naročitvi pa je prišlo del podatkov po pošti in del po mailu (dve distribucijske poti). Za prevzem certifikata smo potrebovali oba dela, tako se preprečuje(otežuje) kraja certifikatov in s tem tudi identitete.
  - Vse certifikacijske agencije morajo zastopati svoje certifikate.



Veriga zaupanja certifikatov

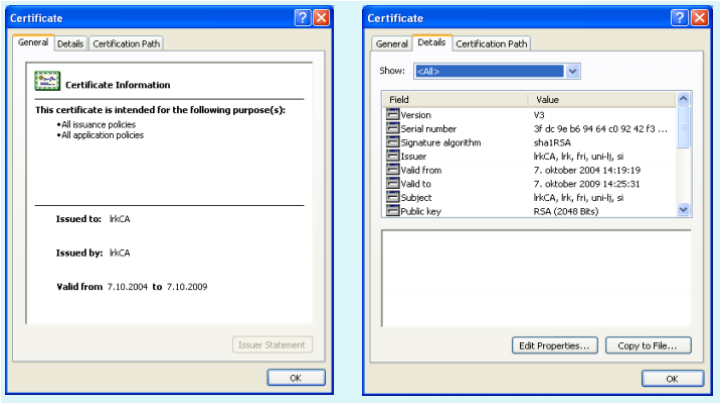
Podobno kot pri hierarhiji DNS strežnikov bi bila tudi tu slaba ideja vse certifikate vezati na en strežnik (skalabilnost, single point of failure) zato:

- veljavnost certifikatov preverimo z verigo zaupanja
- so korenske avtoritete znane(v računalniku/brskalniku imamo Trusted Certificate Authorities)



Certifikati

- certifikat vsebuje: ime izdajatelja, ime osebe, naslov, ime domene, javni ključ osebe, digitalni podpis(podpis z zasebnim ključem izdajatelja)
- uveljavljen standard za zapis certifikatov je X.509



Operativna varnost

Varnost v omrežju

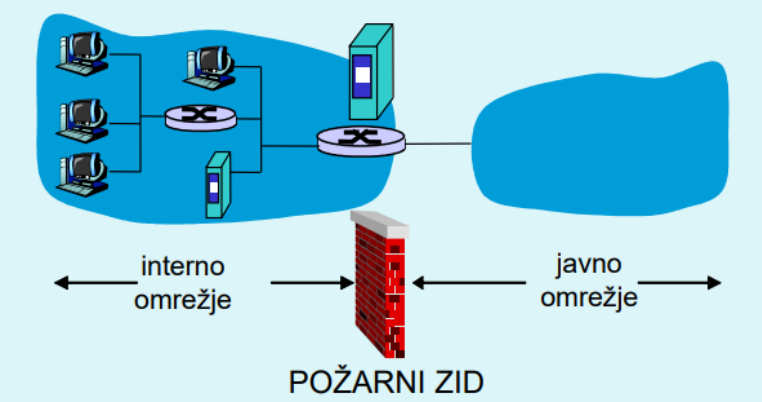
- Administrator omrežja lahko uporabnike deli na:
- good guys: uporabniki, ki legitimno uporabljajo vire omrežja, pripadajo organizaciji,
- bad guys: vsi ostali, njihove dostope moramo skrbno nadzorovati
- Omrežje ima običajno eno samo točko vstopa, kontroliramo dostope v njej:
- požarni zid (firewall)
    - ima varnostno politiko (kaj gre noter/ven)
- sistem za zaznavanje vdorov(IDS, intrusion detection system)
- sistem za preprečevanje vdorov(IPS, intrusion prevention system)

Razlika med IDS in IPS je, da IDS le zaznavajo, IPS pa zaznajo in ukrepajo.

Požarni zid (firewall)

Izolira interno omrežje od velikega javnega omrežja, določenim paketom dovoli prehod, druge blokira.

Ima 3 naloge: • filtrira **ves** promet, • prepušča samo promet, ki je **dopusten** glede na politiko, • je **imun** na napade (ne more ga napadalec kar izklopiti)



Požarni zid: vrste filtriranj

1. izolirano filtriranje paketov (angl. stateless, traditional) - na 3. in 4. plasti ISO/OSI
    - pretežno filtriranje na podlagi podatkov v glavi: izvorni in ponorni naslovi ter vrata
2. filtriranje paketov v kontekstu (angl. stateful filter) - na 3. in 4. plasti ISO/OSI
    - nadzoruje vzpostavljenost povezave
3. aplikacijski prehodi (angl. application gateways) - na 7. plasti ISO/OSI
    - filtriranje z vpogledom v podatke aplikacijske plasti (vsebina, aplikacijski protokol, uporabniško ime...)

Izolirano filtriranje paketov (stateless/traditional)

- filtriranje običajno izvaja že usmerjevalnik, ki meji na javno omrežje. Na podlagi vsebine paketov seodloča, ali bo posredoval posamezen paket,
- **odločitev na podlagi:**
  - IP izvornega/ponornega naslova
  - številke IP protokola: TCP, UDP, ICMP, OSPF itd.
  - ** TCP/UDP izvornih in ciljnih vrat**
  - tip sporočila pri protokolu ICMP
  - zastavice TCP: SYN in ACK bit (sta aktivni za prvi segment pri povezovanju, nadzorujemo dopustnostvzpostavljanja povezave)

Izolirano filtriranje: dostopovni seznami

- dostopovni seznam (angl. access control list, ACL)
- tabela pravil, upošteva (procesira) se jo od vrha proti dnu
- zapisi so par (pogoj, akcija)
- primer: onemogoči ves promet razen WWW navzven in DNS v obe smeri

izvorni naslov	ciljni naslov	protokol	izvorna vrata	ciljna vrata	zastavica	akcija	
222.22/16	izven 222.22/16	TCP	> 1023	80	any	dovoli	← Zahtevek
izven 222.22/16	222.22/16	TCP	80	> 1023	ACK	dovoli	← Odgovor
222.22/16	izven 222.22/16	UDP	> 1023	53	---	dovoli	
izven 222.22/16	222.22/16	UDP	53	> 1023	----	dovoli	← Odhodni porti so navadno > 1023
all	all	all	all	all	all	zavrzi	← Splošni primer (else)

Filtriranje paketov v kontekstu (stateful filter)

- angl. stateful filter, upošteva povezavo
  - izolirano filtriranje lahko dovoli vstop nesmiselnim paketom (npr. vrata = 80, ACK =1; čeprav notranji odjemalec ni vzpostavil povezave), ne gleda konteksta
- izboljšava: filtriranje paketov v kontekstu spremlja in vodi evidenco o vsaki vzpostavljeni TCP povezavi
  - zabeleži vzpostavitev povezave (SYN) in njen konec (FIN): na tej podlagi odloči, ali so paketi smiselni
  - po preteku določenega časa obravnavaj povezavo kot neveljavno (timeout)
  - uporabljaj podoben dostopovni seznam, ki določa, kdaj je potrebno kontrolirati veljavnost povezave(angl. check connection)

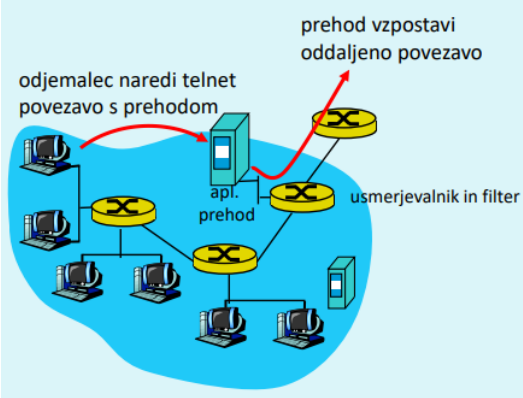
izvorni naslov	ciljni naslov	protokol	izvorna vrata	ciljna vrata	zastavica	akcija	preveri povezavo
222.22/16	izven 222.22/16	TCP	> 1023	80	any	dovoli	
izven 222.22/16	222.22/16	TCP	80	> 1023	ACK	dovoli	X
222.22/16	izven 222.22/16	UDP	> 1023	53	---	dovoli	
izven 222.22/16	222.22/16	UDP	53	> 1023	----	dovoli	X
all	all	all	all	all	all	zavrzi	

X pomeni, da preveri če povezava obstaja(ali so bili ti podatki prej zahtevani?) in če ne obstaja, ne spusti skozi.

Aplikacijski prehodi (application gateways)

- kos programske opreme (prejšnja dva sta lahko bila tudi namenski kos strojne opreme ali pa del usmerjevalnika)
- omogočajo dodatno filtriranje glede na izbiro uporabnikov, ki lahko uporabljajo določeno storitev
- omogočajo filtriranje na podlagi podatkov na aplikacijskem nivoju poleg polj IP/TCP/UDP.

Delovanje:



1. vsi uporabniki vzpostavljajo povezavo preko prehoda
2. samo za avtorizirane uporabnike prehod vzpostavi povezavo do ciljnega strežnika
3. prehod posreduje podatke med 2 povezavama
4. usmerjevalnik blokira vse povezave razen tistih, ki izvirajo od prehoda

Omejitve aplikacijskih prehodov:

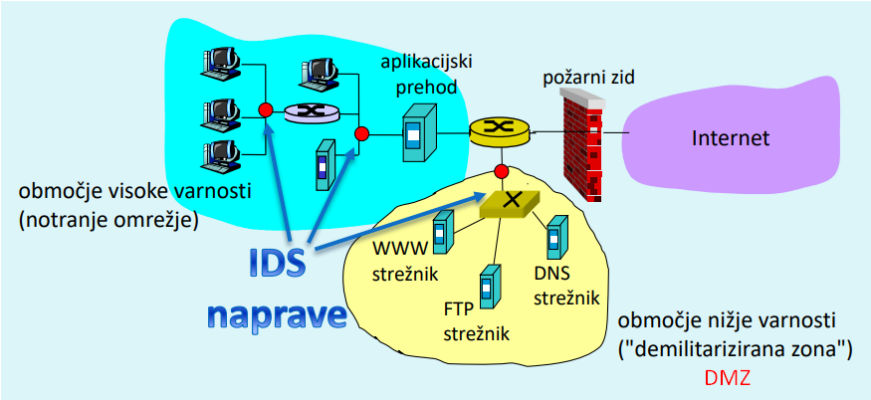
- če uporabniki potrebujejo več aplikacij (telnet, HTTP, FTP itd.), potrebuje vsaka aplikacija svoj aplikacijski prehod,
- kliente je potrebno nastaviti, da se znajo povezati s prehodom (npr. IP naslov medstrežnika v brskalniku)

Sistemi za zaznavanje vdorov

- dodatna naprava - IDS, ki izvaja poglobljeno analizo paketov. Na podlagi vstopa sumljivih paketov v omrežje lahko naprava prepreči njihov vstop ali razpošlje obvestila.
    - sistem za zaznavanje vdorov (IDS) pošlje sporočilo o potencialno škodljivem prometu
    - sistem za preprečevanje vdorov (IPS) ukrepa pri pojavitvi sumljivega prometa
    - primeri: Cisco, CheckPoint, Snort IDS

Načini zaznavanja vdorov (delovanje IDS/IPS)

1. primerjava s shranjenimi vzorci napadov (angl. **signatures**)
2. opazovanje netipičnega prometa (angl. **anomaly-based**)

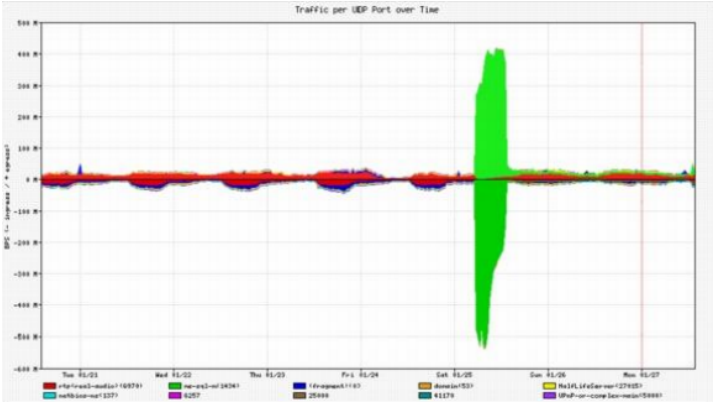


Zaznavanje z vzorci napadov (signatures)

- vzorci napadov lahko hranijo izvorni IP, ponorni IP, protokol, zaporedje bitov v podatkih paketa, lahko so vezani na serijo paketov
- varnost je torej odvisna od baze znanih vzorcev; IDS/IPS slabo zaznava še nevidene napade
- možni lažni alarmi
- zahtevno procesiranje (lahko spregleda napad). Včasih ne dohitevajo prometa in ga morajo spregledati.

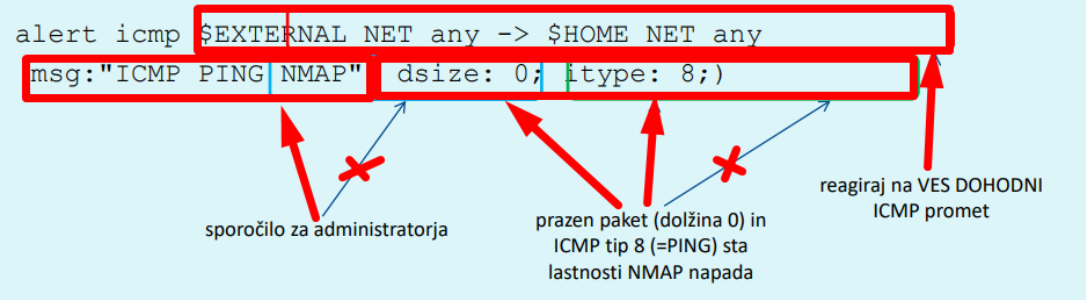
Zaznavanje netipičnega prometa (anomalies)

- sistem opazuje običajen promet in izračuna statistike, vezane nanj
- sistem reagira na statistično neobičajen promet (npr. nenadno velik delež ICMP paketov)
- možno zaznavanje še nevidenih napadov
- težko ločevanje med normalnim in nenavadnim prometom
- možni lažni alarmi, recimo če administrator nenadoma začne izvajati portscan.



Primer IDS/IPS sistema

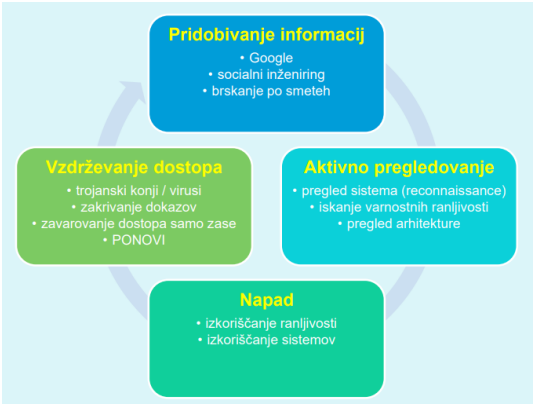
- Snort IDS
- public-domain, odprtokodni IDS za Linux, UNIX, Windows (uporablja isto knjižnico za branje omrežnega prometa kot Wireshark)
- primer vzorca napada



Napadi in grožnje

Pogosti napadi na omrežne sisteme

- **NAMEN?** Namenjeni so škodovanju ali obhodu računalniških in omrežnih funkcij.
- **ZAKAJ?** Denarna dobrobit, škodovalnost, poneverbe, ekonomske dobrobiti, čast in slava?
- **KAKO?** Ogrožanje zaupnosti, integritete in razpoložljivosti omrežnih sistemov
  - napadi s spreminjanjem informacij (modification attack)
  - zanikanje komunikacije (repudiation attack)
  - odpoved delovanja sistema (denial-of-service attack)
  - nepooblaščen dostop (access attack)



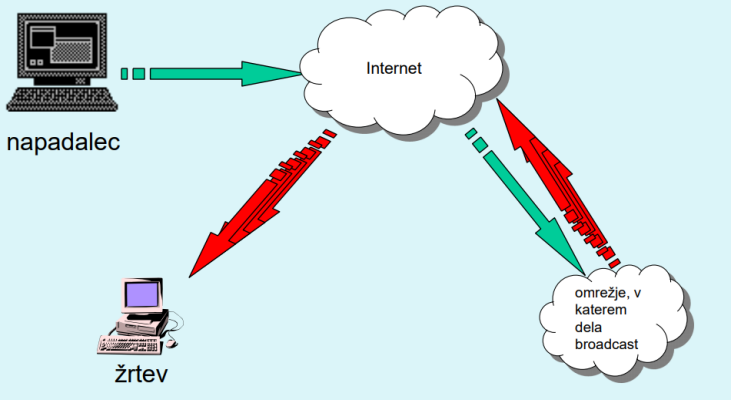
Pogosti napadi

1. **prisluškovanje in ponarejanje sporočil**
2. **matematični napadi** na kriptografske algoritme in ključe
3. **ugibanje gesel** (brute force, napad s slovarjem)
4. **virusi, črvi, trojanci**
5. **izkoriščanje šibkosti v programski opremi**
6. **socialni inženiring** (preko e-maila, telefona, servisov) - zelo učinkovito
7. **pregled vrat (port scan):** napadalec testira, kateri strežniki so delujoči (npr. ping) in katere storitve ponujajo. Napadalec lahko pridobiva podatke o sistemu: DNS, storitve, operacijski sistemi)
8. **brskanje po smeteh (dumpster diving):** način, s katerim lahko napadalci pridejo do informacij o sistemu (navodil za uporabo, seznamov gesel, telefonskih številk, opisa organizacije dela)
9. **rojstnodnevni napad (birthday attack):** je napad na zgoščevalne funkcije, za katere zahtevamo, da nobeni dve sporočili ne generirata iste zgoščene vrednosti. Pri slabših funkcijah napadalec išče sporočilo, ki bo dalo isto zgoščeno vrednost.
10. **zadnja vrata (back door):** napadalec zaobide varnostne kontrole in dostopi do sistema preko druge poti,
11. **ponarejanje IP naslovov (IP spoofing):** napadalec prepriča ciljni sistem, da je nekdo drug (poznan) sspreminjanjem paketov,
12. **prestreganje komunikacije (man-in-the-middle):** napadalec prestreže komunikacijo in se obnaša, kot da je ciljni sistem (pri uporabi certifikatov lahko žrtev uporablja tudi javni ključ napadalca)
13. **ponovitev komunikacije (replay):** napadalec prestreže in shrani stara sporočila ter jih ponovno pošlje kasneje, predstavljajoč se kot eden izmed udeležencev
14. **ugrabitev TCP sej (TCP hijacking):** napadalec prekine komunikacijo med uporabnikoma in se vrine v mesto enega od njiju; drugi verjame, da še vedno komunicira s prvim
15. **napadi s fragmentacijo (fragmentation attack):** z razbijanjem paketa na fragmente razdelimo glavo paketa med fragmente tako, da jih požarni zid ne more filtrirati
    - tiny fragment attack: deli glavo prvega paketa
    - overlapping fragment attack: napačen offset prepiše prejšnje pakete
16. **odpoved delovanja sistema (Denial-of-Service):**
    - cilj napadalca: obremeni omrežne vire tako, da se nehajo odzivati zahtevam regularnih uporabnikov(npr. vzpostavitev velikega števila povezav, zasedanje diskovnih kapacitet, ...)
    - lahko je porazdeljen (distributed DOS = DDoS)

Napadi DOS

- **prekoračitev medpomnilnika (buffer overflow):** procesu pošljemo več podatkov, kot jih lahko sprejme (Ping of death: ICMP z več kot 65K podatkov je povzročil sesutje sistema)
- **SYN napad:** napadalec pošlje veliko število zahtev za vzpostavitev povezave in se na odgovor sistema ne odzove; pride do preobremenitve vrste zahtev v sistemu
    - rešitev: omejitev števila odprtih povezav, timeout
- **napad Teardrop:** napadalec spremeni podatke o številu in dolžini fragmentov v IP paketu, kar zmede prejemnika
- **napad Smurf:** uporaba posrednega broadcasta za preobremenitev sistema
    - porazdeljen DDoS
    - uporabniki porazdeljenih omrežnih sistemov lahko da ne vejo, da je napadalna oprema nameščena pri njih

DOS Smurf



Izkorišča omrežje v katerem deluje broadcast na tak način, da ga lahko naslovimo iz nekega drugega omrežja in da gredo odgovori na ta broadcast lahko ven iz tega omrežja.

Napadalec pripravi IP paket, kjer nastavi SRC IP na IP žrtve(sem bodo leteli vsi odgovori - DDOS) DEST IP pa nastavi na broadcast naslov nekega omrežja v katerem deluje broadcast na zgoraj omenjeni način. Rezultat napda je, da vsi računalniki v omrežju začnejo odgovarjati na broadcast, odgovori letijo na žrtev. Gre za porazdeljeni DOS saj je v omrežju več računalnikov.

Uporaba bot-ov

- (web roBOR) - boti so lahko računalniki, okuženi s trojanskimi konji
- njihovi uporabniki običajno ne vejo, da sodelujejo v napadu zato jim pravimo tudi zombie računalniki
- subjekti v napadu: napadalec, centralni računalnik za krmiljenje botov (herder), boti (zombie), cilj