

Računalniške komunikacije

2020/21

kriptografija

bločna, asimetrična, principi varovanja

operativna varnost

požarni zidovi

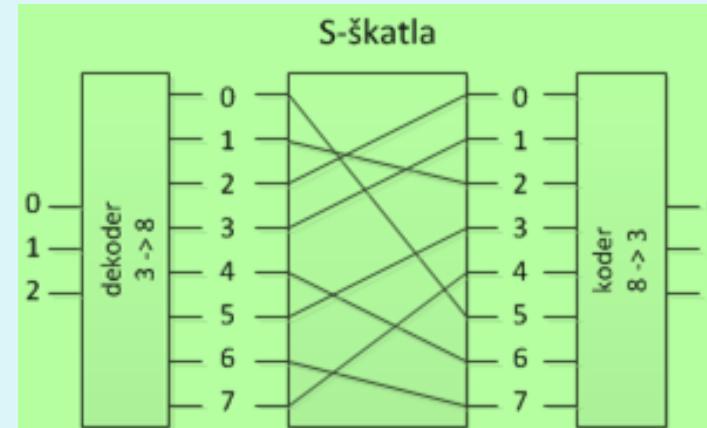
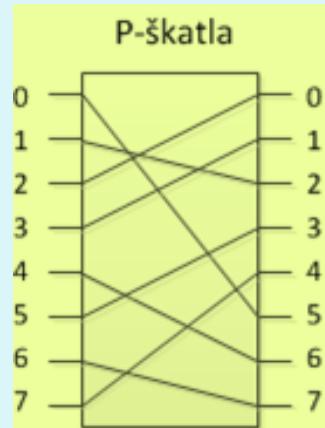
Metode kriptiranja

- 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)



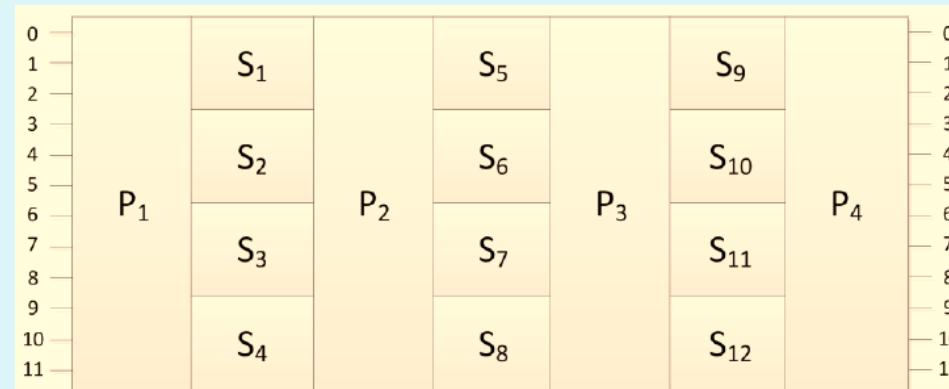
Bločna kriptografija

- permutacijska škatla (s ključem 23157046)
 - substitucijska škatla (dekoder-permutacija-koder)



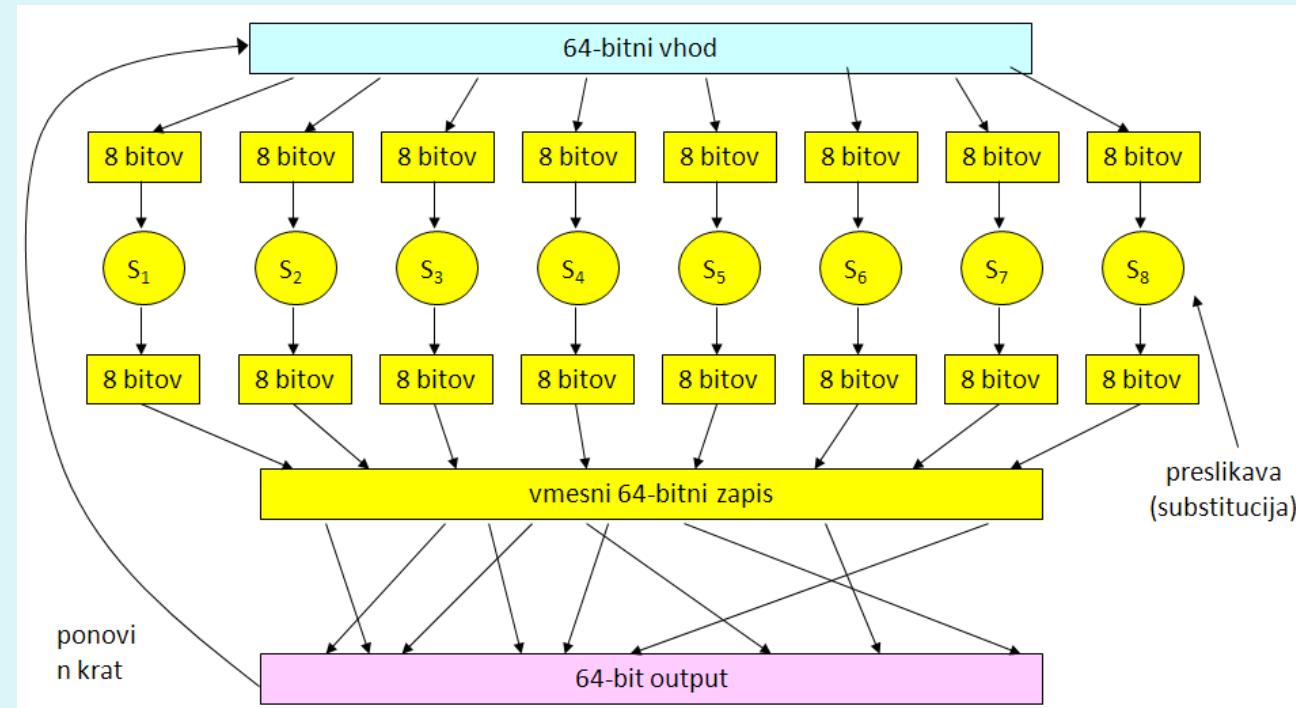
čistopis	criptogram
000	101
001	010
010	000
011	001
100	110
101	011
110	111
111	100

- možno je kombiniranje škatel v preslikovalno kaskado za poenostavitev logike



Bločna kriptografija

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

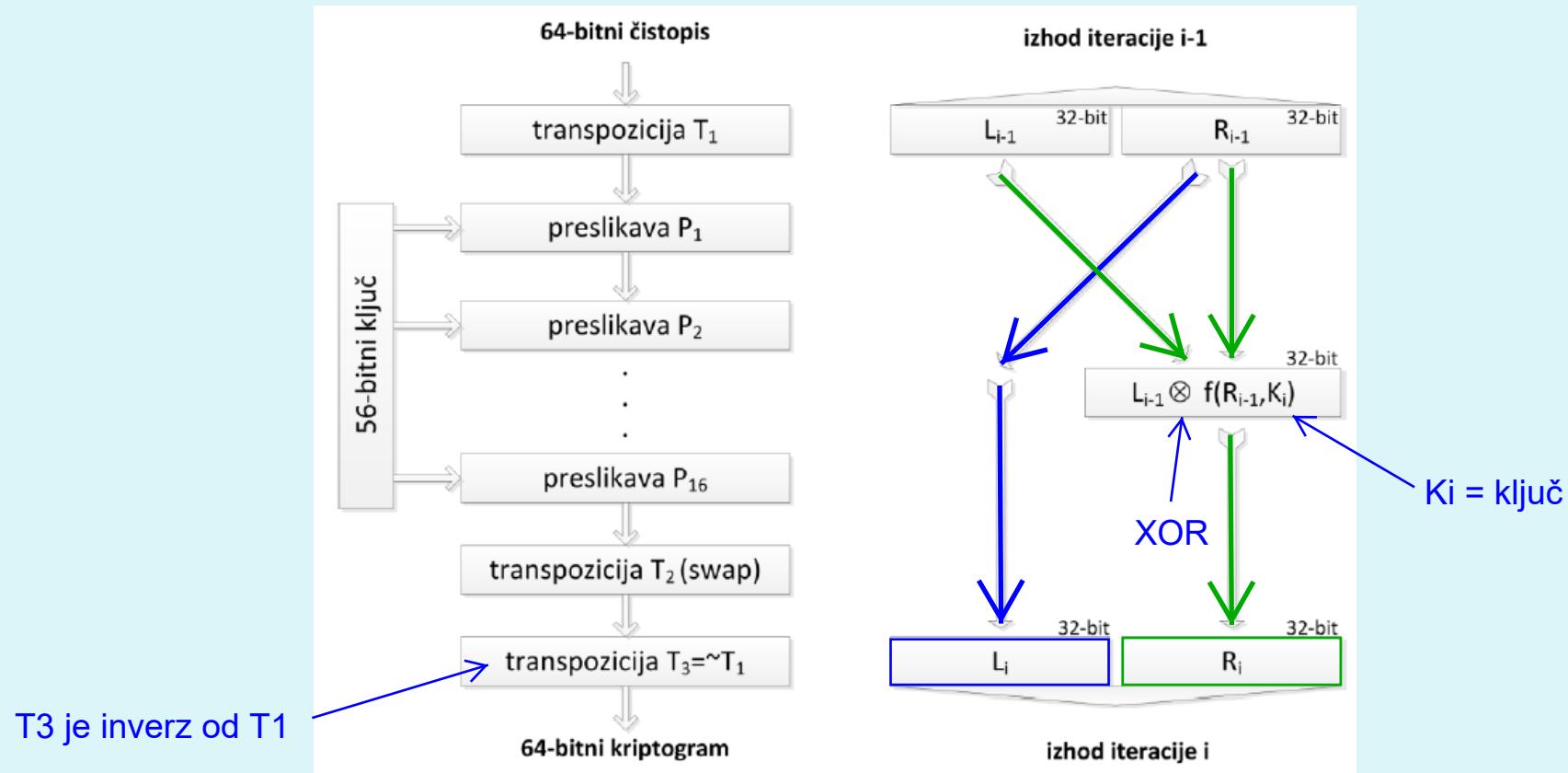


Bločna kriptografija

- **primeri algoritmov** za bločno kriptografijo
 - DES (*Data Encryption Standard*) vzame blok po 64 bit in dela na njem, nato naslednjih 64 bitov itd.
 - 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

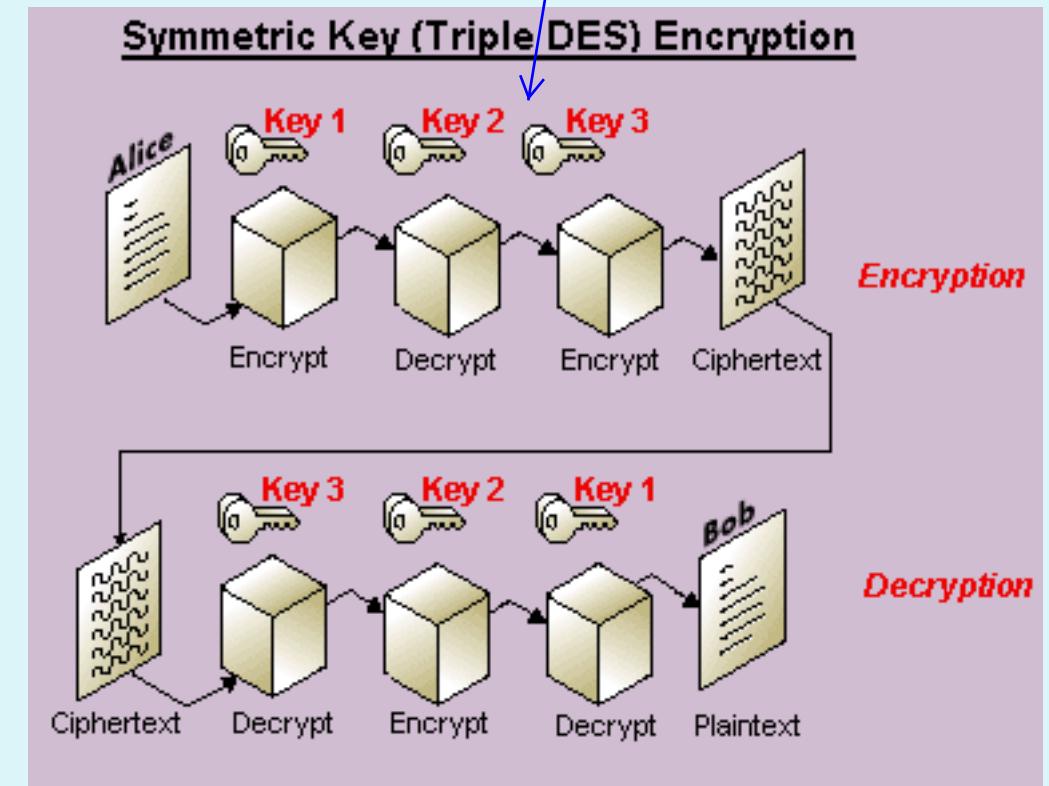


3DES (trojni DES)

2 dodatna ključa

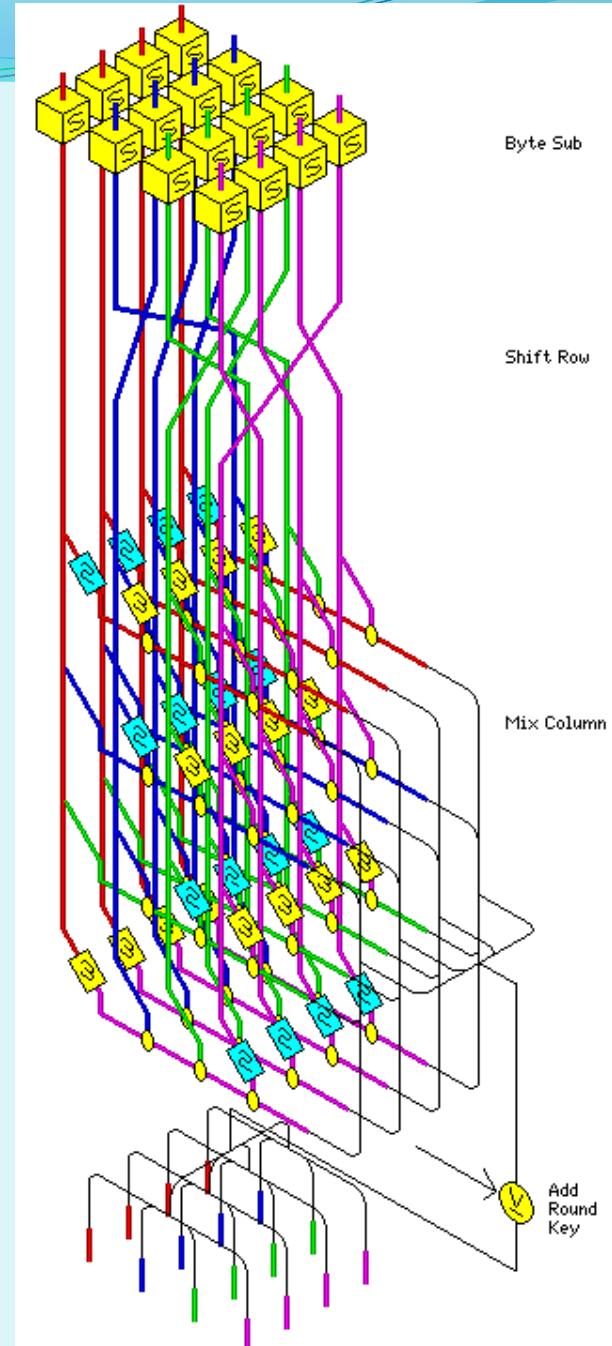
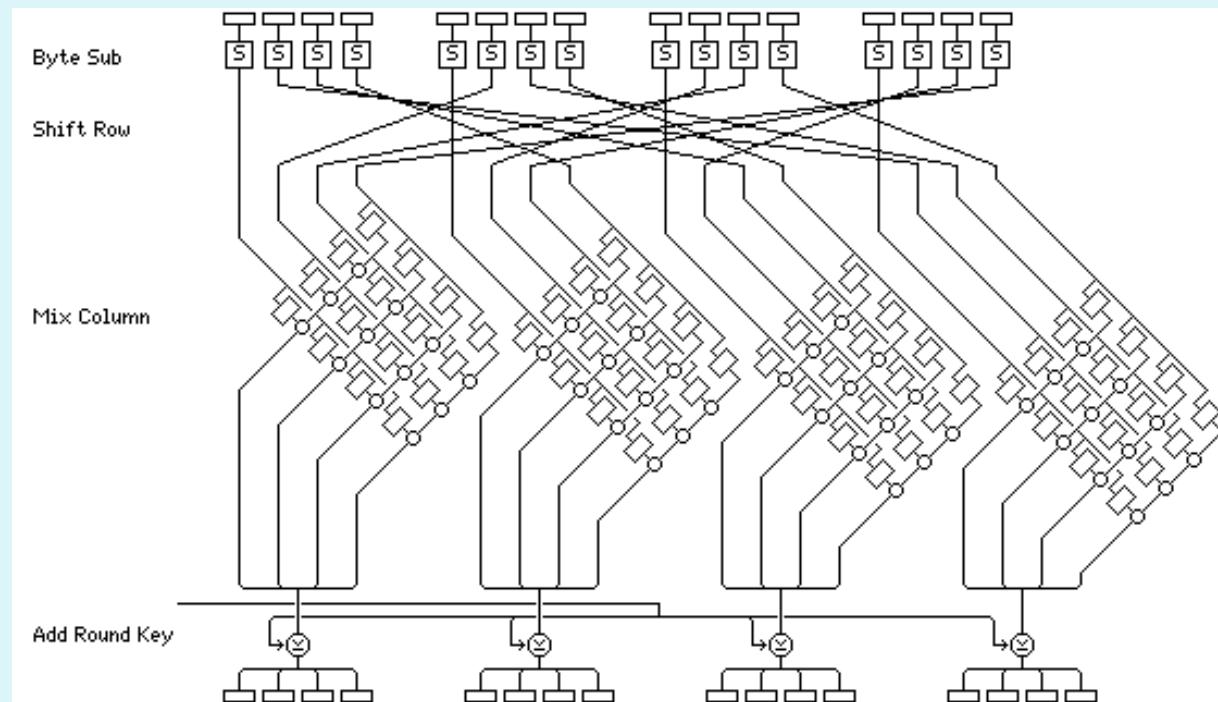
- 3 x kriptiranje (kriptiranje, dekriptiranje, kriptiranje)
- 3 x počasnejši
- $2^{2 \cdot 56}$ krat varnejši za napad z grobo silo, ker uporablja 2 dodatni kriptirani s 56-bitnim ključem
- združljivost z DES (če $K_2 == K_3$)

če sta K_2 in K_3 enaka,
se 2. in 3. faza nevtralizirata
(dekriptiramo -> kriptiramo isto stvar)



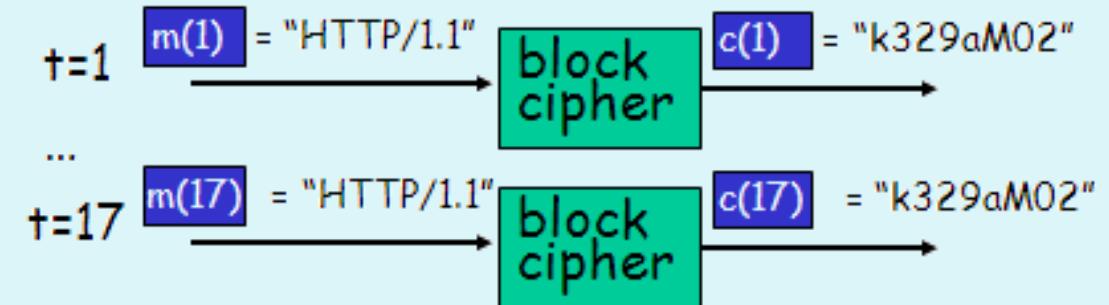
AES

- algoritem Rijndael, najbolj učinkovita in varna metoda
- bloki 128 bitov, ključ 128/192/256 bitov
- dober računalnik bi potreboval 10^{10} let za razbijanje
- različne AES operacije: zamikanje vrstic, zamenjave stolpcev, izpeljave ključev



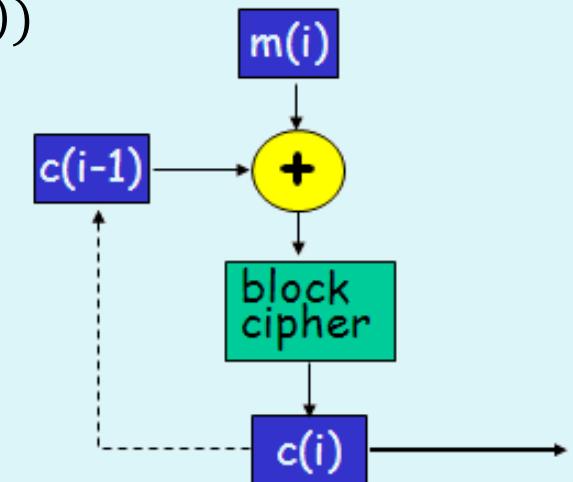
Verižno kriptiranje blokov (*Cipher Block Chaining*)

- bločna kriptografija ni praktična za pošiljanje velikih sporočil, ker dajejo ponavljači se bloki (npr. HTTP/1.1) iste kriptogramme (možna kriptoanaliza sporočila)



- REŠITEV:** verižno kriptiranje
 - pošiljatelj s prvim sporočilom pošlje neko naključno vrednost (inicializacijski vektor) $c(0)$
 - pošiljatelj kriptira vsako sporočilo v kriptogram: $c(i) = K_S(m(i) \otimes c(i - 1))$
 - prejemnik odkriptira sporočilo: $m(i) = K_S^{-1}(c(i)) \otimes c(i - 1)$
- ni potrebno, da je IV tajen
- omogoča, da se isto sporočilo kriptira v različne kriprograme!

zaradi tega potrebujemo inicializacijski vektor



Verižno kriptiranje blokov (*Cipher Block Chaining*)

- pošiljatelj kriptira: $c(i) = K_S(m(i) \otimes c(i - 1))$
- prejemnik odkriptira: $m(i) = K_S^{-1}(c(i)) \otimes c(i - 1)$

PRIMER

- $m=010010010$, $\text{IV}=c(0)=001$, $k=3$
- pošiljatelj:
 - $c(1) = K_S(m(1) \text{ XOR } c(0)) = K_S(010 \text{ XOR } 001) = K_S(011) = \textcolor{blue}{100}$
 - $c(2) = K_S(m(2) \text{ XOR } c(1)) = K_S(010 \text{ XOR } \textcolor{blue}{100}) = K_S(110) = \textcolor{red}{000}$
 - $c(3) = K_S(m(3) \text{ XOR } c(2)) = K_S(010 \text{ XOR } \textcolor{red}{000}) = K_S(010) = 101$

\underline{m}	$K_S(\underline{m})$
000	110
001	111
010	101
011	100
100	011
101	010
110	000
111	001

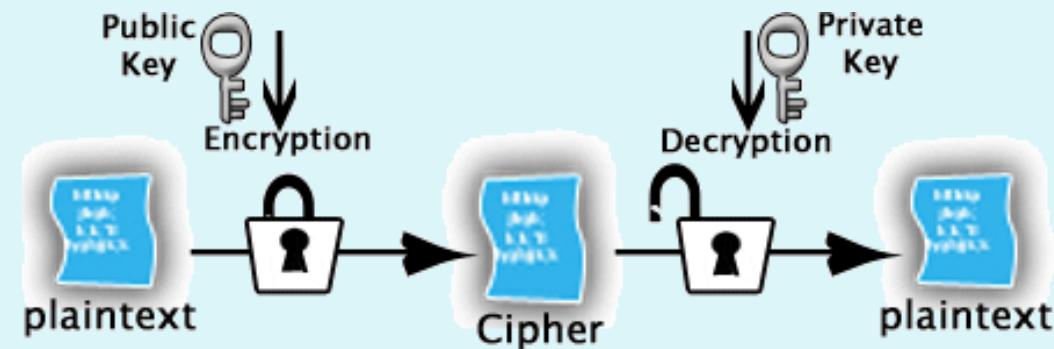
Razbijanje bločnih kriptosistemov z grobo silo

- ocena glede na stanje tehnologije v letu 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	∞	∞	∞	∞	∞

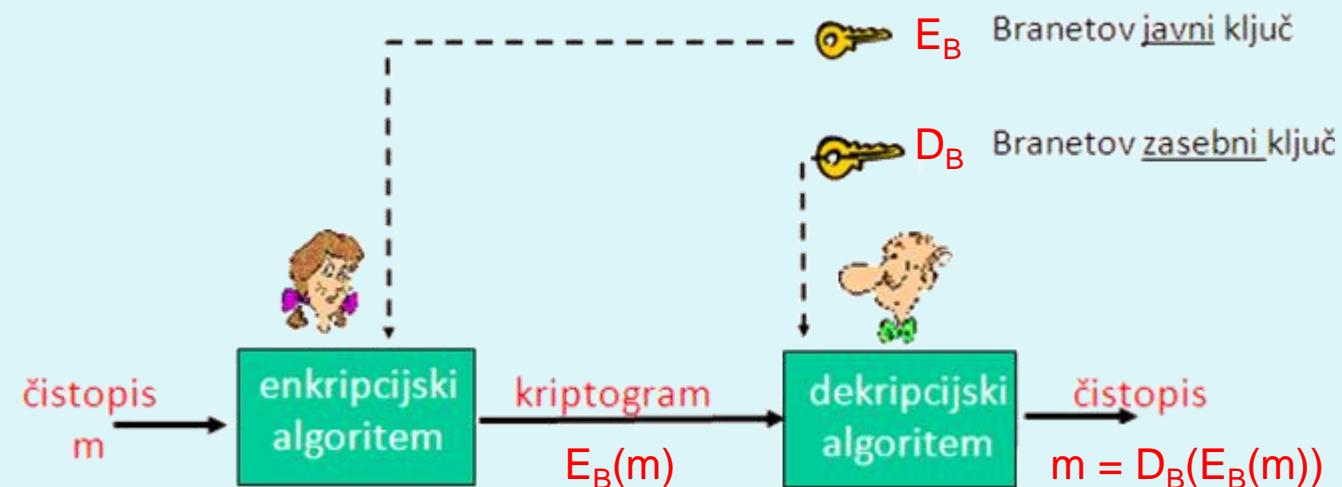


Asimetrična kriptografija



Asimetrična kriptografija

- enkripcijski (**E**) in dekripcijski (**D**) ključ sta lahko različna
- E je lahko javen, D mora biti tajen. Zahteve:
 - $D(E(m)) = m$
 - iz m in $E(m)$ je nemogoče ugotoviti D
 - iz E je nemogoče ugotoviti D
- primer: algoritmom RSA (Rivest, Shamir, Adleman)



Algoritem RSA

- izberemo p, q : veliki praštevili (1024 bitov)

$$n = pq$$

$$z = (p-1)(q-1)$$

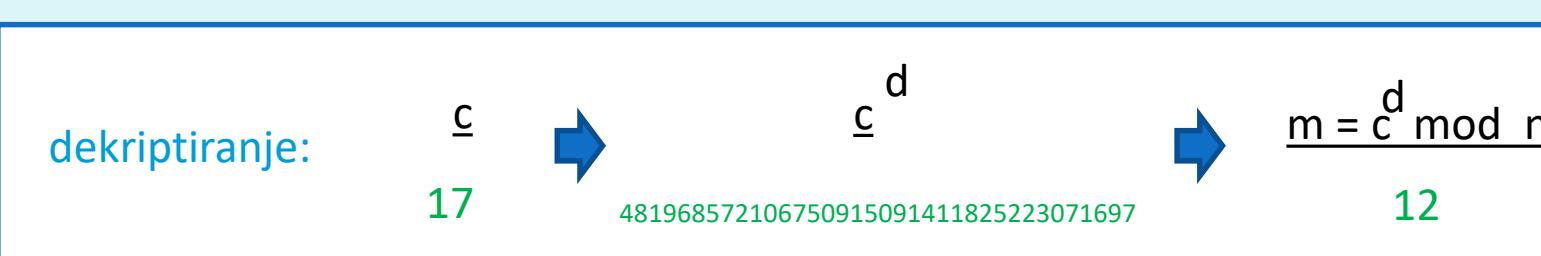
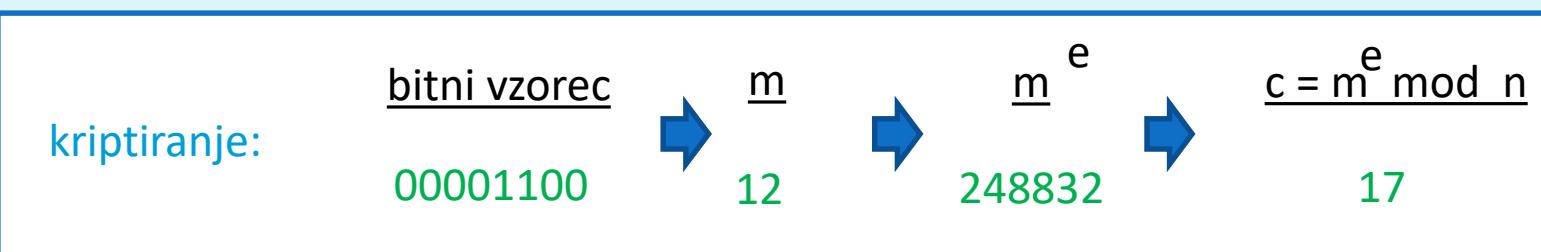
- izberemo e tako, da nima skupnih deliteljev z z.
- izberemo d tako, da $ed \bmod z = 1$
- definiramo:
 - javni ključ: $E = (n, e)$
 - zasebni ključ: $D = (n, d)$

$m \rightarrow c = m^e \bmod n$ *criptiranje*

$c \rightarrow m = c^d \bmod n$ *dekriptiranje*

RSA: primer

- Brane izbere: $p=7$, $q=5$
 - sledi $n=35$, $\varphi=24$
 - izberemo $e=5$ (e in φ sta si tuji števili) in $d=29$ (da je $e-1$ deljivo z φ)



Dejstva o RSA

- 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
- ker je RSA počasen, ga običajno uporabljam le za izmenjavo simetričnega ključa za izvedbo nadaljnje seje K_s (*session key*)

Principi varne komunikacije: avtentikacija, integriteta, elektronski podpis



Contracts Financial statements
Company strategy Human resources
documents User names and passwords
Social Security numbers Credit card details
Invoices Market research Legal advice
Sales figures Customer information

Avtentikacija udeležencev

- 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 $E_A(D_A(m)) = m$

s tem vemo, da je Ana res poslala avtentikacijo, ker samo ona pozna svoj zasebni ključ!

- avtentikacija **prejemnika**:

- sporočilo m kriptiramo v $E_B(m)$, prejemnik odkriptira v $D_B(E_B(m)) = m$

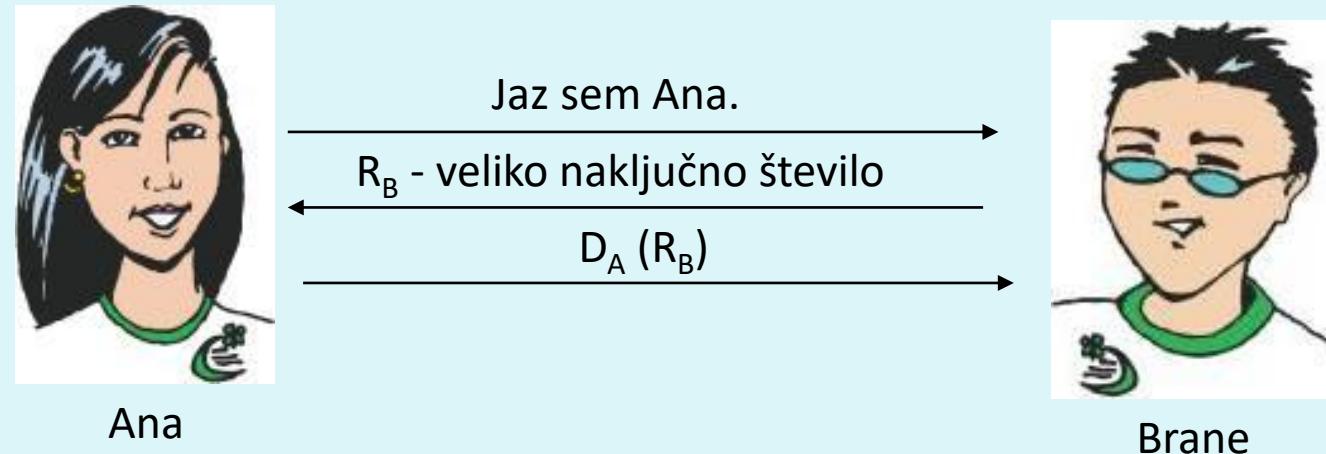
samo prejemnik lahko odkriptira sporočilo, ker samo on pozna svoj zasebni ključ!

- **vzajemna** avtentikacija:

- sporočilo m kriptiramo v $D_A(E_B(m))$, prejemnik odkriptira v $D_B(E_A(D_A(E_B(m)))) = m$

Avtentikacija pošiljatelja in prejemnika

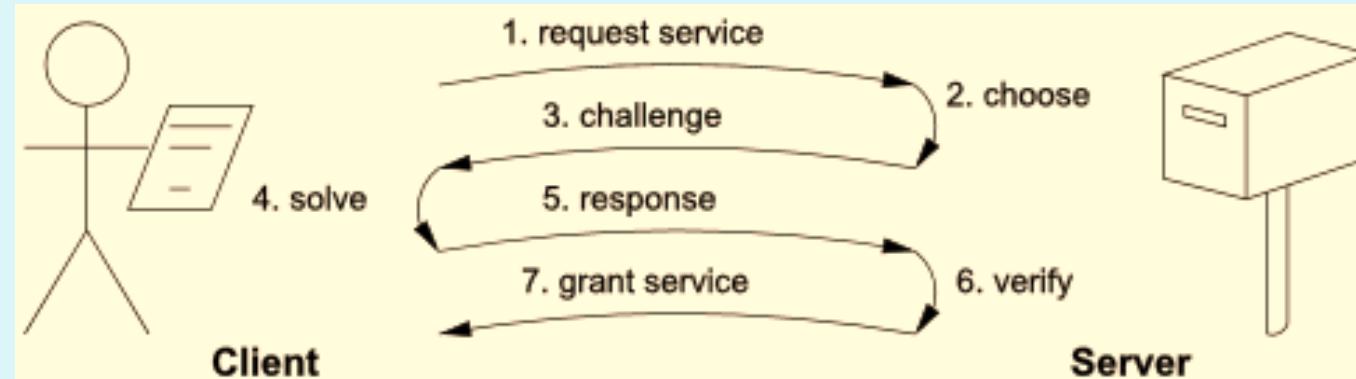
- kako preveriti, s kom zares komuniciramo?
- mehanizem izziv-odgovor (challenge-response)



- Brane dobi dokaz, da komunicira z Ano tako, ker lahko z Aninim javnim ključem odkriptira kodirano naključno število (tega je kriptirala Ana s svojim zasebnim ključem), ki ga je definiral on in ga torej pozna.
- varnostna luknja: nekdo se lahko vrine v zgornjo komunikacijo (man-in-the-middle attack) in se obnaša kot posrednik
 - REŠITEV?

Avtentikacija udeležencev

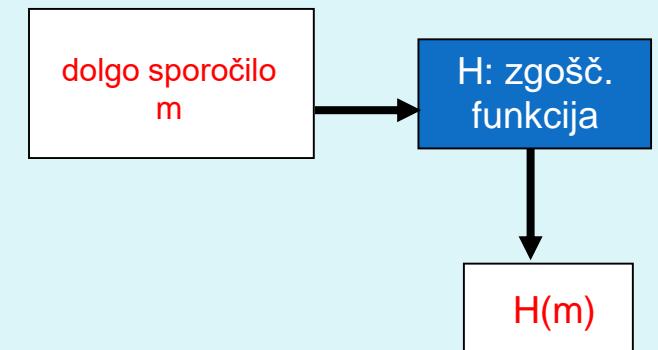
- dvostranska avtentikacija?
- učinkovitost postopka?



- drugi avtentikacijski protokoli? napadi?
 - glej učbenik, poglavje 7.6.2

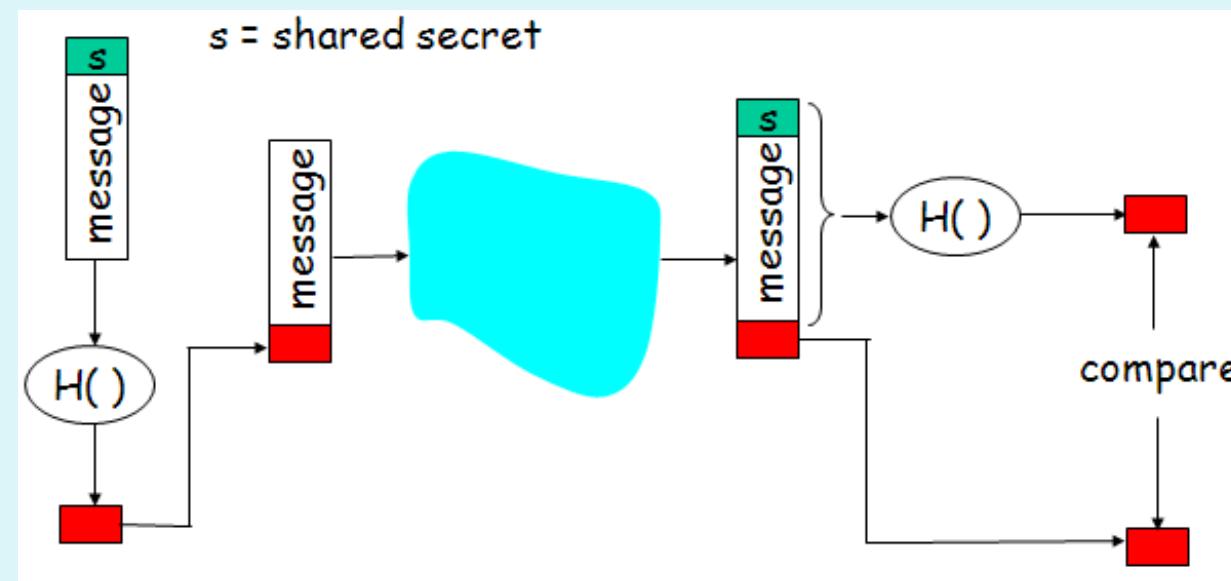
Integriteta komunikacije

- želimo imeti komunikacijsko okolje, v katerem lahko udeleženci preverijo, da:
 - sporočilo ni bilo **spremenjeno**,
 - je sporočilo "sveže" (**ni ponovljeno** posneto staro sporočilo),
- uporabljam **zgoščevalne kriptografske funkcije**: (hashing functions)
 - preprosto izračunljive,
 - iz $H(m)$ ne moremo ugotoviti m
 - težko je najti m in m' , da velja $H(m)=H(m')$
 - izgleda kot naključen niz
 - primer: internet checksum (zakaj je slab?)
 - MD5, SHA-1
 - primer: <http://hash.online-convert.com/>



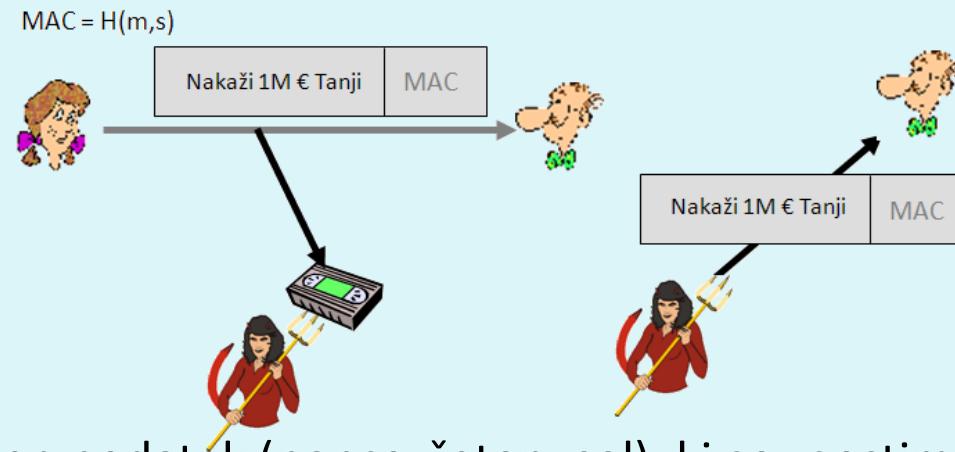
Zgoščena vrednost sporočila

- 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*)
 - problem: distribucija avtentikacijskega ključa

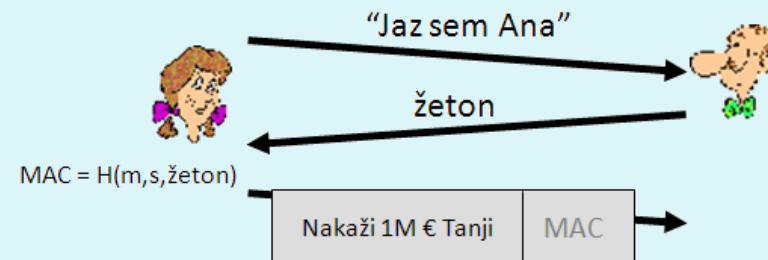


Napad s ponavljanjem komunikacije

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



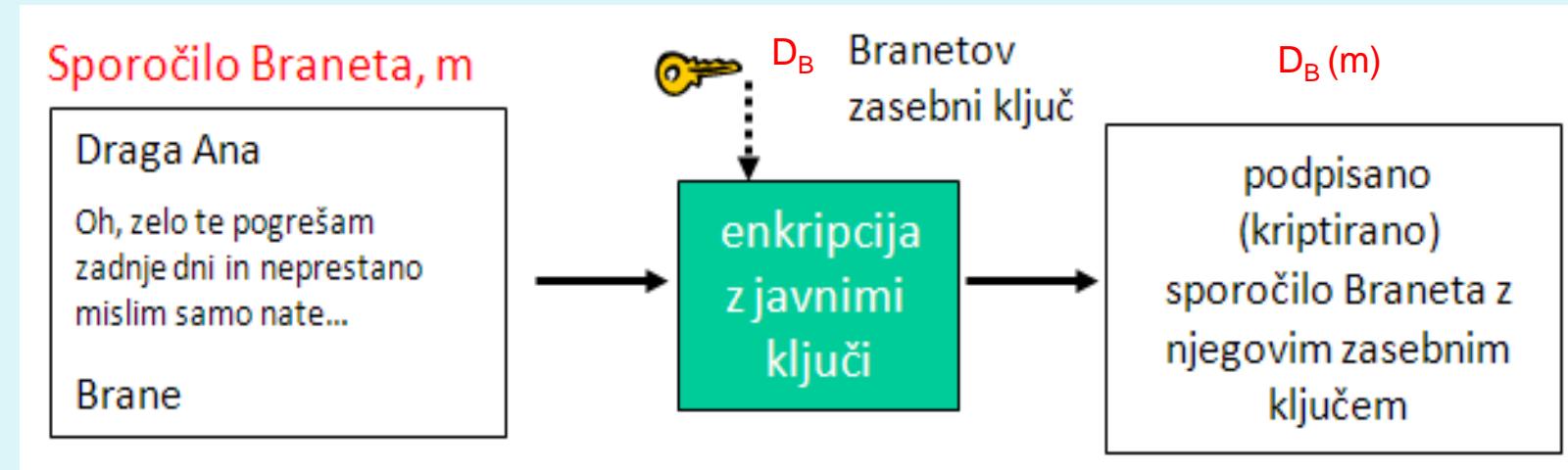
- REŠITEV: uporabimo enkraten podatek (*nonce*, žeton, sol), ki ga zgostimo skupaj s sporočilom in ključem: $H(m, s, \text{žeton})$



žeton pove, da je sporočilo že bilo poslano/prejeto, zato napadalec ne more več s shranjevanjem celotne komunikacije izvesti napada, ker ne pozna žetona (oz. zaporedne št. žetona)

Digitalni podpis

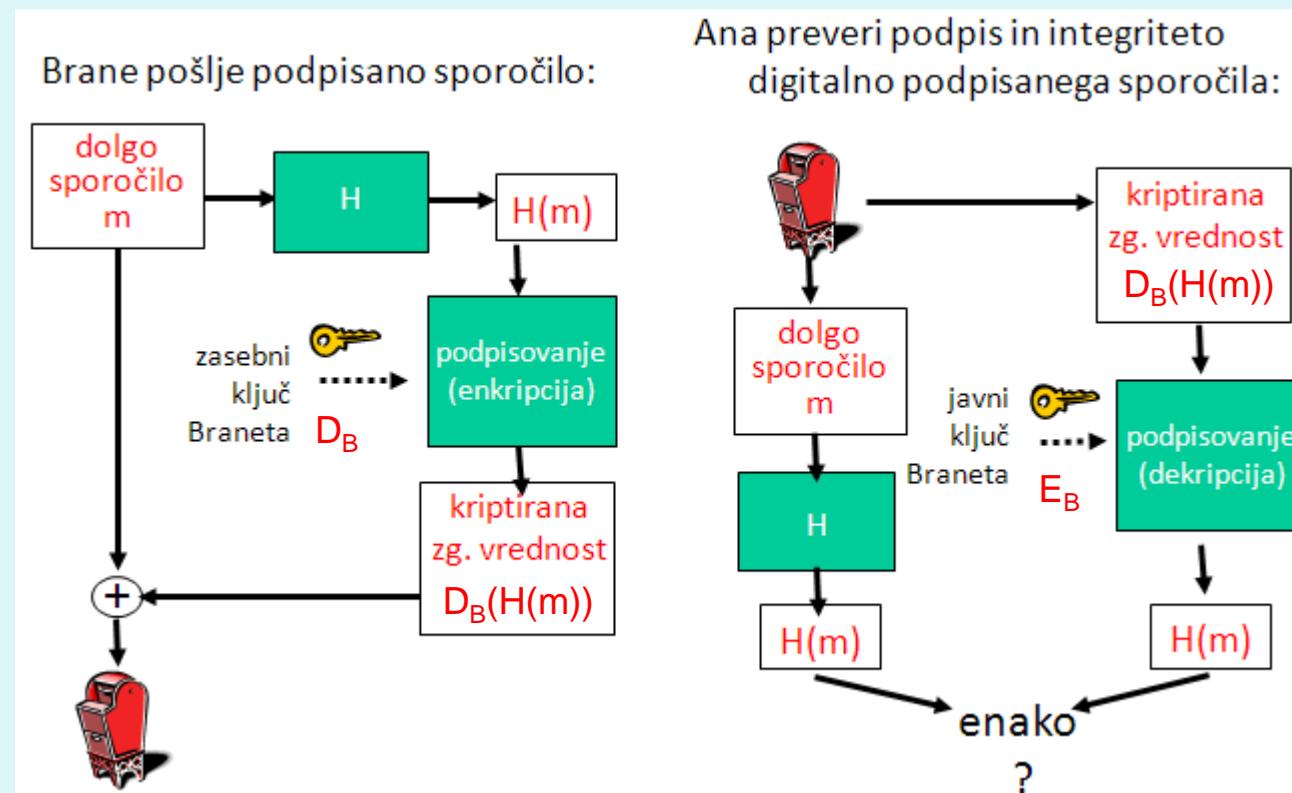
- način za zamenjavo osebnega podpisa (informacija, ki enolično potrjuje identiteto posameznika)
- avtentikacija z MAC ni enolična, uporabimo kriptografijo z javnimi ključi
- NAČIN 1:
 - pošiljatelj B lahko izračuna $D_B(m)$, kar lahko predstavlja podpisan dokument
 - prejemnik izračuna $E_B(D_B(m)) = m$
 - časovno zahtevno pri dolgih sporočilih



Digitalni podpis

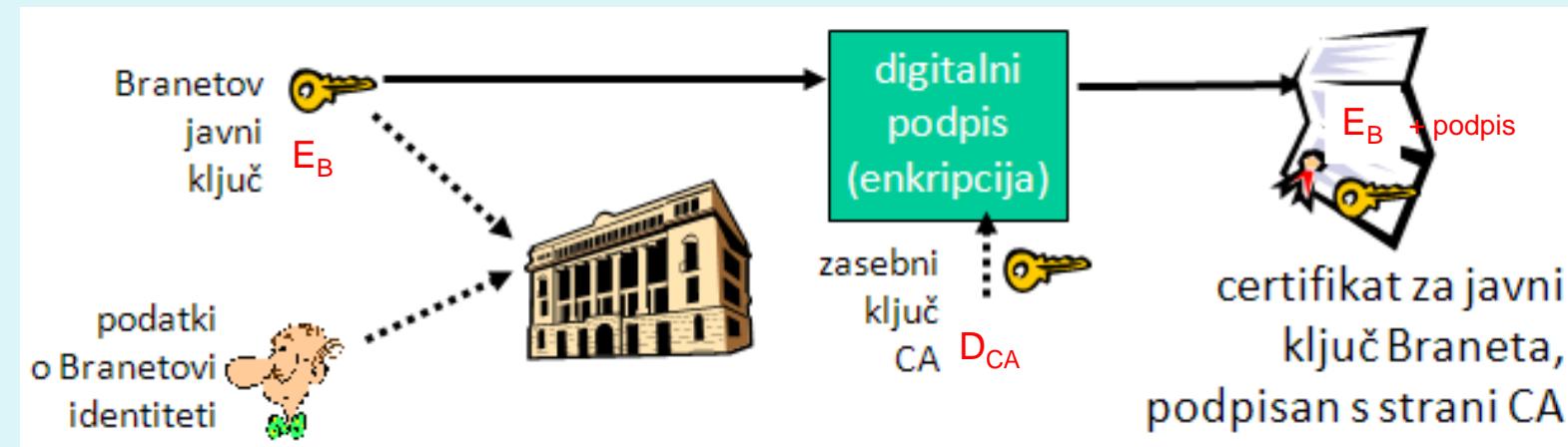
- NAČIN 2:

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



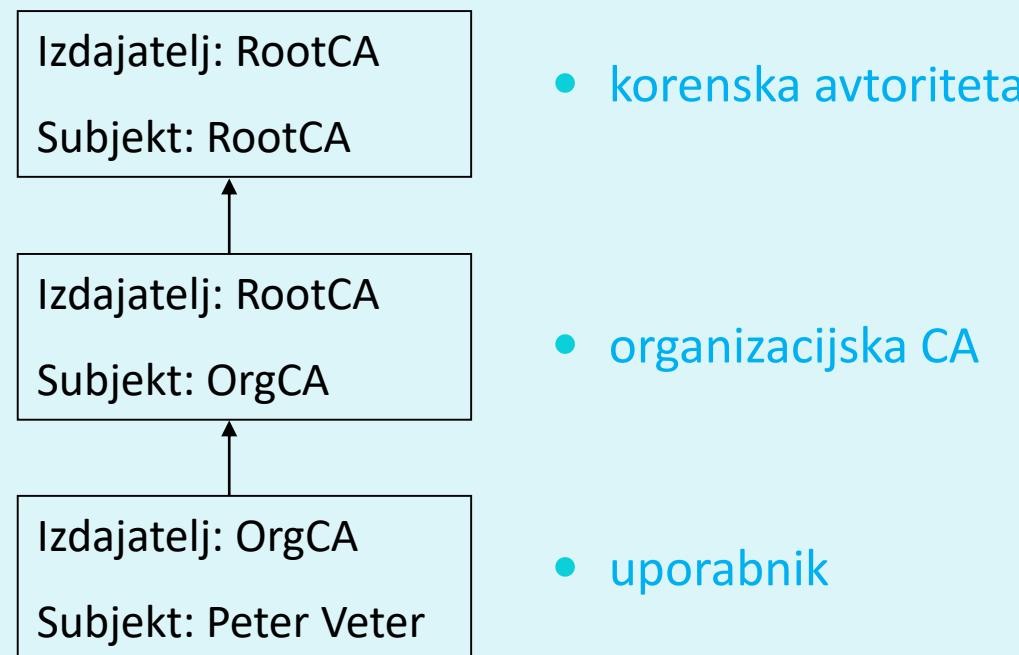
Certifikacijska agencija

- 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)



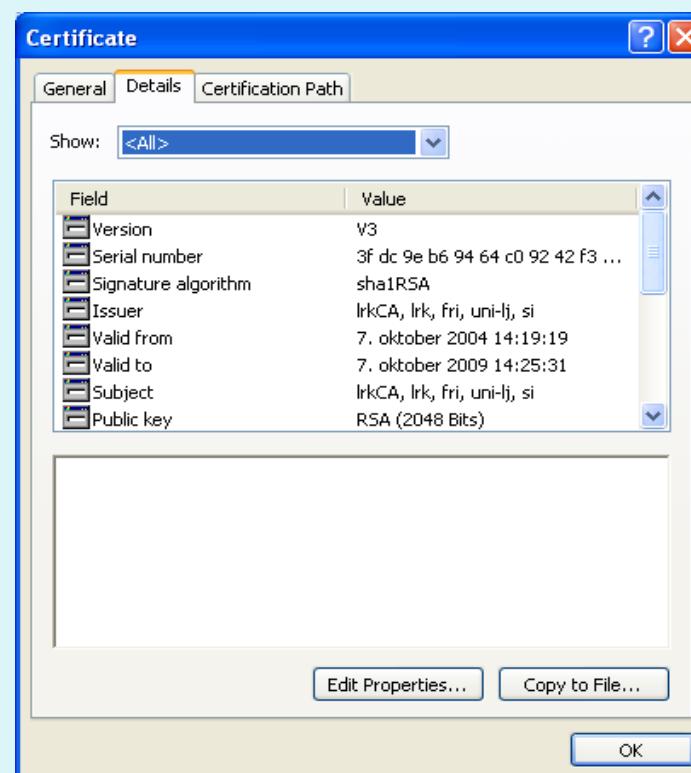
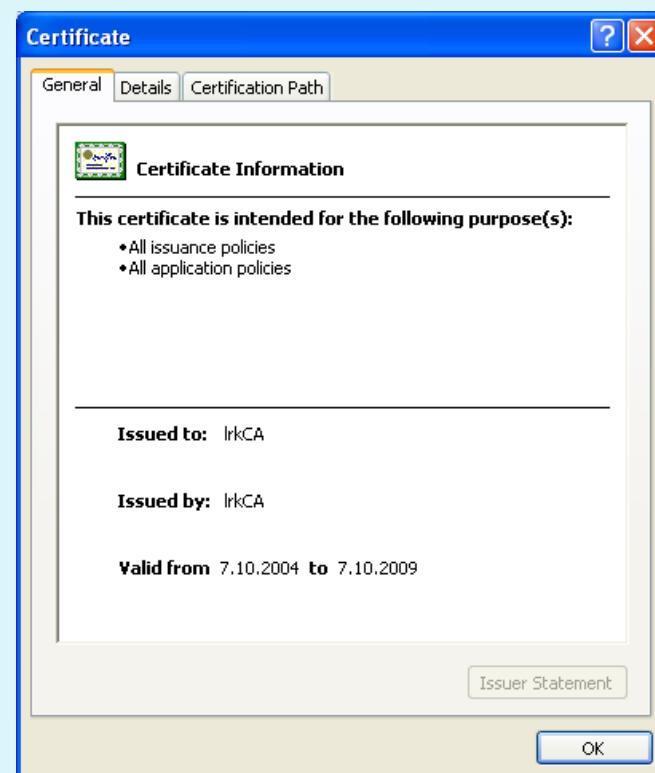
Veriga zaupanja certifikatov

- veljavnost certifikatov preverimo z verigo zaupanja (podobno kot pri DNS)
- korenske avtoritete so znane (vključene v OS itd.)



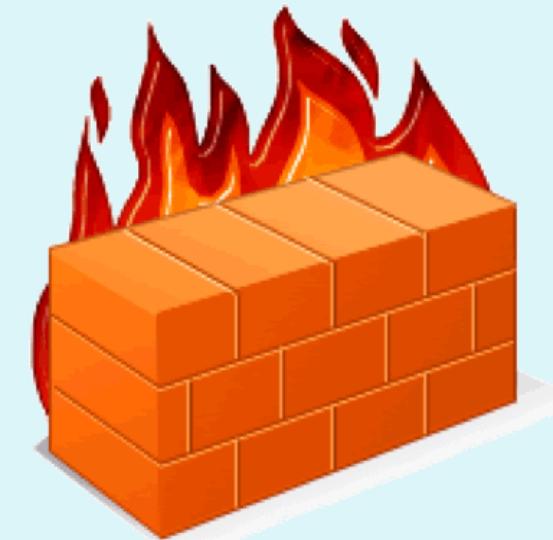
Certifikati

- 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



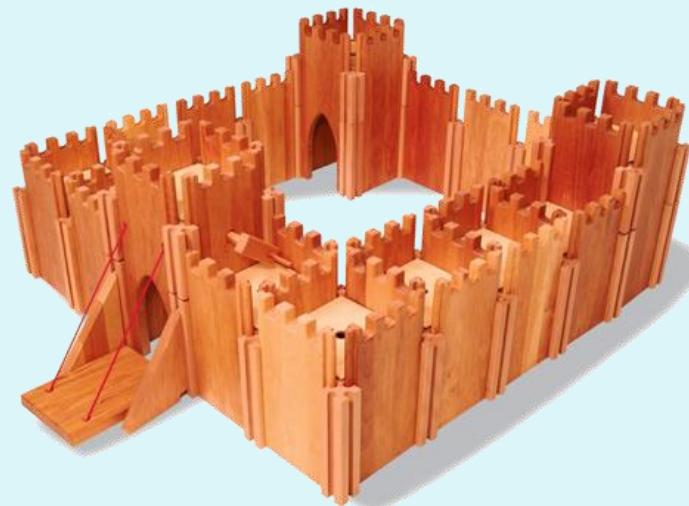
Operativna varnost:

požarni zidovi in sistemi za zaznavanje vdorov



Varnost v omrežju

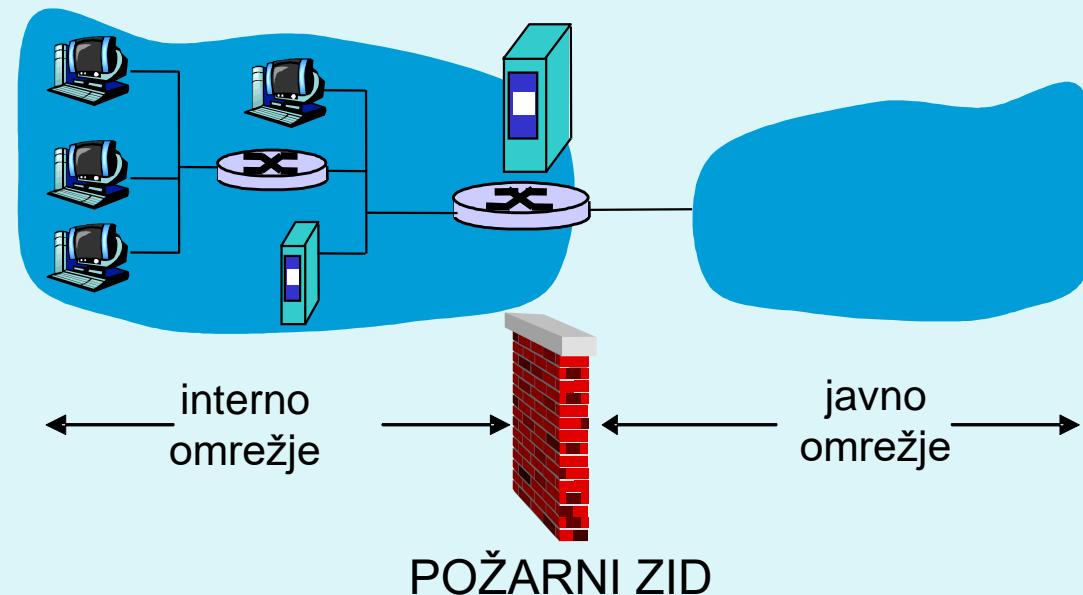
- Administrator omrežja lahko uporabnike deli na:
 - **good guys**: uporabniki, ki legitimno uporabljajo vire omrežja, pripadajo organizaciji,
 - **bad guys**: vsi ostali, njihove dostope moramo skrbno nadzorovati
- Omrežje ima običajno eno samo točko vstopa, kontroliramo dostope v njej:
 - **požarni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



Požarni zid (firewall)

izolira interno omrežje od velikega javnega omrežja, določenim paketom dovoli prehod, druge blokira. Ima 3 naloge:

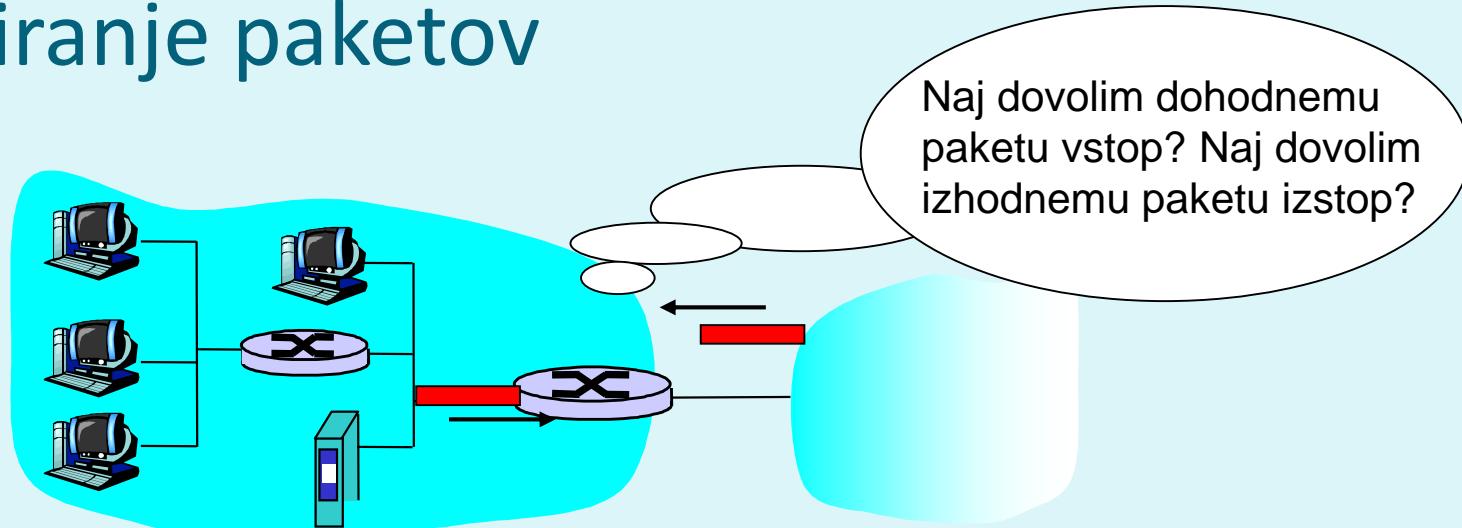
- filtrira VES promet,
- prepušča samo promet, ki je DOPUSTEN glede na politiko,
- je IMUN na napade



Požarni zid: vrste filtriranj

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

Izolirano filtriranje paketov



- filtriranje običajno izvaja že usmerjevalnik, ki meji na javno omrežje. Na podlagi vsebine paketov se odloča, 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 dopustnost vzpostavljanja povezave)

Izolirano filtriranje: dostopovni seznam

- 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
izven 222.22/16	222.22/16	TCP	80	> 1023	ACK	dovoli
222.22/16	izven 222.22/16	UDP	> 1023	53	---	dovoli
izven 222.22/16	222.22/16	UDP	53	> 1023	----	dovoli
all	all	all	all	all	all	zavrzi

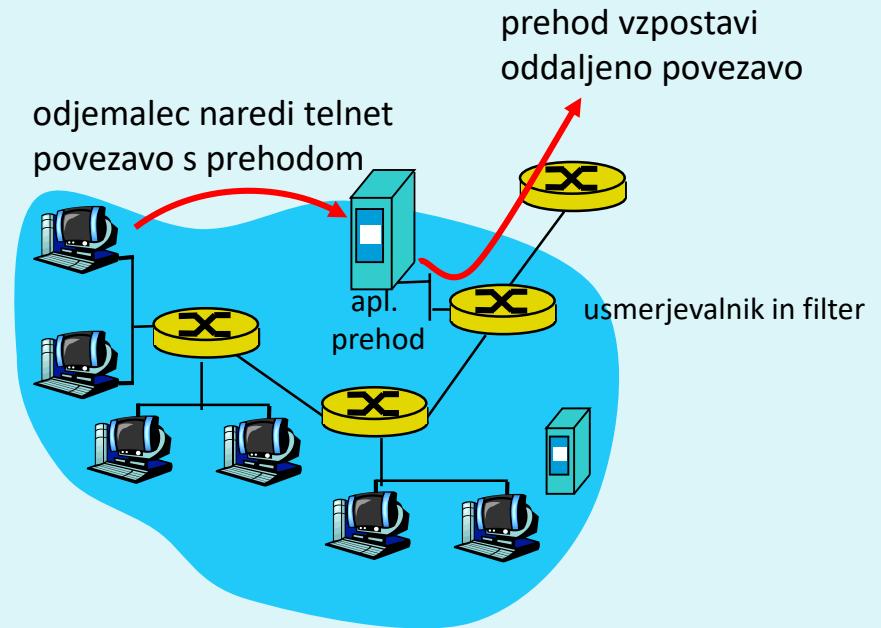
Filtriranje paketov v kontekstu

- angl. *stateful filter*, upošteva povezavo
 - izolirano filtriranje lahko dovoli vstop nesmiselnim paketom (npr. vrata = 80, ACK =1; čeprav notranji odjemalec ni vzpostavil povezave)
- IZBOLJŠAVA: **filtriranje paketov v kontekstu** spremišča 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)
 - uporabljam podoben dostopovni seznam, ki določa, kdaj je potrebno kontrolirati veljavnost povezave (angl. *check connection*)

Filtriranje paketov v kontekstu

Aplikacijski prehodi

- 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.



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

Aplikacijski prehodi

Tudi aplikacijski prehodi imajo omejitve:

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



Naslednjič gremo naprej!

- ## • operativna varnost

