Mihael Plevnik Borut Jereb

ELEKTRONSKO POSLOVANJE V LOGISTIKI

Gradivo za vaje



Univerzitetna založba
Univerze v Mariboru

Univerza v Mariboru

Fakulteta za logistiko

Elektronsko poslovanje v logistiki

Gradivo za vaje

Avtorja

Mihael Plevnik

Borut Jereb

November 2026

Naslov *Title*
Elektronsko poslovanje v logistiki
Electronic Business in Logistics

Podnaslov *Subtitle*
Gradivo za vaje
Exercise Materials

Avtorja *Authors*
Mihael Plevnik
(Univerza v Mariboru, Fakultete za logistiko; Slovenska vojska)

Borut Jereb
(Univerza v Mariboru, Fakultete za logistiko)

Recenzija *Review*
Rok Rupnik
(Univerza v Ljubljani, Fakulteta za računalništvo in informatiko)

Matevž Obrecht
(Univerza v Mariboru, Fakultete za logistiko)

Lektoriranje *Language editing*
Sonja Mlaker Kač
(Univerza v Mariboru, Fakultete za logistiko)

Tehnični urednik *Technical editor*
Jan Perša
(Univerza v Mariboru, Univerzitetna založba)

Oblikovanje ovitka *Cover designer*
Jan Perša
(Univerza v Mariboru, Univerzitetna založba)

Grafika na ovitku *Cover graphics*
Kontejnerji, foto: Pat Whelen, unsplash.com, 2020

Grafične priloge *Graphic material*
Vsi viri so lastni, če ni navedeno drugače.
Vir ločni strani vseh ločnih poglavij je lasten, pripravljeno s pomočjo orodja UI.
Plevnik, Jereb (avtorja), 2026

Založnik *Published by*
Univerza v Mariboru
Univerzitetna založba
Slomškov trg 15, 2000 Maribor, Slovenija
https://press.um.si, zalozba@um.si

Izdajatelj *Issued by*
Univerza v Mariboru
Fakulteta za logistiko
Mariborska cesta 7, 3000 Celje, Slovenija
https://fl.um.si, info.fl@um.si

Izdaja *Edition*
Prva elektronska izdaja

Vrsta publikacije *Publication type*
E-knjiga

Izdano *Published at*
Maribor, Slovenija, november 2026

Dostopno na *Available at*
https://press.um.si/index.php/ump/catalog/book/1178

© Univerza v Mariboru, Univerzitetna založba
/ University of Maribor, University of Maribor Press

Besedilo / *Text* © Plevnik, Jereb (avtorja), 2026

To delo je objavljeno pod licenco Creative Commons Priznanje avtorstva-Nekomercialno-Brez predelav 4.0 Mednarodna. / *This work is licensed under the Creative Commons Attribution-NonCommercialNoDerivs 4.0 International License.*

Uporabnikom je dovoljeno reproduciranje brez predelave avtorskega dela, distribuiranje, dajanje v najem in priobčitev javnosti samega izvirnega avtorskega dela, in sicer pod pogojem, da navedejo avtorja in da ne gre za komercialno uporabo.

Vsa gradiva tretjih oseb v tej knjigi so objavljena pod licenco Creative Commons, razen če to ni navedeno drugače. Če želite ponovno uporabiti gradivo tretjih oseb, ki ni zajeto v licenci Creative Commons, boste morali pridobiti dovoljenje neposredno od imetnika avtorskih pravic.

https://creativecommons.org/licenses/by-nc-nd/4.0/

Kataložni zapis o publikaciji (CIP) pripravili v Narodni in univerzitetni knjižnici v Ljubljani
COBISS.SI-ID 304202755
ISBN 978-961-299-216-3 (PDF)

ISBN 978-961-299-216-3 (pdf)

DOI https://doi.org/10.18690/um.fl.9.2026

Cena Brezplačni izvod

Odgovorna oseba založnika Prof. dr. Dean Korošak,
For publisher rektor Univerze v Mariboru

Citiranje Plevnik, M., Jereb, B (2026). *Elektronsko poslovanje v logistiki: gradivo za vaje.*
Attribution Univerza v Mariboru, Univerzitetna založba. doi: 10.18690/um.fl.9.2026

Univerzitetna založba
Univerze v Mariboru

Kazalo

1 Uvod 1
1.1 Namen 1
1.2 Cilji 1
1.3 Ciljna publika 2
1.4 Struktura 2

2 Uvod v elektronsko poslovanje 5
2.1 Vloga informatike v logistiki 5

3 XML (extensible markup language) 9
3.1 Kaj je XML in zakaj je pomemben? 10
3.1.1 Osnovne komponente obdelave XML dokumentov 11
3.1.2 Mini kviz: preveri svoje razumevanje 13
3.1.3 Vaja 1 – Prva XML datoteka 13
3.1.4 Povzetek 14
3.2 Osnovna struktura XML 14
3.2.1 Elementi in njihova hierarhija 14
3.2.2 Atributi in njihova vloga (metapodatki) 15
3.2.3 Odločanje med elementi in atributi (enostavna pravila) 16
3.2.4 Komentarji in posebni znaki 17
3.2.5 Vaja 2 – Seznam izdelkov 17
3.2.6 Povzetek 18
3.3 Imenski prostori (Namespaces) 18
3.3.1 Zakaj potrebujemo imenske prostore 18
3.3.2 Kako delujejo imenski prostori 19
3.3.3 Privzeti (default) imenski prostor 19
3.3.4 Pogoste napake pri uporabi imenskega prostora 20
3.3.5 Dobre prakse pri imenskem prostoru 20
3.3.6 Vaja 3 – Dodajanje imenskega prostora 20
3.4 Prikaz in validacija XML 21
3.4.1 XML kot drevo podatkov 21
3.4.2 Prikaz XML v brskalniku 22
3.4.3 Osnovna pravila za dobro oblikovan XML (ang. well-formed) 23
3.4.4 Validacija XML z uporabo shem 23
3.4.5 Uporaba Notepad++ za preverjanje in potrjevanje ustreznosti 24
3.4.6 Vaja 4 – Preverjanje XML 25
3.4.7 Povzetek 25
3.5 Povzetek in pogoste napake 25
3.5.1 Elementi in atributi 26

3.5.2 Namespace deklaracije ... 26
3.5.3 Kodiranje in posebni znaki ... 27
3.5.4 Uporaba orodij ... 27
3.5.5 Mini kviz in refleksija ... 28
3.5.6 Povzetek ... 30
3.6 Dodatno gradivo ... 30
3.6.1 XPath – osnovni izrazi za iskanje po XML drevesu ... 30
3.6.2 XSLT – preoblikovanje XML v HTML ... 31
3.6.3 RSS in XLink – primeri iz logistike ... 32
3.6.3.1 RSS (Really Simple Syndication) ... 32
3.6.3.2 XLink (XML Linking Language) ... 33
3.6.4 Reference in orodja za nadaljnje učenje ... 33
3.6.5 Povzetek ... 34
3.6.6 Mini projekt: RSS obvestila za logistično skladišče ... 34
3.7 eSLOG – standard za elektronsko poslovanje v Sloveniji ... 37
3.7.1 Razvoj standarda eSLOG ... 38
3.7.2 Vrste dokumentov eSLOG 2.0 ... 39
3.7.3 Semantični in sintaktični model ... 41
3.7.4 Procesni tok eSLOG v logistiki ... 43
3.7.5 Prikaz XML dokumenta DESADV ... 47
3.7.6 Prikaz dokumenta v prijazni obliki (XSLT) ... 47
3.7.7 Vaje ... 47

4 Šifriranje (encryption) ... 51
4.1 Uvod ... 52
4.1.1 Povezava med šifriranjem in varnostnimi cilji (zaupnost, celovitost, razpoložljivost – CIA) ... 54
4.1.2 Zgodovinski razvoj šifriranja ... 55
4.1.2.1 Antično obdobje – rojstvo šifriranja ... 55
4.1.2.2 Renesansa in predmoderno obdobje – izumi in izboljšave ... 55
4.1.2.3 19. in 20. stoletje – mehanične in vojaške šifre ... 56
4.1.2.4 Digitalno obdobje – računalniška kriptografija ... 56
4.2 Osnovni pojmi ... 57
4.3 Vrste šifriranja ... 58
4.3.1 Simetrično šifriranje ... 58
4.3.2 Asimetrično šifriranje ... 59
4.3.3 Hibridni sistemi ... 61
4.4 Najpogostejše napake pri uporabi šifriranja ... 62
4.5 Priporočila za varno uporabo ... 63
4.6 Dodatni mehanizmi za celovitost in istovetnost ... 64
4.6.1 Zgoščevalne funkcije (hash funkcije) ... 64
4.6.1.1 Primer – Izračun zgoščevalne vrednosti ... 67
4.6.1.2 Primer – Razbijanje zgoščevalne funkcije ... 68
4.6.1.3 Vaje ... 69
4.6.1.4 Povzetek ... 74
4.6.2 HMAC (Hash-based Message Authentication Code) ... 75
4.6.2.1 Vaja: Prepoznavanje prave datoteke z uporabo HMAC ... 78
4.6.2.2 Povzetek ... 79
4.6.3.1 Povzetek ... 84
4.6.4 Digitalni podpis ... 84
4.6.4.1 Demonstracija ... 89
4.6.4.2 Vaja SI-PASS Podpis ... 89
4.6.4.3 Povzetek ... 97

4.6.5 PGP (ang. Pretty Good Privacy) 98
4.6.5.1 Vaja: Uporaba PGP (Mailvelope) za varno e-pošto v Gmailu 101
4.6.5.2 Povzetek 104
4.6.6 Primerjava med PKI in PGP 105
4.6.6.1 Povzetek 106

5 Koda 109
5.1 Power Shel skripta za izračun MD5 109
5.2 Power Shel skripta za izračun SHA-1 110
5.3 Power Shel skripta za izračun SHA-256 110
5.4 Power Shel skripta za izračun HMAC 111
5.5 Power Shel skripta za preverjanje HMAC-SHA256 Hash 111

6 Uporaba orodij 115
6.1 Notepad ++ (https://notepad-plus-plus.org/) 116
6.2 Hash Tool 1.2 (https://hash-tool1.software.informer.com/) 117
6.3 IT – TOOLS (https://it-tools.tech/) 118
6.4 CrackStation (https://crackstation.net/) 119
6.5 Mailvelope (https://mailvelope.com/en) 120
6.6 VeraCrypt (https://veracrypt.jp/en/Home.html) 121
6.7 7-Zip (https://www.7-zip.org/) 122
6.8 Total Commander (https://www.ghisler.com/) 123

Viri in literatura 127

Elektronsko poslovanje v logistiki

Gradivo za vaje

ELEKTRONSKO POSLOVANJE
Digitalno. Povezano. Učinkovito.

01
UVOD
Namen, cilji, ciljna publika in struktura učbenika.

02
UVOD V ELEKTRONSKO POSLOVANJE
Vloga informatike, digitalne tehnologije, integracija in varnost.

03
XML
Struktura XML, imenski prostori in validacija.

04
ŠIFRIRANJE
Kriptografija, digitalni podpisi, PKI in varna uporaba.

05
KODA
Sintaksa, logika, struktura, napake in najboljše prakse.

06
ORODJA
Uredniki kode, validatorji, ogledovalniki, baze podatkov in testna okolja.



Univerzitetna založba
Univerze v Mariboru

1 Uvod

Digitalizacija poslovnih procesov in vse večja odvisnost logistike od informacijsko-komunikacijskih tehnologij zahtevata, da bodoči strokovnjaki temeljito razumejo elektronsko poslovanje ter z njim povezane tehnologije in varnostne mehanizme. Namen tega učbenika je zagotoviti praktično in sistematično podporo pri izvajanju vaj pri predmetu *Elektronsko poslovanje v logistiki* ter omogočiti študentom postopno pridobivanje znanja in spretnosti, ki jih bodo potrebovali v sodobnem logističnem okolju.

1.1 Namen

Učbenik je zasnovan kot kombinacija **teoretičnega uvoda** in **praktičnih vaj**, s katerimi študenti pridobijo uporabno znanje s področij elektronskega poslovanja, digitalne varnosti, standardov podatkovnih formatov (npr. XML) ter osnov kriptografskih mehanizmov, ki omogočajo varno izmenjavo informacij. Namenjena je podpori pedagoškemu procesu pri vajah in samostojnemu učenju, hkrati pa lahko služi tudi kot referenčni priročnik pri kasnejšem strokovnem delu.

1.2 Cilji

Z uporabo tega učbenika študenti:

- spoznajo temeljne pojme in principe elektronskega poslovanja v logistiki;
- pridobijo praktično znanje o uporabi standardiziranih podatkovnih struktur (npr. XML, XPath);
- razumejo pomen digitalne varnosti in uporabe kriptografskih mehanizmov pri elektronskem poslovanju;

- znajo samostojno izvajati osnovne postopke šifriranja, izračuna zgoščevalnih vrednosti in digitalnega podpisovanja;
- znajo uporabljati izbrana programska orodja za preverjanje in zaščito podatkov;
- povezujejo pridobljeno znanje z realnimi logističnimi procesi in informacijskimi sistemi.

1.3 Ciljna publika

Učbenik je namenjen predvsem **študentom 1. stopnje** na Fakulteti za logistiko, pri predmetih, ki obravnavajo elektronsko poslovanje, informacijske sisteme in kibernetsko varnost v logističnem okolju. Primerna je tudi za udeležence strokovnih usposabljanj v organizacijah, kjer se srečujejo z digitalizacijo logističnih procesov in potrebo po varni izmenjavi podatkov.

1.4 Struktura

Učbenik je razdeljen v več vsebinskih sklopov, ki sledijo logičnemu zaporedju učenja in postopnemu nadgrajevanju znanja:

1. **Uvod v elektronsko poslovanje** – poglavje predstavi temeljne pojme in pomen digitalnih tehnologij v logistiki. Obravnava vlogo informacijskih sistemov pri podpori logističnih procesov, digitalno preobrazbo in ključne koncepte elektronskega poslovanja. Študenti spoznajo, kako elektronski tokovi podatkov povezujejo različne dele oskrbovalne verige ter kakšne prednosti prinaša standardizirana in avtomatizirana obdelava informacij.
2. **XML in standardizacija podatkov** – ta sklop uvaja pomen standardiziranih podatkovnih formatov kot temelj učinkovitega elektronskega poslovanja in digitalne logistike, ki temelji na elektronskih dokumentih. Predstavljeni so osnovni gradniki XML (elementi, atributi, hierarhična struktura), pomen pravilne oblikovanosti dokumentov ter uporaba imenskih prostorov za preprečevanje konfliktov med različnimi podatkovnimi slovarji. Sledi razlaga postopkov validacije XML dokumentov z XSD shemami in prikaz naprednejših tehnologij, kot sta XPath za iskanje podatkov in XSLT za pretvorbo XML v pregledne formate (npr. HTML). Predstavljeni so tudi primeri uporabe XML v realnih logističnih scenarijih, kot so naročila, skladiščni podatki in distribucija informacij prek RSS virov. Pomemben del sklopa predstavlja uporaba XML v okviru slovenskega standarda eSLOG, ki omogoča standardizirano elektronsko izmenjavo naročil, potrditev naročil, dobavnic in računov. Poseben poudarek je na povezavi med semantičnim in sintaktičnim modelom ter procesnemu

toku dokumentov v logistiki. S praktičnimi vajami študenti ustvarijo in validirajo lastne XML datoteke, uporabijo imenske prostore, XPath in XSLT, pripravijo RSS vire ter analizirajo eSLOG dokumente (npr. DESADV). Sklop spodbuja razumevanje povezave med strukturiranimi podatki, interoperabilnostjo informacijskih sistemov in avtomatizacijo logističnih procesov.

3. **Šifriranje in povezani varnostni mehanizmi** – poglavje postopno uvaja temeljne koncepte kriptografije in njihove praktične uporabe. Začne se z razlago osnovnih pojmov in zgodovinskim pregledom razvoja šifriranja, nato predstavi različne pristope (simetrično, asimetrično in hibridno šifriranje) ter pojasni, kje se posamezne metode uporabljajo v sodobnih informacijskih sistemih. Sledijo vsebine, ki opozarjajo na najpogostejše napake pri uporabi šifriranja in priporočila za varno uporabo algoritmov in ključev. V nadaljevanju so predstavljene zgoščevalne funkcije, ki omogočajo preverjanje celovitosti podatkov, ter mehanizmi HMAC za preverjanje pristnosti sporočil. Poseben poudarek je namenjen digitalnim podpisom in infrastrukturi javnih ključev (PKI), ki omogočajo preverjanje istovetnosti in zaupanja v digitalnem okolju. Poglavje se zaključi s predstavitvijo uporabe PGP za zaščito elektronske pošte in datotek v praksi.
4. **Koda** – v tem sklopu so predstavljeni preprosti, vendar uporabni primeri skript, ki podpirajo razumevanje kriptografskih postopkov v praksi. S poudarkom na PowerShell okolju študenti spoznajo ukaze za izračun zgoščevalnih vrednosti (MD5, SHA-1, SHA-256), uporabo HMAC in preverjanje integritete datotek. Skripte so kratke, pregledne in prilagojene učnemu okolju, kar omogoča samostojno eksperimentiranje ter krepitev razumevanja osnovnih varnostnih mehanizmov brez potrebe po zapleteni programski opremi.
5. **Uporaba orodij** – ta del združuje teoretično znanje z uporabo praktičnih, brezplačno dostopnih orodij, ki se pogosto uporabljajo v logističnih in IT okoljih. Predstavljena so orodja za delo z XML (npr. Notepad++ z vtičniki), preverjanje zgoščevalnih vrednosti, šifriranje datotek in elektronske pošte (npr. 7-Zip, Mailvelope, VeraCrypt, Gpg4win) ter pregledovanje vsebine datotek in map (npr. Total Commander). Namen sklopa je razviti praktične spretnosti, ki jih študenti lahko takoj prenesejo v realne delovne procese.

Vsako poglavje se začne s kratkim teoretičnim uvodom, ki mu sledijo praktični primeri, vaje ali mini projekti. Poglavja se zaključijo s povzetki, opozorili na pogoste napake in refleksijskimi vprašanji za utrjevanje znanja. Takšna struktura omogoča postopno učenje, spodbuja samostojno delo ter povezuje teoretične koncepte z realnimi primeri iz logističnega okolja.

Elektronsko poslovanje v logistiki

Gradivo za vaje

UVOD V ELEKTRONSKO POSLOVANJE

Tehnologija povezuje.
Informacije ustvarjajo vrednost.

VLOGA INFORMATIKE
Omogoča zbiranje, obdelavo, izmenjavo in uporabo informacij v logistiki.

DIGITALNE TEHNOLOGIJE
Cloud, IoT, mobilne rešitve, AI in analitika kot gonilo sprememb.

INTEGRACIJA
Povezovanje partnerjev, sistemov in procesov v celovite verige.

INFORMACIJSKA VARNOST
Zaupnost, celovitost in razpoložljivost podatkov ter sistemov.



PREDNOSTI ELEKTRONSKEGA POSLOVANJA

- Večja učinkovitost in hitrost.
- Nižji stroški poslovanja.
- Boljše odločitve na osnovi podatkov.
- Boljše sodelovanje z partnerji.
- Večja varnost in zanesljivost.



Univerzitetna založba
Univerze v Mariboru

2 Uvod v elektronsko poslovanje

Elektronsko poslovanje predstavlja temelj sodobne logistike, saj omogoča učinkovito povezovanje informacijskih tokov med partnerji v oskrbovalnih verigah. Logistični procesi ne vključujejo zgolj fizičnega premika blaga, temveč tudi natančno, pravočasno in varno izmenjavo podatkov, ki so potrebni za načrtovanje, spremljanje in upravljanje teh tokov. Digitalne tehnologije zato postajajo enakovreden del logistične infrastrukture.

V eni izmed definicij logistike je zapisano, da logistika zajema »fizični tok materiala in tok informacij od dobavitelja naprej« — kar pomeni, da je upravljanje informacij enako pomembno kot upravljanje materiala. Informatika v tem kontekstu deluje kot **osnovna infrastruktura logistike**, saj zagotavlja orodja, postopke in sisteme, ki omogočajo delovanje logističnih procesov. Brez ustrezne informacijske podpore je danes skoraj nemogoče izvajati konkurenčne in zanesljive logistične storitve.

2.1 Vloga informatike v logistiki

Informatika se ukvarja z zbiranjem, obdelavo, shranjevanjem, varovanjem in izmenjavo informacij. V logistiki ima dvojno vlogo:

- **povezovalno**, saj omogoča usklajevanje procesov med različnimi deležniki v oskrbovalni verigi, ter
- **odločevalno**, saj zagotavlja podatke, ki so podlaga za načrtovanje in optimizacijo transportnih, skladiščnih in drugih procesov.

Za učinkovito izvajanje logističnih storitev morajo biti podatki:

- **razpoložljivi,**
- **celoviti in**
- **zaupni.**

v takšnem obsegu, da je zagotovljeno pravo blago, v pravi količini in kakovosti, na pravem mestu, ob pravem času, po najnižjih stroških in z najmanjšim vplivom na okolje.

2.2 Digitalne tehnologije kot gonilo sprememb

Razmah informacijske tehnologije (IT) je temeljito spremenil način izvajanja logističnih dejavnosti. IT danes vključuje širok nabor orodij — od računalnikov in komunikacijskih omrežij do specializiranih programskih rešitev in oblačnih platform. V logistiki so najpogosteje uporabljene tehnologije:

- sistemi za upravljanje transporta (TMS);
- sistemi za upravljanje skladišč (WMS);
- elektronska izmenjava podatkov (EDI);
- spletne in mobilne aplikacije za sledenje pošiljk;
- tehnologije veriženja blokov (blockchain) in interneta stvari (IoT).

Njihova skupna naloga je omogočiti natančen, hiter in varen pretok informacij skozi celotno oskrbovalno verigo.

2.3 Elektronsko poslovanje kot integracijski okvir

Elektronsko poslovanje združuje različne informacijske tehnologije v celovit okvir za izvajanje poslovnih procesov prek elektronskih kanalov. V logistiki omogoča:

- avtomatizirano izmenjavo dokumentov (naročilnic, računov, odpremnic);
- skrajšanje časa obdelave naročil in dobav;
- zmanjšanje napak zaradi ročnega vnosa podatkov;
- večjo transparentnost in sledljivost pošiljk v realnem času;
- boljše sodelovanje med partnerji v verigi.

Učinkovita uvedba elektronskega poslovanja zahteva standardizirane podatkovne formate, zanesljivo informacijsko infrastrukturo in ustrezne varnostne mehanizme.

2.4 Informacijska varnost kot del elektronskega poslovanja

Ker se elektronsko poslovanje izvaja prek digitalnih kanalov, je izpostavljeno različnim tveganjem – od nepooblaščenega dostopa do napadov na informacijske sisteme. Zagotavljanje varnosti informacij zato ni dodatna možnost, temveč nujen sestavni del elektronskega poslovanja.

Osnovna načela informacijske varnosti v logistiki so:

- **razpoložljivost** informacij in sistemov;
- **celovitost** podatkov;
- **zaupnost** poslovno občutljivih informacij;
- **sledljivost in odgovornost** vseh aktivnosti v sistemu.

Sistemi za upravljanje informacijske varnosti temeljijo na kombinaciji tehničnih ukrepov (npr. šifriranja, avtentikacije) in organizacijskih pravil (npr. politik dostopa, postopkov odzivanja na incidente).

Elektronsko poslovanje v logistiki

Gradivo za vaje

XML

eXtensible Markup Language

- Strukturiran format
- Razširljiv po potrebi
- Neodvisen od platforme
- Podpora za integracijo




```xml
<prodaja>
  <naročilo id="12345">
    <kupec>
      <ime>Logistika d.o.o.</ime>
      <naslov>Cesta 1</naslov>
      <pošta>3000 Celje</pošta>
    </kupec>
    <izdelek id="P001">
      <naziv>Transportna storitev</naziv>
      <količina>1</količina>
      <cena>250.00</cena>
    </izdelek>
  </naročilo>
</prodaja>
```



Univerzitetna založba
Univerze v Mariboru

3 XML (extensible markup language)

V poglavju je predstavljen jezik XML, ki omogoča strukturirano, standardizirano in neodvisno izmenjavo podatkov med različnimi informacijskimi sistemi. XML je zaradi svoje berljivosti, prilagodljivosti in široke podpore postal ena temeljnih tehnologij v logistiki, informatiki in spletnem okolju. Z njegovo uporabo lahko podjetja, dobavitelji, prevozniki in skladišča učinkovito »govorijo skupen jezik«, kar zmanjšuje napake, pospešuje pretok informacij in zagotavlja boljšo sledljivost procesov.

Pomni!

XML je preprost, a zelo natančen jezik saj že majhne napake (npr. manjkajoče zaklepne oznake ali nepravilno kodirani znaki) lahko povzročijo, da dokument postane neberljiv za avtomatično obdelavo. Zato je redno preverjanje s prevajalnikom in urejevalniki, ki podpirajo XML, nujno za zanesljivo delo.

Poglavje najprej pojasni osnovne značilnosti XML in njegovo vlogo pri elektronski izmenjavi podatkov, nato postopoma predstavi notranjo strukturo dokumentov (elementi, atributi, komentarji, posebni znaki) in uporabo imenskih prostorov za preprečevanje trkov imen. Sledi prikaz praktičnih postopkov za preverjanje in validacijo XML z orodji, kot so brskalniki in Notepad++, ter poudarek na pravilih za dobro oblikovane in veljavne dokumente. Zaključni deli vključujejo najpogostejše napake, kontrolni seznam za preverjanje pravilnosti ter predstavitev naprednejših tehnologij (XPath, XSLT, RSS, XLink), ki razširjajo uporabo XML v realnih informacijskih in logističnih sistemih.

Primer v praksi

Logistično podjetje dnevno izmenjuje naročilnice in odpremnice z različnimi partnerji. Namesto klasičnih papirnih dokumentov in posledično ročnega vnosa podatkov uporablja standardizirane XML dokumente, ki jih različni informacijski sistemi lahko samodejno generirajo in obdelujejo v skladu s splošno sprejetimi in poznanimi standardi za obliko tovrstnih dokumentov. S tem podjetje prihrani čas, zmanjša število napak pri vnosu in zagotovi sledljivost pošiljk od naročila do dostave. Pri tem je velikega pomena tudi digitalni arhiv dokumentov, saj so XML dokumenti v osnovi v digitalni obliki. Digitalni arhiv omogoča predvsem zmanjševanje stroškov za iskanje posameznega dokumenta in zagotavlja, da se dokument ne založi ali uniči v svojem življenjskem ciklu obdelave. Seveda to velja ob predpostavki, da so informacijski sistemi ustrezno varovani.

3.1 Kaj je XML in zakaj je pomemben?

XML (angl. *eXtensible Markup Language*) je jezik za opisovanje strukturiranih podatkov v besedilni obliki. Zasnovan je bil tako, da so podatki:

- **neodvisni od programske in strojne opreme,**
- **čitljivi za ljudi in za računalnike,**
- **enostavni za izmenjavo med različnimi sistemi.**

V logistiki se XML pogosto uporablja za:

- elektronsko izmenjavo naročilnic, odpremnic, računov, tovornih listov in drugih dokumentov, ki so del logističnih tokov,
- izmenjavo podatkov med informacijskimi sistemi dobaviteljev, naročnikov, prevoznikov in skladišč,
- shranjevanje konfiguracij, metapodatkov in poročil v standardizirani obliki.

Uporaba XML omogoča, da različni informacijski sistemi »**govorijo skupen jezik**«, kar zmanjšuje število napak, pospeši pretok informacij in poveča sledljivost v oskrbovalnih verigah.

Primer preproste XML datoteke

Spodaj je primer enostavne XML datoteke, ki opisuje naročilnico:

```
<?xml version="1.0" encoding="UTF-8"?>
<narocilo>
    <stevilka>2025-001</stevilka>
    <datum>2025-10-04</datum>
    <stranka>Logistika d.o.o.</stranka>
    <postavke>
        <izdelek>
            <naziv>Paletni voziček</naziv>
            <kolicina>2</kolicina>
        </izdelek>
        <izdelek>
            <naziv>Transportni trak</naziv>
            <kolicina>1</kolicina>
        </izdelek>
    </postavke>
</narocilo>
```

Ključne značilnosti:

- dokument ima **en korenski element** (<narocilo>),
- elementi so **pravilno gnezdeni**,
- podatki so zapisani v **strukturi drevesa**.

Tabela 1: Ključni pojmi

Pojem	Pojasnilo
Korenski element	Najvišji element v dokumentu (npr. <narocilo>). Dokument lahko ima samo en korenski element.
Element	Osnovni gradnik XML. Ima začetno in končno oznako (npr. <datum>…</datum>).
Dobro oblikovan dokument	Dokument, ki sledi pravilom sintakse XML (pravilno gnezdenje, en koren, ujemajoče oznake).
Kodiranje	Način zapisa znakov (najpogosteje UTF-8). Navedeno je v deklaraciji dokumenta.

3.1.1 Osnovne komponente obdelave XML dokumentov

Za razumevanje delovanja XML je koristno poznati osnovne komponente, ki sodelujejo pri branju, preverjanju in obdelavi dokumentov. Te komponente delujejo zaporedno — od surovega besedila do strukturiranih podatkov, ki jih lahko uporablja programska oprema.

Tabela 2

Komponenta	Vloga in naloga	Primer delovanja
Leksikalni analizator (Lexer) / Tokenizer	Prebere surovo besedilo in ga razdeli na osnovne gradnike (žetone), kot so oznake, atributi, vrednosti.	Iz <izdelek sifra="A001"> ustvari tok žetonov: <izdelek, sifra, "A001", >
Razčlenjevalnik (Parser)	Iz žetonov zgradi notranjo drevesno strukturo dokumenta (npr. DOM) in preveri skladnost s pravili XML.	Ugotovi, kateri elementi so gnezdeni v katere, preveri ujemajoče oznake, strukturo in vrstni red.
Prevajalnik sintakse (Syntax checker)	Preveri, ali dokument sledi osnovnim pravilom sintakse (npr. en korenski element, pravilno gnezdenje, ujemajoče oznake).	Ob napaki javi: »Error on line 12, mismatched closing tag.«
Validator	Preveri, ali dokument poleg sintakse ustreza tudi določeni shemi (npr. XSD ali DTD), ki določa dovoljene elemente, tipe podatkov in strukturo.	Če manjka zahtevan element, validator javi napako o neveljavnosti.
DOM (Document Object Model)	Drevesna predstavitev XML dokumenta v pomnilniku, ki omogoča programom, da dostopajo do posameznih elementov in jih spreminjajo.	Z DOM lahko v programu poiščemo vse <izdelek> elemente in preberemo njihove atribute.
XPath / XSLT procesorji	Omogočajo iskanje, filtriranje in preoblikovanje podatkov znotraj XML dokumenta.	XPath izraz //izdelek[@sifra='A001'] poišče točno določen element.

Pomni!

- **Leksikalni analizator in razčlenjevalnik** delujeta "v ozadju" večine orodij (npr. brskalnikov, Notepad++ XML Tools) — uporabnik ju neposredno ne vidi, a brez njiju preverjanje dokumentov sploh ne bi bilo mogoče.
- **Prevajalnik sintakse** preverja **obliko**, **validator** pa **vsebino glede na pravila**.
- **Objektni model dokumenta - DOM** je ključnega pomena za nadaljnjo strojno obdelavo (npr. iz XML v HTML z uporabo XSLT).

Primer v praksi

Ko XML datoteko izdelki.xml odpreš v Firefoxu ali Notepad++:

1. **Leksikalni analizator** prebere besedilo in ustvari žetone.
2. **Razčlenjevalnik** preveri strukturo in ustvari DOM.
3. **Prevajalnik sintakse** javi napako, če je struktura nepravilna.
4. Če je datoteka povezana s shemo, **validator** preveri skladnost s pravili.
5. Če je vse pravilno, brskalnik prikaže lepo drevesno strukturo, orodja pa omogočijo uporabo XPath izrazov.

3.1.2 Mini kviz: preveri svoje razumevanje

Odgovori z »**Da**« ali »**Ne**«:

1. XML dokument lahko vsebuje več korenskih elementov.
2. XML je občutljiv na velikost črk.
3. XML je namenjen samo prikazovanju podatkov v brskalniku.
4. Dobro oblikovan dokument ima pravilno gnezdene elemente.
5. XML je uporaben tudi za dolgoročno arhiviranje podatkov.

```
<resitve>
    <vprasanje stevilka="1">
        <odgovor>Ne</odgovor>
        <pravilen>false</pravilen>
    </vprasanje>
    <vprasanje stevilka="2">
        <odgovor>Da</odgovor>
        <pravilen>true</pravilen>
    </vprasanje>
    <vprasanje stevilka="3">
        <odgovor>Ne</odgovor>
        <pravilen>false</pravilen>
    </vprasanje>
    <vprasanje stevilka="4">
        <odgovor>Da</odgovor>
        <pravilen>true</pravilen>
    </vprasanje>
    <vprasanje stevilka="5">
        <odgovor>Da</odgovor>
        <pravilen>true</pravilen>
    </vprasanje>
</resitve>
```



3.1.3 Vaja 1 – Prva XML datoteka

Naloga:

1. Ustvari XML dokument z imenom izdelek.xml.
2. V njem opiši en izdelek (naziv, šifra, cena in količina).
3. Poskrbi, da bo dokument dobro oblikovan (en koren, pravilno gnezdenje, ujemajoče oznake).
4. Odpri ga v Notepad++ in preveri sintakso z vtičnikom *XML Tools* (Pretty Print + Check XML syntax).

Namig: če dokument ni dobro oblikovan, bo validator vrnil napako z opisom vrstice in znaka, kjer se težava pojavi.

3.1.4 Povzetek

- XML je standardiziran jezik za izmenjavo strukturiranih podatkov.
- V logistiki omogoča avtomatizirano in zanesljivo komunikacijo med informacijskimi sistemi.
- Dobro oblikovan XML dokument ima en korenski element, pravilno gnezdenje in jasno strukturo.
- Za delo z XML lahko uporabimo preprosta orodja, kot je Notepad++ z XML vtičnikom.

3.2 Osnovna struktura XML

Po osnovnem razumevanju, kaj je XML in zakaj ga uporabljamo, se zdaj osredotočimo na **njegovo notranjo zgradbo**. Razumevanje elementov, atributov, komentarjev in posebnih znakov je temelj za ustvarjanje pravilnih in uporabnih XML dokumentov.

3.2.1 Elementi in njihova hierarhija

Elementi so osnovni gradniki XML. Vsak element ima:

- **začetno oznako** (npr. <izdelek>),
- **vsebino** (npr. Paletni voziček),
- **končno oznako** (npr. </izdelek>).

Primer preprostega XML elementa:

```
<naziv>Paletni voziček</naziv>
```

Elementi so **gnezdeni v hierarhijo drevesa**, kjer obstaja:

- **korenski (root)** element — dokument ima vedno samo en korenski element,
- **podelementi (child elements)**, ki so znotraj drugih elementov,
- **besedilna vozlišča**, ki vsebujejo dejanske podatke.

Primer hierarhije:

```
<narocilo>
    <stevilka>2025-001</stevilka>
    <postavke>
        <izdelek>
            <naziv>Paletni voziček</naziv>
            <kolicina>2</kolicina>
        </izdelek>
        <izdelek>
            <naziv>Transportni trak</naziv>
            <kolicina>1</kolicina>
        </izdelek>
    </postavke>
</narocilo>
```

Tukaj je:

– <narocilo> korenski element,
– <postavke> je njegov podelement,
– <izdelek> je gnezden dvakrat znotraj <postavke>.

Ta hierarhija omogoča **jasno strukturo podatkov**, kar je ključnega pomena pri logističnih dokumentih, kjer je potrebna natančna in avtomatizirana obdelava.

3.2.2 Atributi in njihova vloga (metapodatki)

Atributi so dodatne informacije, zapisane znotraj začetne oznake elementa.

Običajno opisujejo lastnosti elementa, ne pa vsebine same.

Primer:

```
<izdelek koda="P001" enota="kos">
    <naziv>Paletni voziček</naziv>
    <kolicina>2</kolicina>
</izdelek>
```

Tukaj:

– koda="P001" in enota="kos" sta atributa elementa <izdelek>,
– vsebina elementa (naziv in količina) ostaja ločena.

Atributi so odlični za kratke, opisne podatke, ki ne potrebujejo dodatne hierarhije.

Elementi pa so primernejši za podatke, ki imajo lastno strukturo ali več vrstic.

Primer: knjige, naročila, izdelki

Poglejmo primer knjižnega kataloga (poenostavljeno):

```
<knjiga isbn="978-961-000000-0">
    <naslov xml:lang="sl">Uvod v XML</naslov>
    <avtor>Janez Novak</avtor>
    <leto>2025</leto>
</knjiga>
```

V logistiki bi lahko bil podoben primer **opis izdelka v naročilu**:

```
<izdelek sifra="X123" enota="kos">
    <naziv>Embalažna škatla</naziv>
    <kolicina>50</kolicina>
</izdelek>
```

Atributi (sifra, enota) opisujejo **metapodatke**, medtem ko elementi vsebujejo **dejanske podatke** (naziv, količina).

3.2.3 Odločanje med elementi in atributi (enostavna pravila)

Pri ustvarjanju XML dokumentov se pogosto pojavi vprašanje:

Ali naj nekaj zapišem kot atribut ali kot element?

Spodnja tabela pomaga pri hitri odločitvi:

Tabela 3

Uporabi element, kadar:	Uporabi atribut, kadar:
ima podatek lastno strukturo ali podelemente.	je podatek kratek, opisni metapodatek.
je vsebina daljša, večvrstična.	je podatek identifikator ali oznaka.
želiš omogočiti razširjanje strukture.	vrstni red podatkov ni pomemben.

Primer:

```
<!-- Bolj primerno kot atribut: več podatkov, možnost gnezdenja -->
<naslov>
    <glavni>Uvod v XML</glavni>
    <podnaslov>Standard za izmenjavo podatkov</podnaslov>
</naslov>
```

```
<!-- Primerno kot atribut: kratki podatki -->
<izdelek sifra="P123" enota="kos"/>
```

3.2.4 Komentarji in posebni znaki

Komentarji v XML so namenjeni bralcem in jih parser ignorira:

```
<!-- To je komentar -->
```

Posebni znaki, kot so <, >, &, morajo biti zapisani z **entitetami**, da ne zmedejo parserja:

Tabela 4

Znak	Entiteta
<	<
>	>
&	&
"	"
'	'

Primer:

```
<opomba>Uporabi &lt;tag&gt; za označevanje elementov.</opomba>
```



3.2.5 Vaja 2 – Seznam izdelkov

Naloga:

Ustvari XML dokument z imenom izdelki.xml, ki vsebuje seznam vsaj treh izdelkov.

Za vsak izdelek zapiši:

- naziv,
- šifro (atribut),
- enoto mere (atribut),
- količino (element).

Primer začetka:

```
<izdelki>
    <izdelek sifra="A001" enota="kos">
        <naziv>Transportni trak</naziv>
        <kolicina>3</kolicina>
    </izdelek>
    <!-- Dodaj še dva izdelka -->
</izdelki>
```

Preveri, da ima dokument en korenski element (<izdelki>), pravilno gnezdenje in nobenih posebnih znakov brez entitet.

Za preverjanje uporabi Notepad++ in XML Tools.

3.2.6 Povzetek

- Elementi so osnovni gradniki XML in tvorijo drevesno strukturo.
- Atributi služijo kot metapodatki elementov — uporabni za kratke opise in oznake.
- Pravilna odločitev med elementi in atributi poveča preglednost in uporabnost dokumenta.
- Komentarji pomagajo ljudem, posebni znaki pa zahtevajo ustrezno kodiranje.
- Preprost primer iz logistike je seznam izdelkov z nazivi, količinami in šiframi.

3.3 Imenski prostori (Namespaces)

3.3.1 Zakaj potrebujemo imenske prostore

Pri enostavnih XML dokumentih, ki vsebujejo le nekaj elementov, večinoma ni težav. Ko pa **združujemo podatke iz različnih virov** (npr. notranje evidence podjetja in zunanje standarde), lahko pride do **trkov imen** - to pomeni, da **dva elementa z enakim imenom pomenita različni stvari**.

Primer težave:

```
<dokument>
    <naslov>Seznam izdelkov</naslov>
    <naslov>
        <ulica>Glavna cesta 1</ulica>
        <posta>3000 Celje</posta>
    </naslov>
</dokument>
```

Tu element <naslov> pomeni:

– enkrat **naslov dokumenta**,
– drugič **poštni naslov** podjetja.

Parser ne zna razlikovati med njima - oba imata isto ime. Rešitev: **imenski prostori (namespaces)**.

3.3.2 Kako delujejo imenski prostori

Imenski prostor je v bistvu **oznaka (URI)**, ki določa, *iz katerega slovarja izvirajo elementi.*

V dokumentu jih povežemo s **prefiksi** (npr. ex: za »example«).

Primer:

```
<dokument xmlns:doc="http://primer.si/dokument"
          xmlns:adr="http://primer.si/naslovi">
    <doc:naslov>Seznam izdelkov</doc:naslov>
    <adr:naslov>
        <adr:ulica>Glavna cesta 1</adr:ulica>
        <adr:posta>3000 Celje</adr:posta>
    </adr:naslov>
</dokument>
```

– xmlns:doc="http://primer.si/dokument" → deklarira imenski prostor za **dokumente**
– xmlns:adr="http://primer.si/naslovi" → deklarira prostor za **naslove**

Elementi so zdaj **nedvoumni**, ker so označeni s prefiksoma doc: in adr:.

URI je lahko katerikoli niz - pogosto je v obliki URL-ja, vendar **ni nujno, da kaže na dejansko spletno stran**. Pomemben je kot **enolični identifikator slovarja**.

3.3.3 Privzeti (default) imenski prostor

Namesto prefiksov lahko določimo tudi **privzeti imenski prostor** za elemente brez prefiksa:

```
<knjiga xmlns="http://primer.si/knjige">
    <naslov>Uvod v XML</naslov>
    <avtor>Janez Novak</avtor>
</knjiga>
```

– xmlns="..." določi, da vsi elementi brez prefiksa (knjiga, naslov, avtor) pripadajo temu prostoru.
– Atributi **ne podedujejo** privzetega prostora — če želimo atribut v imenskem prostoru (namespace), ga moramo eksplicitno prefiksirati.

3.3.4 Pogoste napake pri uporabi imenskega prostora

Tabela 5

Napaka	Primer	Razlaga
Uporaba prefiksa brez deklaracije	<abc:izdelek>...</abc:izdelek>	Prefiks abc ni definiran z xmlns:abc
Napačno gnezdenje elementov		Dokument ni dobro oblikovan
Podvajanje deklaracij	Dvakrat xmlns:ex z različnimi URI	Prefiks mora imeti en pomen
Neveljavni znaki v imenih	<1element>	Element se ne sme začeti s številko
Zanašanje na prefiks namesto URI	dva različna prefiksa → isti URI	Pomemben je URI, ne ime prefiksa

3.3.5 Dobre prakse pri imenskem prostoru

– Deklariraj imenske prostore **na začetku dokumenta** (korenski element).
– Uporabljaj **kratke in smiselne prefikse** (npr. ex, adr, dc).
– Loči podatke različnih domen v različne prostore.
– Dokumentiraj, kateri URI pripada kateremu prefiksu.
– Izogibaj se prekomerni uporabi imenskih prostorov in jih uporabi, ko dejansko rešujejo težavo.

3.3.6 Vaja 3 – Dodajanje imenskega prostora

Naloga:

1. Vzemi svoj XML dokument izdelki.xml iz prejšnje vaje.
2. Dodaj mu imenski prostor xmlns:ex="http://primer.si/izdelki".
3. Vsem elementom dodaj prefiks ex: (npr. <ex:izdelek> namesto <izdelek>).
4. Dodaj en element iz tujega standarda (npr. Dublin Core za <dc:title>):
5. xmlns:dc="http://purl.org/dc/elements/1.1/"
6. Namerno **izpusti deklaracijo enega prefiksa** in preveri, kakšno napako bo vrnil validator.
7. Nato napako popravi.

Primer rešitve:

```
<ex:izdelki xmlns:ex="http://primer.si/izdelki"
            xmlns:dc="http://purl.org/dc/elements/1.1/">
    <ex:izdelek sifra="A001" enota="kos">
        <dc:title>Transportni trak</dc:title>
        <ex:kolicina>3</ex:kolicina>
    </ex:izdelek>
</ex:izdelki>
```

Validatorji bodo takoj zaznali, če kak prefiks ni deklariran ali je napačno uporabljen.

Povzetek

- Imenski prostori preprečujejo **trke imen** v XML dokumentih, še posebej pri združevanju različnih slovarjev.
- Določimo jih z atributom xmlns in po potrebi uporabimo prefikse.
- Pomemben je URI, ki enolično označuje slovar — ne nujno spletna stran.
- Pogoste napake vključujejo nedeklarirane prefikse, napačno gnezdenje in podvajanje deklaracij.
- Uporaba imenskih prostorov je pogosta v standardih, kot so **Dublin Core**, **UBL (Universal Business Language)** ali **GS1 XML** za logistične procese.

3.4 Prikaz in validacija XML

Razumevanje notranje strukture XML je zelo pomembno, vendar je enako bistveno znati **dokument pravilno prikazati in preveriti**, ali je ustrezno oblikovan in veljaven. V tem poglavju bomo spoznali, kako XML deluje kot drevo podatkov, kako ga prikažemo v brskalniku in kako ga preverimo z orodji, kot je Notepad++.

3.4.1 XML kot drevo podatkov

Vsak XML dokument je logično organiziran kot **drevo vozlišč** (ang. tree structure):

- na vrhu je **korenski element**,
- pod njim so **podelementi** (otroci),
- ti lahko vsebujejo svoje podelemente, **besedilna vozlišča** in **atribute**.

Primer:

```
<bookstore>
    <book category="children">
        <title lang="en">Harry Potter</title>
        <author>J. K. Rowling</author>
        <year>2005</year>
        <price>29.99</price>
    </book>
    <book category="cooking">
        <title lang="en">Everyday Italian</title>
        <author>Giada De Laurentiis</author>
        <year>2005</year>
        <price>30.00</price>
    </book>
</bookstore>
```

Drevesna predstavitev:

```
bookstore (root)
├── book (category="children")
│   ├── title (lang="en") → "Harry Potter"
│   ├── author → "J. K. Rowling"
│   ├── year → 2005
│   └── price → 29.99
└── book (category="cooking")
    ├── title (lang="en") → "Everyday Italian"
    ├── author → "Giada De Laurentiis"
    ├── year → 2005
    └── price → 30.00
```

Razumevanje drevesne strukture omogoča učinkovito uporabo drugih XML tehnologij, kot sta XPath in XSLT, saj se nanašata na posamezna vozlišča.

3.4.2 Prikaz XML v brskalniku

XML ni namenjen vizualnemu oblikovanju kot HTML, temveč **strukturiranemu prikazu podatkov**.

Če XML datoteko odprete v sodobnem brskalniku (npr. Firefox, Edge in Chrome), boste običajno videli:

- podatke z zamiki,
- razširljive elemente,
- barvno označevanje oznak in vsebine,
- opozorila, če dokument ni pravilno oblikovan.

Primer:

- Shrani primer XML kot books.xml.
- Odpri ga v Firefoxu.
- Brskalnik prikaže drevesno strukturo z možnostjo razširjanja in zapiranja elementov.

Če XML ni pravilno oblikovan, brskalnik prikaže napako in označi vrstico, kjer parser »zastane«.

3.4.3 Osnovna pravila za dobro oblikovan XML (ang. well-formed)

Za uspešen prikaz v brskalniku mora dokument izpolnjevati naslednje pogoje:

- en sam korenski element,
- pravilno gnezdenje elementov,
- ujemajoče začetne in končne oznake,
- pravilno kodiranje znakov (UTF-8),
- kodirane posebne znake (<, >, & …).

Primer napačnega XML:

```
<bookstore>
    <book>
        <title>Primer knjige<title>
    </book>
</bookstore>
```

Manjka zaklepna oznaka </title> → dokument ni dobro oblikovan.

3.4.4 Validacija XML z uporabo shem

Dobro oblikovan dokument je sintaktično pravilen.

Veljaven dokument pa poleg tega **ustreza določeni shemi** (npr. XSD), ki določa:

- katere elemente lahko uporabimo,
- kakšna je njihova struktura,
- katere tipe podatkov vsebujejo.

Primer XSD (poenostavljeno):

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="izdelek">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="naziv" type="xs:string"/>
                <xs:element name="kolicina" type="xs:integer"/>
            </xs:sequence>
            <xs:attribute name="sifra" type="xs:string" use="required"/>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

Primer XML, ki ustreza tej shemi:

```
<izdelek sifra="A001">
    <naziv>Transportni trak</naziv>
    <kolicina>3</kolicina>
</izdelek>
```

Če bi manjkala oznaka sifra ali element kolicina, bi validator dokument označil kot neveljaven.

3.4.5 Uporaba Notepad++ za preverjanje in potrjevanje ustreznosti

Notepad++ je brezplačno, hitro in preprosto orodje, primerno za delo z XML v učnem okolju.

Namestitev in priprava:

1. Namesti Notepad++ (Windows).
2. Odpri **Plugins Admin** → Namesti vtičnik **XML Tools**.
3. V meniju **Language** izberi »XML« za barvanje sintakse.
4. Nastavi kodiranje na **UTF-8 (without BOM)**.

Osnovne funkcije (XML Tools):

- **Pretty Print (XML only)** → samodejno poravna zamike.
- **Check XML syntax now** → preveri, ali je dokument dobro oblikovan.
- **Validate now** → preveri dokument glede na XSD shemo (če je navedena).
- **Evaluate XPath** → omogoča iskanje vozlišč po drevesu.

Hitri postopek:

1. Odpri XML datoteko.
2. Plugins → XML Tools → Check XML syntax now.
3. Če je dokument pravilen, ga oblikuj (Pretty Print).
4. Če je shema dodana, izberi Validate now.



3.4.6 Vaja 4 – Preverjanje XML

Naloga:

1. Uporabi svoj dokument izdelki.xml.
2. Odpri ga v brskalniku in preveri, ali se pravilno prikaže.
3. Nato ga odpri v Notepad++ in preveri sintakso (Check XML syntax).
4. Namenoma odstrani eno zaklepno oznako → preveri, kakšno napako vrne validator.
5. (Dodatno) Ustvari preprosto XSD shemo in jo poveži z dokumentom s pomočjo:
6. xsi:schemaLocation="http://primer.si/izdelki izdelki.xsd"
7. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

S tem študenti spoznajo razliko med dobro oblikovanim in veljavnim dokumentom v praksi.

3.4.7 Povzetek

– XML dokumenti so organizirani kot drevesa, kar omogoča sistematično delo z vozlišči.
– Brskalniki omogočajo preprost prikaz XML in hitro zaznajo napake v sintaksi.
– Dobro oblikovan dokument je nujen za pravilno delovanje orodij in nadaljnjo obdelavo.
– Validacija z XSD zagotavlja, da dokument sledi določeni strukturi in tipom podatkov.
– Notepad++ z XML Tools je uporabno orodje za preverjanje, oblikovanje in validacijo XML dokumentov v učnem procesu.

3.5 Povzetek in pogoste napake

XML je osnova za standardizirano izmenjavo podatkov v številnih informacijskih sistemih, tudi v logistiki. Čeprav je sintaksa XML razmeroma preprosta, se napake pogosto pojavijo zaradi površnosti, nepozornosti ali nepravilne uporabe osnovnih pravil. V tem

poglavju bomo ponovili ključne elemente dobro oblikovanega XML dokumenta in opozorili na pogoste napake, ki se jim lahko izognemo z nekaj preprostimi preveritvami.

3.5.1 Elementi in atributi

Elementi in atributi imajo različno vlogo in ju je smiselno uporabljati skladno z namenom:

Tabela 5

Elementi	Atributi
Vsebujejo podatke in strukturo.	Vsebujejo metapodatke.
Lahko se gnezdijo in razširjajo.	Ne morejo imeti podelementov.
Primerni so za večvrstične vsebine.	Primerni so za kratke oznake in lastnosti.
Njihov vrstni red je lahko pomemben.	Vrstni red atributov ni pomemben.

Pogosta napaka: shranjevanje dolgih opisov kot atributov.

Rešitev: uporabite elemente za besedilo, atribute pa za kratke oznake (npr. šifre, enote). Primer:

```
<!-- Napačno -->
<izdelek opis="Transportni trak, ki se uporablja v skladiščih"/>

<!-- Pravilno -->
<izdelek>
    <opis>Transportni trak, ki se uporablja v skladiščih</opis>
</izdelek>
```

3.5.2 Namespace deklaracije

Imenski prostori (namespaces) omogočajo, da so elementi nedvoumni tudi, ko združujemo različne slovarje. Najpogostejše napake se pojavijo pri deklaracijah.

Tabela 6

Napaka	Posledica	Rešitev
Uporaba prefiksa brez deklaracije.	Parser vrne napako »prefix not bound«.	Dodaj manjkajočo xmlns:prefix deklaracijo.
Napačno gnezdenje deklaracij.	Elementi niso pravilno razpoznani.	Deklaracije postavi na korenski element.
Podvajanje deklaracij z različnimi URI.	Nejasnost, nedoslednost.	Uporabi enoten URI za isti prefiks.

Primer:

```
<!-- Napačno -->
<ex:izdelek>
    <dc:title>Primer</dc:title>
</ex:izdelek>

<!-- Manjka deklaracija ex in dc -->
<!-- Pravilno -->
<ex:izdelek xmlns:ex="http://primer.si/izdelki"
            xmlns:dc="http://purl.org/dc/elements/1.1/">
    <dc:title>Primer</dc:title>
</ex:izdelek>
```

3.5.3 Kodiranje in posebni znaki

Ker XML uporablja znake <, >, & in narekovaje kot del sintakse, jih je treba znotraj besedil ali atributov **ustrezno kodirati**.

Tabela 7

Znak	Entiteta
<	<
>	>
&	&
"	"
'	'

Primer:

```
<opomba>Uporabi &lt;izdelek&gt; za označevanje elementov.</opomba>
```

Pogosta napaka:

Vnos posebnih znakov neposredno → parser zazna napako.
Rešitev: uporaba ustreznih entitet.

Poleg tega mora biti celoten dokument enotno kodiran, običajno v **UTF-8 brez BOM**. To omogoča pravilno obravnavo šumnikov (č, š, ž) in drugih znakov.

3.5.4 Uporaba orodij

Pri delu z XML je smiselno uporabljati preprosta, a zmogljiva orodja, ki pomagajo hitro odkriti napake:

- **Brskalnik** – hiter prikaz XML in osnovno preverjanje sintakse.
- **Notepad++ + XML Tools** – preverjanje sintakse, Pretty Print, validacija proti XSD.
- **Online validatorji** – npr. https://www.xmlvalidation.com.
- **XPath testerji** – za preverjanje iskanja po drevesu.

Redna uporaba validatorjev med delom prepreči kopičenje napak in olajša nadaljnjo obdelavo dokumentov.

Kontrolni seznam za dober XML dokument:

- en sam korenski element,
- pravilno gnezdenje in ujemajoče oznake,
- kodirani posebni znaki,
- uporabljeni ustrezni elementi in atributi,
- pravilne deklaracije namespaces (če so uporabljeni),
- UTF-8 kodiranje brez BOM,
- preverjeno z validatorjem ali orodjem (npr. Notepad++)

3.5.5 Mini kviz in refleksija

Navodila

Preglej spodnje primere in zapiši, kaj je narobe ter kako bi napako popravil. Odgovori naj bodo kratki in jasni.

Primer 1

```
<izdelek sifra="A001">
      <naziv>Transportni trak</naziv>
      <kolicina>3</kolicina>
</izdelek
```

Primer 2

```
<ex:izdelek>
    <dc:title>Primer</dc:title>
</ex:izdelek>
```

Primer 3

```
<opomba>Uporabi <izdelek> za označevanje.</opomba>
```

Primer 4

```
<izdelek sifra=A001>
    <naziv>Izdelek brez narekovajev</naziv>
</izdelek>
```

Primer 5

```
<narocilo>
    <izdelek sifra="B123"/>
</narocilo>
<narocilo>
    <izdelek sifra="C456"/>
</narocilo>
```

Primer 6

```
<izdelek sifra="A001">
    <naziv>Paletni voziček</naziv>
<izdelek>
```

Primer 7

```
<izdelek sifra="A001">
    <kolicina>5</kolicina>
    <kolicina enota="kos">5</kolicina>
</izdelek>
```

Primer 8

```
<ex:izdelek xmlns:ex="http://primer.si/izdelki"
xmlns:ex="http://drugprimer.si/izdelki">
    <ex:naziv>Transportni trak</ex:naziv>
</ex:izdelek>
```

Primer 9

```
<izdelek sifra="A001">
    <opis>Transportni trak & dodatki</opis>
</izdelek>
```

Primer 10

```
<izdelek sifra="A001">
    <naziv>Napaka v imenu</naziv>
</izdelek>
```

Refleksija

- Katera napaka se ti zdi najlažja za spregled?
- Kako bi si pri delu zagotovil, da jih pravočasno zaznaš?
- Kako ti lahko validator pomaga pri vsakodnevnem delu z XML?

3.5.6 Povzetek

- Elementi in atributi imata različne funkcije; pravilna uporaba izboljša preglednost.
- Namespace deklaracije morajo biti dosledne in jasno definirane.
- Posebni znaki zahtevajo kodiranje, sicer pride do napak pri obdelavi.
- Orodja in validatorji so nepogrešljivi za hitro odkrivanje in odpravljanje težav.
- Kontrolni seznam in sprotno preverjanje bistveno zmanjšata verjetnost napak v dokumentih.

3.6 Dodatno gradivo

Poleg osnov XML obstajajo številne razširitve in tehnologije, ki omogočajo naprednejšo obdelavo in uporabo podatkov. Celovit uvod v XML ponujata Dykes in Tittel (2005) ter spletni vodnik W3Schools (b. d.), vsebine pa dopolnjuje repetitorij Informatika in informacijska varnost (Jereb, 2019). V tem poglavju so predstavljene štiri pomembne teme za samostojno raziskovanje: XPath, XSLT, RSS in XLink. Te tehnologije se pogosto uporabljajo v realnih informacijskih sistemih in logističnih procesih.

3.6.1 XPath – osnovni izrazi za iskanje po XML drevesu

XPath (XML Path Language) je jezik za **naslavljanje in iskanje podatkov** v XML dokumentih. Omogoča natančno določanje elementov ali atributov glede na njihovo mesto v drevesu.

Primer XML:

```
<izdelki>
    <izdelek sifra="A001">
        <naziv>Transportni trak</naziv>
        <kolicina>3</kolicina>
    </izdelek>
    <izdelek sifra="A002">
        <naziv>Paletni voziček</naziv>
        <kolicina>2</kolicina>
    </izdelek>
</izdelki>
```

Osnovni XPath izrazi:

Tabela 8

XPath izraz	Pojasnilo	Rezultat
/izdelki/izdelek/naziv	absolutna pot do elementov <naziv>	oba naziva
//naziv	poišče vse elemente z imenom <naziv> kjerkoli v dokumentu	oba naziva
//izdelek[@sifra='A001']	poišče <izdelek> z določenim atributom	prvi izdelek
//izdelek[2]/naziv	poišče naziv drugega izdelka	"Paletni voziček"
//@sifra	poišče vse vrednosti atributa sifra	"A001", "A002"

XPath je še posebej uporaben pri preverjanju dokumentov, testiranju iskalnih izrazov in transformacijah.

Praktični nasvet:

V Notepad++ lahko z vtičnikom *XML Tools* hitro preverite XPath izraze: Plugins → XML Tools → Evaluate XPath Expression.

3.6.2 XSLT – preoblikovanje XML v HTML

XSLT (Extensible Stylesheet Language Transformations) je jezik za **pretvorbo XML v druge formate**, najpogosteje v HTML za prikaz v brskalniku.

Primer XML (izdelki.xml):

```
<izdelki>
    <izdelek sifra="A001">
        <naziv>Transportni trak</naziv>
        <kolicina>3</kolicina>
    </izdelek>
</izdelki>
```

```
Primer XSLT (izdelki.xsl):
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet          xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="1.0">
  <xsl:template match="/">
    <html>
      <head><title>Seznam izdelkov</title></head>
      <body>
        <h2>Seznam izdelkov</h2>
        <table border="1">
          <tr><th>Šifra</th><th>Naziv</th><th>Količina</th></tr>
          <xsl:for-each select="izdelki/izdelek">
            <tr>
              <td><xsl:value-of select="@sifra"/></td>
              <td><xsl:value-of select="naziv"/></td>
              <td><xsl:value-of select="kolicina"/></td>
            </tr>
          </xsl:for-each>
        </table>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

Povezava XML z XSLT:

```
<?xml-stylesheet type="text/xsl" href="izdelki.xsl"?>
```

Če XML odprete v brskalniku, bo prikazan kot **pregledna HTML tabela**, ne več kot surov XML.

XSLT se pogosto uporablja za generiranje poročil, prikaz naročil ali za pretvorbo podatkov v druge formate (CSV, HTML, besedilo).

3.6.3 RSS in XLink – primeri iz logistike

3.6.3.1 RSS (Really Simple Syndication)

RSS je XML format za **distribucijo obvestil** (npr. novice, objave, spremembe zalog). Primer RSS vira:

```
<rss version="2.0">
  <channel>
    <title>Obvestila logistike</title>
    <link>https://primer.si/logistika</link>
    <description>Novice iz logističnega centra</description>
    <item>
      <title>Nov prihod zalog</title>
      <link>https://primer.si/obvestila/zaloga123</link>
      <pubDate>Sat, 04 Oct 2025 09:00:00 +0100</pubDate>
    </item>
```

```
  </channel>
</rss>
```

V logistiki bi tak vir lahko objavljal:

– prihode novih pošiljk,
– spremembe voznih redov,
– obvestila o zamudah ali zaporah poti.

RSS omogoča avtomatsko obveščanje uporabnikov ali drugih sistemov brez ročnega preverjanja.

3.6.3.2 XLink (XML Linking Language)

XLink omogoča, da v XML **dodajamo hiperpovezave** med dokumenti, podobno kot HTML uporablja <a href="...">.

Primer:

```
<izdelek xmlns:xlink="http://www.w3.org/1999/xlink"
         xlink:href="https://primer.si/izdelek/A001"
         xlink:type="simple">
    <naziv>Transportni trak</naziv>
</izdelek>
```

Tukaj povezava vodi na spletno stran z dodatnimi informacijami o izdelku.

XLink je uporaben za:

– povezovanje logističnih dokumentov (naročilnice ↔ odpremnice),
– sklicevanje na zunanje kataloge,
– sledenje blagu preko zunanjih sistemov.

3.6.4 Reference in orodja za nadaljnje učenje

Za tiste, ki želijo raziskovati naprednejše uporabe XML, so spodaj zbrani izbrani viri:

Dokumentacija in vodiči

– W3C XPath: https://www.w3.org/TR/xpath

- W3C XSLT: https://www.w3.org/TR/xslt
- W3C XLink: https://www.w3.org/TR/xlink
- RSS 2.0 Specification: https://www.rssboard.org/rss-specification

Orodja

- **Notepad++** – preizkus XPath izrazov, ogled XSLT transformacij.
- **Online XPath testerji** – npr. https://xpather.com.
- **Oxygen XML Editor** – profesionalno orodje za delo z XML, XSLT, XSD.
- **VS Code + XML vtičniki** – primerno za razvojne projekte.

3.6.5 Povzetek

- XPath omogoča natančno iskanje in filtriranje podatkov v XML dokumentih.
- XSLT omogoča preoblikovanje XML v druge formate, zlasti HTML.
- RSS se uporablja za razpošiljanje obvestil, XLink pa za povezovanje dokumentov.
- Napredne tehnologije razširjajo uporabnost XML v praksi, zlasti pri povezovanju različnih sistemov.
- Na voljo je veliko odprtokodnih in spletnih orodij za eksperimentiranje in nadaljnje učenje.

3.6.6 Mini projekt: RSS obvestila za logistično skladišče

Cilj naloge

Študent izdela delujoč RSS vir za fiktivno logistično skladišče, ki objavlja aktualna obvestila o stanju zalog, prihodih pošiljk in spremembah v distribuciji.

Naloga spodbuja razumevanje praktične uporabe XML in krepi zmožnost strukturiranja podatkov za avtomatsko obveščanje uporabnikov ali drugih sistemov.

Opis scenarija

Predstavljaj si, da si odgovoren za IT podporo logističnega skladišča podjetja **LogiTrans d.o.o.**.

Skladišče vsak dan objavlja različna obvestila, kot so:

- prihodi novih pošiljk,
- spremembe zalog,
- pomembna logistična opozorila (npr. zamude, posebni dogodki),
- načrtovane prekinitve ali vzdrževalna dela.

Namesto ročnega obveščanja želi podjetje uvesti **RSS kanal**, ki ga bodo lahko spremljali zaposleni in partnerji z brskalnikom ali RSS bralnikom.

Zahteve projekta

1. Ustvari XML datoteko z imenom logistika_rss.xml.
2. XML naj bo strukturiran v skladu s standardom RSS 2.0.
3. V glavi (<channel>) mora dokument vsebovati naslednje elemente:
   - <title> – naslov RSS kanala,
   - <link> – povezava na spletno stran podjetja,
   - <description> – kratek opis vsebine kanala,
   - <language> – jezik vsebine (npr. sl),
   - <pubDate> – datum zadnje posodobitve.
4. Dodaj **vsaj tri <item> elemente**, ki predstavljajo aktualna obvestila:
   - <title> – naslov obvestila,
   - <link> – povezava do podrobnosti,
   - <description> – opis dogodka,
   - <pubDate> – datum objave.
5. V opisu vsaj enega elementa uporabi pravilno zakodirane **posebne znake** (npr. <, >).
6. Preveri, ali je dokument dobro oblikovan in se pravilno prikaže v brskalniku ali RSS bralniku.
7. (Neobvezno za višjo oceno) Dodaj **XLink** povezavo iz enega <item> elementa na drugo z uporabo xlink:href.

Primer rešitve

```
<?xml version="1.0" encoding="UTF-8"?>
<rss version="2.0" xmlns:xlink="http://www.w3.org/1999/xlink">
  <channel>
    <title>LogiTrans RSS Obvestila</title>
    <link>https://www.logitrans.si/</link>
```

```
    <description>Aktualne informacije iz logističnega
skladišča</description>
    <language>sl</language>
    <pubDate>Sat, 04 Oct 2025 09:00:00 +0100</pubDate>

    <item>
      <title>Nov prihod zalog – Skladišče A</title>
      <link>https://www.logitrans.si/obvestila/zaloga-001</link>
      <description>Danes ob 8:30 je prispela nova pošiljka &lt;Elektronske
naprave&gt;.</description>
      <pubDate>Sat, 04 Oct 2025 08:30:00 +0100</pubDate>
    </item>

    <item>
      <title>Zamuda v dobavi – Transportna enota B12</title>
      <link>https://www.logitrans.si/obvestila/dobava-b12</link>
      <description>Dobava iz Maribora zamuja zaradi prometne
nesreče.</description>
      <pubDate>Sat, 04 Oct 2025 07:15:00 +0100</pubDate>
    </item>

    <item xlink:href="https://www.logitrans.si/obvestila/planirano-
vzdrzevanje" xlink:type="simple">
      <title>Načrtovano vzdrževanje sistema</title>
      <link>https://www.logitrans.si/obvestila/vzdrzevanje</link>
      <description>V nedeljo med 22:00 in 24:00 bo sistem začasno
nedosegljiv.</description>
      <pubDate>Fri, 03 Oct 2025 16:00:00 +0100</pubDate>
    </item>
  </channel>
</rss>
```

Če datoteko odpreš v Firefoxu ali v RSS bralniku (npr. Thunderbird, Feedly), bo prikazana kot seznam obvestil.

Kontrolni seznam za oddajo

Preden nalogo oddaš, preveri:

- Ali imaš pravilno deklaracijo RSS (<rss version="2.0">)?
- Ali ima dokument en sam <channel>?
- Ali so prisotni vsi obvezni elementi <title>, <link>, <description>?
- Ali imaš vsaj tri <item> elemente?
- Ali so vsi posebni znaki pravilno zakodirani?
- Ali se dokument pravilno prikaže v brskalniku ali bralniku?
- (Neobvezno) Ali uporabljaš XLink za dodatne povezave?

Dodatni izzivi (za pogumne)

- Dodaj kategorije (<category>) za različne vrste obvestil (npr. zaloge, transport, IT).

– Poveži RSS z lastno XSLT datoteko, ki ga prikaže kot HTML stran z oblikovano tabelo.
– Ustvari Python ali JavaScript skripto, ki samodejno bere tvoj RSS in izpiše zadnja obvestila v konzolo.

Cilji učenja

Z opravljenim projektom boš znal:

– strukturirati XML dokument po RSS standardu,
– uporabiti pravilno kodiranje posebnih znakov,
– preveriti veljavnost in prikaz XML v realnih orodjih,
– razumeti, kako se XML uporablja za avtomatsko distribucijo informacij.

3.7 eSLOG – standard za elektronsko poslovanje v Sloveniji

Digitalna preobrazba gospodarstva in javnega sektorja zahteva učinkovite, zanesljive in standardizirane načine za izmenjavo poslovnih dokumentov. V ta namen je bil v Sloveniji razvit **standard eSLOG** (*Elektronsko poslovanje slovenskega gospodarstva*), ki predstavlja temeljni okvir za strukturirano elektronsko poslovanje med organizacijami.

Standard eSLOG je oblikovala **Gospodarska zbornica Slovenije** (v nadaljevanju GZS, 2020a) v sodelovanju s podjetji, državnimi organi in strokovnjaki z namenom poenotenja elektronskih dokumentov ter izboljšanja medsebojnega povezovanja in sodelovanja med poslovnimi partnerji. Uporablja se tako v zasebnem kot javnem sektorju, zlasti na področju naročanja, dobave in obračuna blaga ali storitev.

Uvedba standarda prinaša številne koristi:

– omogoča **hitrejšo in natančnejšo izmenjavo naročil, dobavnic in računov** (GZS, 2020c);
– zmanjšuje število napak, ki nastanejo pri ročnem vnosu podatkov;
– omogoča **avtomatizacijo poslovnih postopkov** v ERP in logističnih informacijskih sistemih;
– zagotavlja **skladnost z evropskimi standardi** (EN 16931-1, EANCOM 2002 Syntax 4), kar omogoča povezljivost s tujimi poslovnimi partnerji.

Zaradi teh prednosti je eSLOG postal pomemben gradnik digitalizacije poslovnih procesov v Sloveniji. Posebej pomemben je na področju logistike, kjer omogoča standardizirano, sledljivo in avtomatizirano izmenjavo dokumentov vzdolž celotne oskrbovalne verige: od naročila, prek dobave, do izdaje računa.

V tem poglavju je predstavljen razvoj in namen standarda eSLOG, temeljne vrste dokumentov, ki omogočajo elektronsko izmenjavo naročil, dobavnic in računov, ter procesni tok v logistiki. Poudarek je na povezavi med semantičnim in sintaktičnim modelom ter na praktični uporabi eSLOG v ERP in logističnih informacijskih sistemih.

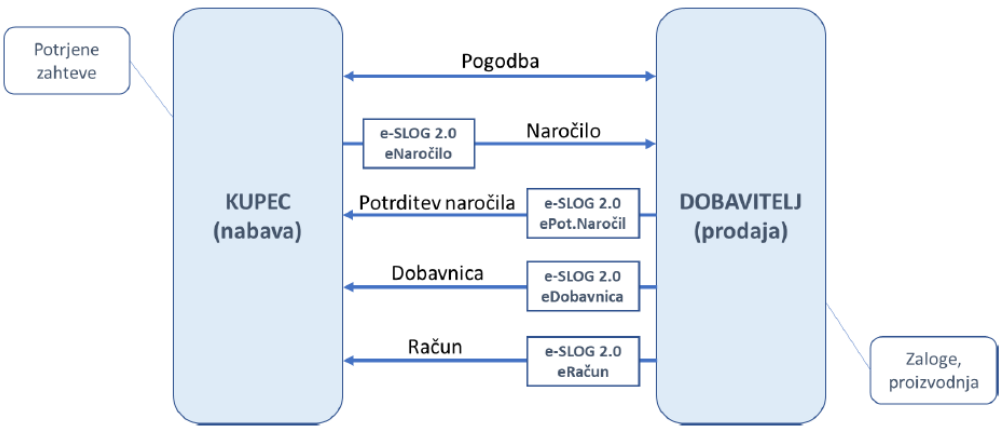
3.7.1 Razvoj standarda eSLOG

Standard eSLOG je bil uveden leta 2001 kot pobuda podjetij za digitalizacijo poslovnih procesov. Prve različice (eSLOG 1.3–1.5) so bile namenjene predvsem **izmenjavi e-računov** (Blažič idr., 2010), zlasti v energetiki, telekomunikacijah in trgovskih verigah.

Kasneje je bil standard nadgrajen v **eSLOG 2.0**, ki vključuje:

- elektronsko naročilo (*ORDERS*),
- elektronsko potrditev naročila (*ORDRSP*),
- elektronsko dobavnico (*DESADV*),
- elektronski račun (*INVOIC*).

eSLOG 2.0 je skladen z evropskim semantičnim standardom EN 16931-1 in temelji na sintaksi UN/EDIFACT. Razvit je bil v okviru projekta **ROSE 2**, sofinanciranega prek instrumenta CEF.



Slika 1: Prikaz uporabe dokumentov eSLOG 2.0 v procesu naročanja.
Vir: Gospodarska zbornica Slovenije, 2020c.

3.7.2 Vrste dokumentov eSLOG 2.0

Standard **eSLOG 2.0** omogoča izmenjavo več vrst elektronskih dokumentov, ki pokrivajo celoten poslovno-logistični proces od naročanja, prek dobave, do obračuna. Vsak dokument ima jasno določen **namen, standardno oznako (kodo)** in **strukturo podatkov**, kar omogoča avtomatizirano obdelavo v ERP/WMS sistemih.

Tabela 9

Dokument	Namen	Koda	Vsebina
eNaročilo (ORDERS)	Naročilo blaga ali storitev	220 / 221 / 227	Glava naročila, postavke, roki, pogoji
ePotrditev naročila (ORDRSP)	Potrditev, sprememba ali zavrnitev naročila	231	Potrjene količine, datumi, spremembe
eDobavnica (DESADV)	Obvestilo o odpremi blaga	351	Podatki o odpremi, prevozniku, pakiranju
eRačun (INVOIC)	Elektronski račun	–	Finančni podatki, postavke, zneski, podpis

eNaročilo (ORDERS)

eNaročilo je začetni dokument v verigi. Pošlje ga **naročnik** dobavitelju, običajno samodejno iz ERP sistema. Vsebuje:

- identifikacijo partnerjev (GLN),
- podatke o dostavi (naslovi, roki),
- seznam naročenih artiklov (GTIN, količine, cene),
- pogoje dobave in plačila.

eNaročilo omogoča, da dobavitelj prejme popolno, strukturirano naročilo brez potrebe po ročnem vnosu. Standardne kode naročil so:

- **220** – naročilo blaga,
- **221** – naročilo storitev,
- **227** – planirano naročilo (npr. okvirne pogodbe in JIT scenariji).

ePotrditev naročila (ORDRSP)

Po prejemu naročila dobavitelj pošlje **ePotrditev naročila**, s katero potrdi, spremeni ali zavrne naročilo. Možni statusi so:

- *»Accepted«* – naročilo potrjeno v celoti,

– *»Accepted with changes«* – naročilo potrjeno z delnimi spremembami (npr. količine, roki),
– *»Rejected«* – naročilo zavrnjeno.

ePotrditev naročila je **ključna za sinhronizacijo pričakovanj** med naročnikom in dobaviteljem, saj se morebitne spremembe uskladijo še pred odpremo blaga. Omogoča sprotno ažuriranje nabavnih planov, transportnih terminov in zalog.

eDobavnica (DESADV)

eDobavnica se uporablja kot **napoved odpreme (ang. ASN – Advance Shipping Notice)**. Pošlje jo dobavitelj, ko je blago pripravljeno za odpremo. Vsebuje:

– referenco na naročilo,
– številko dobavnice, datum odpreme, prevoznika,
– podrobnosti o paketih/paletah (SSCC), količinah, serijah, datumih uporabe ali roku trajanja,
– logistične enote in vsebino posameznih paketov.

eDobavnica omogoča **avtomatsko pripravo prevzema v WMS** naročnika. Skladišče vnaprej prejme podatke o pošiljki, kar omogoča hitro skeniranje SSCC/GTIN kod, takojšen prevzem in zmanjšanje števila napak.

eRačun (INVOIC)

eRačun se pošlje po opravljeni dobavi (ali skladno s pogodbenim pravilom). Vsebuje:

– referenco na naročilo in dobavnico,
– podatke o dobavitelju in prejemniku,
– seznam postavk, cene, davke in skupne zneske,
– podatke za plačilo in digitalni podpis.

eRačun je osnova za **tridimenzionalno ujemanje (3-way match)**:

1. naročilo,
2. dobavnica,
3. račun.

Če se podatki ujemajo, ERP naročnika samodejno potrdi račun in sproži plačilo brez dodatnih ročnih preverjanj.

Pomni!

Vsi dokumenti eSLOG vsebujejo **jasne sklice (ID)** na prejšnje korake:

- DESADV se vedno sklicuje na številko naročila (ORDERS).
- INVOIC se mora sklicevati na naročilo in/ali dobavnico.
- Če sklici manjkajo ali se ne ujemajo, avtomatska obdelava odpove in je potreben ročni poseg.

Primer v praksi

Logistični center trgovske verige odda eNaročilo za 2.000 kosov artikla A. Dobavitelj del naročila potrdi z ePotrditvijo naročila (»Accepted with changes«), ker lahko trenutno dobavi 1.500 kosov. Ob odpremi pošlje eDobavnico z natančnimi podatki o paketih in serijah. Po prevzemu skladišče avtomatsko potrdi količine in poveže dobavnico z naročilom. Dobavitelj nato pošlje eRačun za dejansko dobavljeno količino, ERP pa samodejno izvede tridimenzionalno ujemanje in pripravi plačilo.

3.7.3 Semantični in sintaktični model

Standard eSLOG temelji na uporabi **XML** (eXtensible Markup Language) in **XSD** (XML Schema Definition) shem, ki določajo vsebino in obliko elektronskih dokumentov (GZS, 2020b). Za zagotovitev enotnosti in pravilne interpretacije podatkov uporablja eSLOG dva komplementarna modela:

1. **semantični model**, ki pove, *kaj* podatek pomeni (poslovni pomen, obveznost, pravila),
2. **sintaktični model**, ki pove, *kako* je isti podatek tehnično zapisan v **XML** in preverjen z **XSD**.

Ta razdelitev omogoča, da poslovni uporabniki in razvijalci govorijo isti jezik: prvi o vsebini (BT/BG pojmi) in drugi o strukturi (elementi/atributi XML).

Semantični model (BT/BG)

- BT (*Business Term*) je posamezen podatek (npr. številka dobavnice, datum odpreme, GTIN artikla).
- BG (*Business Group*) je smiselna skupina BT-jev (npr. »Podatki o kupcu«, »Postavka«, »Davki«).
- Določa: pomen, obveznost (obvezno/možno/ponovljivo), format (npr. datum, številka) in validacijska pravila.

Sintaktični model (XML/XSD)

- XML predstavlja **dejansko strukturo dokumenta** (gnezdenje elementov, atribute, vrstni red).
- XSD shema določa **pravila validacije** (tipi, obveznost, ponovljivost).
- Cilj je doseči **strojno obdelavo brez ročnega vnašanja** ter **doslednost** med partnerji in sistemi (ERP/WMS).

Preslikava »pomen → struktura«

Semantični BT-ji se preslikajo v konkretne XML elemente. Primeri (DESADV – dobavnica):

Tabela 10

Semantični izraz	Pomen	Tipična vrednost	XML (primer)
BT-3	Številka dobavnice	587441	<D_1004>
BT-6	Datum odpreme	20180915	<D_2380>
BT-58	GTIN artikla	3831234567890	<D_7140>
BT-63	Vrednost postavke	152.35	<D_5118>

Izsek (DESADV):

```
<S_BGM>
  <C_C002>
    <D_1001>351</D_1001> <!-- Vrsta dokumenta: DESADV -->
  </C_C002>
  <C_C106>
    <D_1004>587441</D_1004> <!-- BT-3: Številka dobavnice -->
  </C_C106>
</S_BGM>

<S_DTM>
  <C_C507>
    <D_2005>11</D_2005>
    <D_2380>20180915</D_2380> <!-- BT-6: Datum odpreme -->
```

```
    </C_C507>
</S_DTM>
```

Zakaj dve plasti?

- Semantika zagotovi, da vsi partnerji **razumejo isto stvar enako**.
- Sintaksa zagotovi, da sistemi **pravilno izmenjajo in preverijo** isto stvar.



Pomni!

Semantični model določa **pravila poslovnega pomena** (kaj je obvezno, kako se računa in kakšen format je pravilen). Tudi če XML formalno »velja«, dokument **ni uporaben**, če semantična pravila niso izpolnjena (npr. manjka obvezni BT). Vedno preverjaj **oboje**: semantiko **in** sintakso.



Primer v praksi

Naročnik pošlje DESADV z datumom odpreme v napačnem formatu (npr. »15/09/2018«).

- **Semantika**: datum je obvezen (BT-6) in mora biti yyyymmdd.
- **Sintaksa** (XSD) zavrne dokument, ker <D_2380> ne ustreza vzorcu. Rezultat: **avtomatska validacija** prepreči vnos napačnega podatka v WMS/ERP in prihrani ročne popravke.

Povzetek

- **Semantika (BT/BG)** predstavlja pomen in poslovna pravila.
- **Sintaksa (XML/XSD)** pomeni zapis in strojno validacijo.
- Skupaj omogočata **nedvoumno razumevanje** in **avtomatizirano obdelavo** eSLOG dokumentov med partnerji.

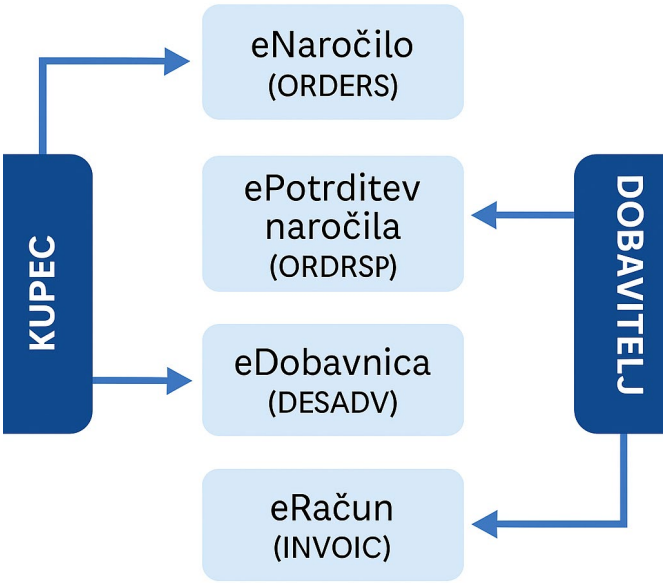
3.7.4 Procesni tok eSLOG v logistiki

Cilj procesa: od naročila do obračuna brez ročnega pretipkavanja, s sledljivostjo vsakega koraka v ERP/WMS in pri partnerjih.

A. Osnovni tok (»happy path«)

1. **eNaročilo (ORDERS)** – *Naročnik* → *Dobavitelj*
   - Vsebuje: številko naročila, partnerje (GLN), roke, lokacije dostave, postavke (GTIN), cene in pogoje.
   - ERP dobavitelja samodejno prevzame naročilo in odpre prodajno naročilo.
2. **ePotrditev naročila (ORDRSP)** – *Dobavitelj* → *Naročnik*
   - Status: »**Accepted**« (potrjeno), »**Accepted with changes**« (delne spremembe), »**Rejected**« (zavrnitev).
   - Namen: uskladitev količin, cen in rokov. Naročnik posodobi svoje nabavno naročilo.
3. **eDobavnica (DESADV)** – *Dobavitelj* → *Naročnik*
   - Ob odpremi blaga.
   - Vsebuje: številko dobavnice, povezavo na naročilo, **SSCC** oznake paketov/palet, prevoznika, logistične enote, serije/roks.
   - WMS naročnika pripravi **ASN prejem**: črtne kode → skeniranje → avtomatski prevzem.
4. **eRačun (INVOIC)** – *Dobavitelj* → *Naročnik*
   - Po fizični dobavi (ali po pogodbenem pravilu).
   - Sklici: številka naročila, dobavnice, pogodbe.
   - ERP naročnika izvede **tridimenzionalno ujemanje** (*3-way match*: naročilo ↔ dobavnica ↔ račun) in pripravi plačilo.

Rezultat je krajši cikel od naročila do plačila, manj napak, popolna sledljivost.



Slika 2: proces izmenjave eSLOG dokumentov.
Vir: Gospodarska zbornica Slovenije, 2020a.

B. Vloge sistemov

- **ERP naročnika:** priprava ORDERS, kontrola potrditve, 3-way match, knjiženje.
- **ERP dobavitelja:** sprejem ORDERS, izdaja ORDRSP, izdaja DESADV in INVOIC.
- **WMS naročnika:** razporeditev dostav, ASN prejem po DESADV, skeniranje SSCC/GTIN, količinski in kakovostni prevzem.
- **TMS/prevoznik:** prevzem podatkov iz DESADV (časi, vozilo, lokacije).

C. Povezave (ID-ji in sklici)

Tabela 11

Dokument	Primarni ID	Sklici naprej/ nazaj	Zakaj je pomembno?
ORDERS	Št. naročila	–	Referenca za vse naslednje korake.
ORDRSP	Št. potrditve	Sklic na št. naročila	Uskladitev obsega/ rokov.
DESADV	Št. dobavnice	Sklic na št. naročila; palete = SSCC	ASN prejem in sledljivost logističnih enot.
INVOIC	Št. računa	Sklic na naročilo in/ali dobavnico	3-way match in plačilo.

Če manjkajo sklici, avtomatika odpove (ročni poseg, zastoja).

D. Kontrole kakovosti (pred/med/po)

- **Pred odpošiljanjem:** ORDRSP mora uskladiti vse spremembe (količine, cene in datume).
- **Ob odpremi:** DESADV mora vsebovati dejanske količine, serije, SSCC; WMS naročnika uporablja ASN za pripravo prevzema.
- **Ob obračunu:** 3-way match preveri ujemanje (količina, cena in sklici). V primeru razlik pride do postopka reklamacije/ dobropisa.

E. Odstopanja in robni primeri

- **Delna dobava / backorder:**
  - ORDRSP: *Accepted with changes* (spremenjena količina ali rok).
  - DESADV: pošlje se za dobavljeni del; preostanek ostane odprt.
  - INVOIC: zaračuna se samo dobavljeno.
- **Zamenjava artikla / substitucija:**
  - ORDRSP predlaga nadomestni GTIN, ki ga naročnik potrdi ali zavrne.
  - DESADV in INVOIC morata odražati dejansko dobavljeno blago.
- **Zavrnjena pošiljka / neujemanje pri prevzemu:**
  - WMS zapiše odstopanje (short/over/damaged); ERP sproži DN (debit note) ali zahtevek za dobropis.

  - Dobavitelj izda **storno** ali **dobropis** (e-račun).
- **Časovne izjeme (just-in-time):**
  - DESADV se pošlje pred fizičnim prihodom; ASN omogoči hiter ramp-in.
  - Če je DESADV prepozen, skladišče izgubi prednost ASN.



Pomni!

Tehnični prenos (»ACK« na nivoju omrežja) **ni** enako poslovni potrditvi.

- Omrežje lahko potrdi dostavo datoteke, a pravi poslovni status vedno nosi **ORDRSP** (sprejeto/spremenjeno/zavrnjeno).
- Pošiljatelj naj DESADV in INVOIC vedno veže na **isto št. naročila** in, kjer je relevantno, na **št. dobavnice** – tako 3-way match deluje brez ročnih popravkov.



Primer v praksi

Trgovec naroči 1.000 kosov (ORDERS). Dobavitelj lahko dobavi 900 kosov zdaj, 100 čez 5 dni.

- Pošlje **ORDRSP – Accepted with changes** (900 kosov takoj, 100 kasneje).
- Ob odpremi 900 kosov: **DESADV** z natančnimi SSCC in serijami. Skladišče jih skenira in samodejno prevzame.
- Dobavitelj izda **INVOIC** za 900 kosov; 3-way match se izvede brez napak.
- Ko je na voljo preostanek, sledi drugi krog: DESADV + INVOIC za 100 kosov.

Povzetek

- **ORDERS → ORDRSP → DESADV → INVOIC** je standardni eSLOG tok.
- Ključ do avtomatizacije so **dosledni sklici, pravočasni dokumenti** in **ASN prevzem** (DESADV + SSCC).
- Ob odstopanjih proces predvidi delne dobave, substitucije in dobropise – brez izgube sledljivosti ali ročnega pretipkavanja.

3.7.5 Prikaz XML dokumenta DESADV

Spodaj je izsek iz primerne eDobavnice (DESADV):

```
<S_BGM>
    <C_C002>
        <D_1001>351</D_1001> <!-- Vrsta dobavnice -->
    </C_C002>
    <C_C106>
        <D_1004>587441</D_1004> <!-- Številka dobavnice -->
    </C_C106>
    <D_1225>9</D_1225> <!-- Vloga dokumenta -->
</S_BGM>
<S_DTM>
    <C_C507>
        <D_2005>11</D_2005>
        <D_2380>20180915</D_2380> <!-- Datum odpreme -->
    </C_C507>
</S_DTM>
<S_NAD>
    <D_3035>BY</D_3035> <!-- Kupec -->
    <C_C080>
        <D_3036>Kupec d.o.o.</D_3036>
    </C_C080>
</S_NAD>
```

3.7.6 Prikaz dokumenta v prijazni obliki (XSLT)

Za boljšo preglednost se lahko uporabi **XSLT datoteka**, ki XML dokument pretvori v HTML tabelo, berljivo v brskalniku.

Primeri takšnih pretvorb se pogosto uporabljajo pri validaciji eSLOG datotek ali pri izobraževanju.

Koraki:

1. XML dokument odpri v Notepad++ in ga poravnaj z *Pretty Print*.
2. Uporabi XSLT za pretvorbo v preglednico.
3. Odpri XML v brskalniku → prikaže se tabelarični pogled dokumenta.



3.7.7 Vaje

Vaja 1: Odpri in preglej XML

1. Odpri datoteko sample_eSLOG20_DESADV_v200.xml v Notepad++.
2. Namesti vtičnik **XML Tools**.

3. Uporabi *Pretty Print (XML only)* za lepši prikaz.
4. Identificiraj osnovne elemente (glava, datumi, kupec in dobavitelj).

Vaja 2: Validacija XML proti XSD

1. Odpri isto XML datoteko.
2. V meniju XML Tools izberi **Validate now**.
3. Poveži datoteko eSLOG20_DESADV_v200.xsd.
4. Analiziraj sporočilo o uspešni validaciji ali napakah.

Vaja 3: Prikaz XML v brskalniku

1. Shrani XSLT datoteko v isti imenik kot XML.
2. Dodaj v glavo XML povezavo:

```
<?xml-stylesheet type="text/xsl" href="style.xsl"?>
```

3. Odpri XML z dvojnim klikom v Chrome/Edge/Firefox.
4. Analiziraj prikazani dokument.

Povzetek

– eSLOG je standard, ki omogoča strukturirano elektronsko poslovanje med podjetji in javnim sektorjem.
– Uporablja XML in XSD za prenos dokumentov, kot so naročilo, potrditev naročila, dobavnica in račun.
– Pravilna uporaba standarda povečuje učinkovitost logističnih procesov, zmanjšuje napake in omogoča interoperabilnost.
– DESADV dokument je jedrni del dobavnega procesa v eSLOG in se pogosto uporablja v kombinaciji z ERP sistemi.


MSKU
MAERSK
maerskline.com

Elektronsko poslovanje v logistiki

Gradivo za vaje

ŠIFRIRANJE

Varuj podatke.
Zagotovi zaupanje.

CIA – temelj informacijske varnosti

ZAUPNOST
Podatki so dostopni le pooblaščenim osebam.

CELOVITOST
Podatki ostanejo točni in nespremenjeni.

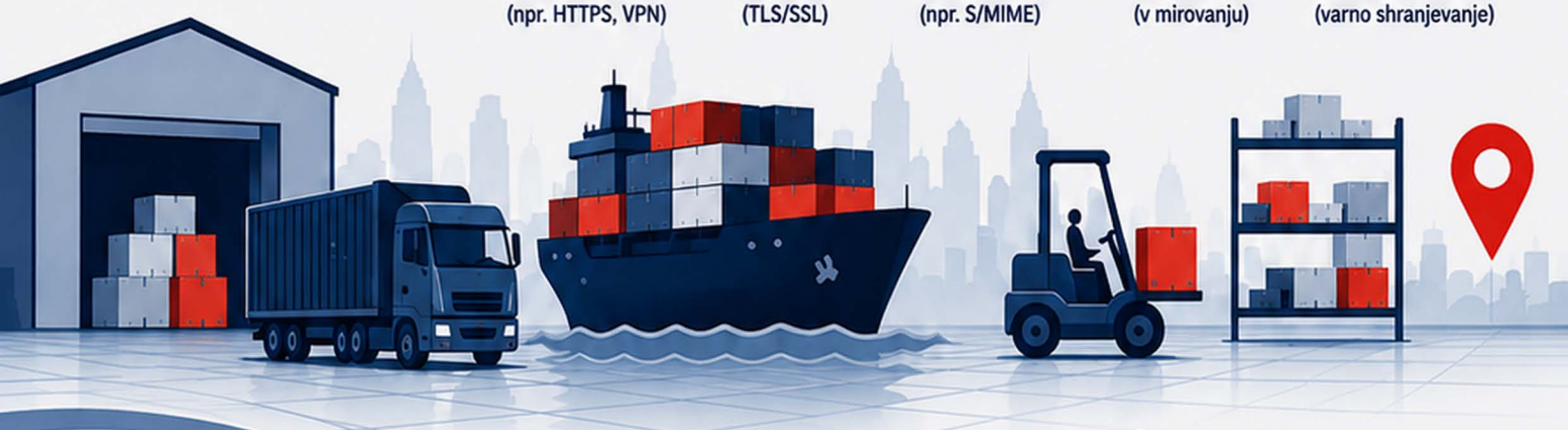
RAZPOLOŽLJIVOST
Pooblaščeni uporabniki imajo dostop, kadar ga potrebujejo.



Izvorno besedilo → ŠIFRIRANJE → Šifrirano besedilo



VARNI PRENOSI (npr. HTTPS, VPN) · PROTOKOLI (TLS/SSL) · ZAŠČITA E-POŠTE (npr. S/MIME) · ZAŠČITA PODATKOV (v mirovanju) · OBLAČNE STORITVE (varno shranjevanje)



Univerzitetna založba
Univerze v Mariboru

4 Šifriranje (encryption)

Šifriranje je osnovni mehanizem varovanja podatkov pri prenosu in shranjevanju: iz berljive vsebine ustvari neberljiv zapis, ki ga lahko v prvotno obliko povrne le imetnik pravega ključa. V tem poglavju na kratko razložimo, kaj šifriranje je, zakaj ga potrebujemo in kako podpira glavne varnostne cilje (zaupnost, celovitost in razpoložljivost). Nato pregledamo osnovne pojme (golo besedilo, šifrirano besedilo, ključe in algoritme), predstavimo simetrično, asimetrično in hibridno šifriranje ter pokažemo, kje jih srečamo v praksi (HTTPS, VPN, e-pošta, šifriranje diskov). Dodamo tudi pogoste napake pri rabi, priporočila za varno konfiguracijo in povezane mehanizme (hash, HMAC, digitalni podpis in PKI), da se teoretični koncepti lahko neposredno prenesejo v varno vsakodnevno delo.



Pomni!

Šifriranje samo po sebi rešuje predvsem zaupnost. Če želiš preveriti tudi nespremenjenost in izvor podatkov, ga vedno kombiniraj z zgoščevalnimi funkcijami/HMAC ali digitalnim podpisom (PKI/PGP).



Primer v praksi

Pri oddaji obrazca na spletni banki brskalnik najprej s pomočjo TLS izvede izmenjavo ključev (asimetrično), nato se ves promet šifrira s hitrim simetričnim algoritmom (npr. AES). Rezultat: tretje osebe ne vidijo vsebine, strežnik in uporabnik pa lahko preverita pristnost povezave.

4.1 Uvod

Šifriranje (ang. encryption) je postopek, s katerim berljive podatke (t. i. **golo besedilo**) pretvorimo v neberljivo obliko (t. i. **šifrirano besedilo**) s pomočjo posebnega **algoritma** in **ključev** (National Institute of Standards and Technology [NIST], 2017). Namen tega postopka je zaščititi podatke pred nepooblaščenim dostopom med shranjevanjem ali prenosom.

Ko je sporočilo šifrirano, ga lahko prebere samo tisti, ki ima ustrezen **dešifrirni ključ**, s katerim šifrirano besedilo ponovno pretvori v prvotno obliko.

Šifriranje je temeljni gradnik **kibernetske varnosti**, saj zagotavlja varovanje podatkov v digitalnem okolju. Brez šifriranja bi bilo mogoče prestreči gesla, osebne podatke, finančne transakcije ali vojaške komunikacije z enostavnim poslušanjem omrežnega prometa.

Glavni razlogi za uporabo šifriranja so:

- **zaupnost**, ki varuje podatke pred nepooblaščenimi osebami,
- **celovitost**, ki preprečuje, da bi nekdo podatke neopazno spremenil.
- **preverjanje istovetnosti**, ki omogoča preverjanje, kdo je pošiljatelj sporočila (npr. z digitalnimi podpisi),
- **varna komunikacija**, ki omogoča zaščito prenosa podatkov prek interneta (npr. HTTPS),
- **zaščita shranjenih podatkov**, ki ščiti podatke na diskih, USB-ključih, strežnikih ali v oblačnih storitvah,
- **varovanje državnih in vojaških skrivnosti** saj zagotavlja varnost občutljivih informacij v obrambnem in varnostnem sektorju.

Preprosto povedano: šifriranje deluje kot "digitalni sef" – tudi če nekdo prestreže sporočilo ali dostopa do datoteke, brez pravilnega ključa vsebine ne more prebrati.

Primer šifriranja

Predstavljaj si, da želi Ana poslati Borutu sporočilo:

Golo besedilo (plaintext): POZDRAV

Da bi zagotovila, da nihče nepooblaščen ne bo razumel vsebine, uporabi **preprosto Cezarjevo šifro** s **ključem 3**.

To pomeni, da vsako črko premakne za 3 mesta naprej v abecedi:

- P → Š
- O → S
- Z → B
- D → G
- R → T
- A → Č
- V → A

Šifrirano besedilo (ciphertext): ŠSBGTČA

Ko Borut prejme sporočilo ŠSBGTČA, ga **dešifrira** z enakim ključem (premik nazaj za 3 črke) in dobi izvirno sporočilo **POZDRAV**.

Zakaj je ta primer pomemben?

– Prikazuje osnovno logiko šifriranja: pretvorba berljivega sporočila v neberljivo obliko.
– Kaže pomen ključa: brez znanja o zamiku 3 je težko pravilno prebrati sporočilo.
– Čeprav je Cezarjeva šifra danes zelo enostavna in ranljiva, lepo ponazori osnovno idejo sodobnih šifrirnih algoritmov (npr. AES, RSA), ki uporabljajo veliko bolj zapletene matematične postopke.

V praksi se namesto premikanja črk uporabljajo močni matematični algoritmi in dolgi ključi, kar zagotavlja visoko raven varnosti.

4.1.1 Povezava med šifriranjem in varnostnimi cilji (zaupnost, celovitost, razpoložljivost – CIA)

V kibernetski varnosti se pogosto uporablja **model CIA** (ang. »*Confidentiality, Integrity, Availability*«), ki opisuje tri temeljne varnostne cilje.

Šifriranje ima pri tem pomembno vlogo, saj neposredno podpira **zaupnost** in **celovitost**, posredno pa vpliva tudi na **razpoložljivost**.

Tabela 12

Varnostni cilj	Pomen	Vloga šifriranja
Zaupnost (ang. Confidentiality)	Zagotavlja, da lahko podatke berejo samo pooblaščene osebe.	Šifriranje podatkov preprečuje, da bi nepooblaščene osebe razumele vsebino, tudi če do nje pridejo (npr. prestrežen e-mail, ukradena datoteka).
Celovitost (ang. Integrity)	Zagotavlja, da podatki niso bili spremenjeni ali poškodovani brez zaznave.	Digitalni podpisi in kriptografske kontrolne vsote (npr. HMAC) omogočajo preverjanje, ali je vsebina avtentična in nespremenjena.
Razpoložljivost (ang. Availability)	Zagotavlja, da so podatki in sistemi dostopni pooblaščenim uporabnikom, ko jih ti potrebujejo.	Čeprav šifriranje ni primarni mehanizem za razpoložljivost, k njej posredno prispeva: zaščiteni podatki so varnejši pred krajo, sabotažo ali izsiljevalskimi napadi (npr. »ransomware«), kar pomaga ohranjati neprekinjeno delovanje sistemov.

Povzetek:

- **Šifriranje** predstavlja **varovanje zaupnosti** podatkov med prenosom in shranjevanjem.
- **Digitalni podpisi in zgoščevalne funkcije** pomenijo **varovanje celovitosti** podatkov.
- **Posredna zaščita razpoložljivosti** izhaja iz zmanjšanja možnosti zlorab ali nepooblaščenih sprememb.

Brez šifriranja bi bilo zagotavljanje teh treh varnostnih ciljev v sodobnem digitalnem okolju praktično nemogoče.

4.1.2 Zgodovinski razvoj šifriranja

Šifriranje ima dolgo in zanimivo zgodovino, ki sega več tisoč let nazaj. Sprva je bilo namenjeno predvsem **skrivanju vojaških in diplomatskih sporočil**, danes pa je nepogrešljiv del vsakdanjega digitalnega sveta — od spletnega bančništva do zaščite pametnih telefonov.

Razvoj šifriranja lahko razdelimo v **štiri glavna obdobja**, ki so predstavljena v naslednjih štirih podpoglavjih.

4.1.2.1 Antično obdobje – rojstvo šifriranja

V starih civilizacijah so uporabljali preproste metode skrivanja sporočil:

- **Cezarjeva šifra** (ok. 50 pr. n. št.):
  premik za 3), da je zaščitil vojaška sporočila (Encyclopaedia Britannica, 2025a). Primer: »POZDRAV« → (premik +3) → »SRCGUDY«
- **Skital (gr. »*scytale*«)**:
  Špartanci so okoli lesenega valja navili trak pergamenta, nanj zapisali sporočilo in trak odvili. Le enak valj je omogočal pravilno branje.
- **Hebrejski kod Atbaš**:
  Preprosta šifra, pri kateri se prva črka zamenja z zadnjo, druga z predzadnjo itd.

Te metode so bile enostavne, a učinkovite za svoj čas, saj je bilo branje šifriranih sporočil brez poznavanja ključa zelo težavno.

4.1.2.2 Renesansa in predmoderno obdobje – izumi in izboljšave

V tem obdobju se pojavijo naslednji bolj zapleteni postopki:

- **Polialfabetske šifre**:
  Najbolj znana je **Vigenèrova šifra** (iz leta 1553), ki uporablja zaporedje različnih premikov glede na ključ. Dolgo časa je veljala za »nezlomljivo« (Encyclopaedia Britannica, 2025c).
- **Kodirne knjige in ročni sistemi**:
  V evropskih kraljevih dvorih in diplomaciji so se uporabljali posebni simboli in tabele za skrivanje sporočil.

– Prvi **kriptografski priročniki** in znanstvene razprave so postavili temelje za sistematičen razvoj šifriranja.

4.1.2.3 19. in 20. stoletje – mehanične in vojaške šifre

S tehnološkim napredkom se šifriranje preseli na nove naprave:

- **Mehanični šifrirni stroji**:
  Najbolj znan šifrirni stroj je **Enigma** in ga je uporabljala nacistična Nemčija med drugo svetovno vojno. Naprava je z rotacijskimi kolesi ustvarjala kompleksne kombinacije črk (Encyclopaedia Britannica, 2025b).
  Britanski in poljski kriptografi (med njimi Alan Turing) so razvozlali šifrirni sistem Enigme in lahko dešifrirali prestrežena sporočila, kar je po ocenah zgodovinarjev bistveno skrajšalo vojno.
- **SIGABA (ZDA)** in drugi stroji:
  ZDA in zavezniki so razvili še naprednejše naprave, katerih šifriranje je bilo bistveno težje razvozlati – šifre stroja SIGABA nasprotnikom med vojno ni uspelo nikoli dešifrirati (Mucklow, 2015).

V tem obdobju je šifriranje postalo strateško orožje, odločilno za uspeh vojaških operacij.

4.1.2.4 Digitalno obdobje – računalniška kriptografija

Teoretične temelje sodobne kriptografije je po vojni postavil Shannon (1949) s teorijo tajnosti komunikacij. Z razvojem računalnikov po drugi svetovni vojni se je začelo moderno obdobje šifriranja:

- **1970–1980**:
  - Uporaba **DES** (ang. **Data Encryption Standard«**) kot prvega široko uporabljenega standarda simetričnega šifriranja.
  - Razvoj **asimetrične kriptografije** (Diffie–Hellman, RSA), ki je omogočila javne in zasebne ključe (Diffie in Hellman, 1976).
- **1990–2000**:
  - Vzpon interneta je zahteval varno komunikacijo in s tem tudi razvoj protokolov, kot npr. **SSL/TLS**, in standardov, kot npr. **AES** (ang. **Advanced Encryption Standard**) (NIST, 2001; Rescorla, 2018).
  - Kriptografija postane vsakodnevno orodje v komercialnih in osebnih napravah.

- **2000–danes**:
  - Napredek je v **eliptičnih krivuljah (ECC)**, ki omogočajo varnost z manjšimi ključi.
  - Razmah mobilnih naprav, oblačnih storitev in šifriranja celotnih diskov.
  - **Kvantna kriptografija** in **post-kvantni algoritmi** postajajo, zaradi prihodnjih kvantnih računalnikov, nova meja.

Povzetek

- **Antika**: preproste ročne šifre za vojaška sporočila.
- **Renesansa**: polialfabetske metode, diplomacija.
- **20. stoletje**: mehanični stroji, vojaška kriptografija, odločilen pomen v vojnah.
- **Digitalna doba**: računalniška kriptografija, standardizacija, internet, kvantni izzivi.

Brez njega bi bili spletne transakcije, komunikacije in podatki popolnoma izpostavljeni prisluškovanju, krajam in zlorabam.

4.2 Osnovni pojmi

Da bi razumeli, kako deluje šifriranje, moramo najprej poznati nekaj temeljnih pojmov. Ti pojmi tvorijo osnovo vsakega šifrirnega postopka, ne glede na to, ali gre za preprosto Cezarjevo šifro ali napredne sodobne algoritme, kot sta AES ali RSA.

Tabela 13

Pojem	Pojasnilo
Golo besedilo (ang. plaintext)	To je izvirno sporočilo ali podatek v berljivi obliki, ki ga želimo zaščititi. Golo besedilo so lahko besede, številke, slike, datoteke ali katera koli druga oblika podatkov.
	Primer: POZDRAV, osebni podatki v dokumentu, številka kreditne kartice.
Šifrirano besedilo (ang. ciphertext)	Rezultat postopka šifriranja. Gre za preoblikovano besedilo, ki je brez ustreznega ključa neberljivo in neuporabno. Namen šifriranja je prav v tem, da prestreženo šifrirano besedilo ne razkrije vsebine.
	Primer: SRCGUDY (če uporabimo Cezarjevo šifro s ključem 3).
Šifrirni ključ (ang. encryption key)	To je skrivnostni ali javni podatek, ki določa, kako bo algoritem izvedel šifriranje in dešifriranje. Ključ je osrednji varnostni element — brez njega šifrirano besedilo ni mogoče pravilno prebrati.
	Primer: pri Cezarjevi šifri je ključ število zamika (npr. 3); pri AES je to dolga naključna vrednost (npr. 256 bitov).
Algoritem (ang. algorithm)	Matematično pravilo ali postopek, ki določa način šifriranja in dešifriranja. Algoritem sam po sebi ni skrivnost — varnost temelji na ključu, ne na skrivanju algoritma (t. i. »Kerckhoffovo načelo«).
	Primer: AES, RSA, ChaCha20, Vigenère.
Dešifriranje (decryption)	Je obraten postopek od šifriranja, saj se šifrirano besedilo s pomočjo pravega ključa pretvori nazaj v prvotno golo besedilo. Brez pravilnega ključa dešifriranje ni mogoče.
	Primer: SRCGUDY → (z zamikom -3) → POZDRAV.

Pomembno za zapomniti:

- Algoritem je kot **"zaklepni mehanizem"**, ključ je **"ključavnica"**, golo besedilo je **"sporočilo v sefu"**, šifrirano besedilo pa je **"zaklenjen sef"**.
- Če je ključ izgubljen, do vsebine pogosto ni več mogoče dostopati – šifriranje je varno tudi za lastnika, če ta ne hrani ključa pravilno.
- Moč šifriranja je odvisna od dolžine in varnosti ključa, ne od zapletenosti algoritma samega.

4.3 Vrste šifriranja

Šifriranje lahko razdelimo glede na način uporabe ključev in namen v **tri osnovne vrste**:

1. **simetrično šifriranje,**
2. **asimetrično šifriranje,**
3. **hibridni sistemi.**

Vsaka vrsta ima svoje prednosti, omejitve in značilna področja uporabe. Dobro razumevanje razlik med njimi omogoča pravilno izbiro varnostnih rešitev v praksi.

4.3.1 Simetrično šifriranje

Pri **simetričnem šifriranju** uporabljata pošiljatelj in prejemnik **isti skrivni ključ** za šifriranje in dešifriranje podatkov.

To je najstarejša in najhitrejša oblika šifriranja, ki je primerna za zaščito **večjih količin podatkov**.

Kako poteka postopek?

Postopek je naslednji:

1. Pošiljatelj izbere skrivni ključ.
2. Podatke šifrira s tem ključem.
3. Prejemnik mora imeti popolnoma enak ključ, da lahko podatke dešifrira.
4. Če skrivni ključ pride v roke tretji osebi, je varnost sporočila ogrožena.

Primer (Cezarjeva šifra s ključem 3):

Golo besedilo: POZDRAV

→ šifrirano besedilo: ŠSBGTČA
→ dešifriranje z istim ključem: POZDRAV

Značilnosti simetričnega šifriranja:

- **Prednosti** so:
  - zelo hitro in učinkovito, tudi pri velikih datotekah,
  - nizka poraba računalniških virov in
  - enostavna implementacija.

- **Omejitve** so:

  - varna izmenjava skrivnega ključa med uporabniki je zahtevna.
  - če isti ključ uporablja več oseb ali sistemov, obstaja večje tveganje za zlorabe,
  - mi primerno za odprta okolja, kjer ni mogoče zagotoviti varnega prenosa ključa.

Tabela 14: Pogosto uporabljeni algoritmi

Algoritem	Opis	Uporaba
AES (ang. *Advanced Encryption Standard*)	Je standardiziran algoritem s 128-, 192- ali 256-bitnimi ključi; zelo varen in hiter.	HTTPS, VPN, šifriranje diskov, aplikacije.
ChaCha20	Je sodobna alternativa AES, hitra tudi na mobilnih napravah, dobra odpornost na stranske napade.	Mobilne aplikacije, TLS, WireGuard VPN.
3DES (ang. *Triple DES*)	Ja nadgradnja DES s trojnim šifriranjem, ki velja danes za zastarelega.	Starejši sistemi, postopno opuščanje.
Blowfish / Twofish	Sta algoritma za hitro šifriranje s spremenljivo dolžino ključa.	VPN, vdelani sistemi, nekatere datotečne rešitve.

4.3.2 Asimetrično šifriranje

Pri **asimetričnem šifriranju** uporabljamo **par matematično povezanih ključev**:

- **javni ključ**, ki je dostopen vsem in se uporablja za šifriranje.
- **zasebni ključ**, ki ostane v lasti uporabnika in se uporablja za dešifriranje.

Tak pristop omogoča varno izmenjavo informacij **brez predhodnega dogovora o skupnem skrivnem ključu**, kar je posebej primerno za odprta okolja, kot je internet.

Kako poteka postopek?

Postopek je naslednji:

1. prejemnik ustvari par javnega in zasebnega ključa,
2. javni ključ posreduje pošiljatelju,
3. pošiljatelj z javnim ključem šifrira sporočilo,
4. prejemnik ga dešifrira z zasebnim ključem,
5. tudi če nekdo prestreže javni ključ, ne more prebrati sporočila.

Značilnosti asimetričnega šifriranja:

- **Prednosti so:**
  - ni potrebe po skrivni izmenjavi ključa,
  - omogoča preverjanje istovetnosti pošiljatelja (npr. z digitalnimi podpisi),
  - primerno za odprta omrežja in spletne storitve.
- **Omejitve so:**
  - počasnejše in bolj zahtevno za procesor v primerjavi s simetričnim šifriranjem,
  - ni učinkovito za šifriranje velikih količin podatkov.

Tabela 15: Pogosto uporabljeni algoritmi

Algoritem	Opis	Uporaba
RSA	Temelji na faktorizaciji velikih števil; najpogosteje uporabljen za šifriranje in digitalne podpise.	HTTPS, digitalni certifikati, PGP
Diffie–Hellman	Omogoča varno izmenjavo simetričnega ključa prek nezavarovanega kanala.	VPN, TLS, protokoli za izmenjavo ključev
ECC (ang. *Elliptic Curve Cryptography*)	Uporablja matematične lastnosti eliptičnih krivulj; zagotavlja enako stopnjo varnosti z manjšimi ključi.	Mobilne naprave, TLS, elektronski podpisi
ElGamal	Temelji na diskretnem logaritmu; uporablja se za šifriranje in podpise.	Kriptografske knjižnice, digitalna identiteta

Primer uporabe:

- Ana ustvari par javnega in zasebnega ključa.
- Borut želi Ani poslati zaupno sporočilo.

– Uporabi Anin javni ključ za šifriranje.
– Le Ana lahko sporočilo prebere, saj ima zasebni ključ.

4.3.3 Hibridni sistemi

V praksi se pogosto uporablja **kombinacija simetričnega in asimetričnega šifriranja**, ki jo imenujemo **hibridni sistemi**.

Tak pristop združuje prednosti obeh metod:

– hitro šifriranje vsebine s simetričnim algoritmom in
– varno posredovanje simetričnega ključa prek asimetričnega algoritma.

Kako poteka postopek?

Postopek je tak:

1. Pošiljatelj ustvari naključni simetrični ključ za šifriranje vsebine.
2. Podatke šifrira s tem ključem (npr. z AES).
3. Simetrični ključ šifrira z javnim ključem prejemnika (npr. RSA).
4. Prejemnik z zasebnim ključem najprej dešifrira simetrični ključ, nato z njim vsebino.

Prednosti hibridnih sistemov so:

– hitro šifriranje podatkov v kombinaciji z varno izmenjavo ključev,
– primeren pristop za velike količine podatkov,
– standardna rešitev v sodobnih komunikacijskih protokolih, kot so HTTPS ali VPN.

Tabela 16: Primeri uporabe

Protokol / aplikacija	Mehanizmi	Opis
HTTPS / TLS	RSA/ECC + AES/ChaCha20	Je asimetrični mehanizem za izmenjavo simetričnega ključa, nato sledi simetrično šifriranje celotne seje.
VPN (npr. OpenVPN, WireGuard)	Diffie–Hellman/ECC + AES/ChaCha20	Je hibridni pristop za varno vzpostavitev šifriranega tunela.
E-pošta (PGP)	RSA + simetrični algoritem	Javnim ključem se šifrira samo ključ sporočila, vsebina pa s hitrim simetričnim algoritmom.

Tabela 17: Primerjalni pregled vrst šifriranja

Vrsta šifriranja	Število ključev:	Hitrost:	Način posredovanja ključa:	Primeri uporabe
Simetrično	en skrivni ključ	zelo hitro	Zahteva varen prenos.	Šifriranje diskov, podatkov, VPN
Asimetrično	par javni + zasebni ključ	počasnejše	Javni ključ je prosto dostopen.	Digitalni podpisi, avtentikacija, izmenjava ključev
Hibridno	kombinacija obeh prej naštetih	hitro in varno	Javni ključ za varno posredovanje simetričnega ključa.	HTTPS, e-bančništvo, varna e-pošta

Povzetek:

- **Simetrično šifriranje** je hitro in učinkovito, vendar zahteva varen način za posredovanje skrivnega ključa.
- **Asimetrično šifriranje** omogoča varno komunikacijo brez predhodnega dogovora, vendar je počasnejše.
- **Hibridni sistemi** združujejo prednosti obeh in so zato standard v sodobnih omrežnih varnostnih protokolih.

4.4 Najpogostejše napake pri uporabi šifriranja

Pri uporabi šifriranja se v praksi pogosto pojavljajo napake, ki lahko bistveno zmanjšajo njegovo varnostno vrednost. Te napake so posledica neustrezne izbire algoritmov, slabe zaščite ključev ali nepravilne konfiguracije sistemov. Spodaj so navedene najpogostejše težave, ki se jim je mogoče izogniti z upoštevanjem preverjenih varnostnih praks in standardov:

- uporaba zastarelih ali ranljivih algoritmov (npr. DES, 3DES, RC4), ki jih je mogoče razbiti z današnjimi računalniškimi zmogljivostmi,
- izbira prekratkih ali šibkih ključev, ki omogočajo napade z ugibanjem ali izčrpavanjem,
- shranjevanje ali pošiljanje šifrirnih ključev po nezavarovanih kanalih, kar poveča možnost njihovega razkritja,
- napačna konfiguracija varnostnih protokolov (npr. napačni TLS certifikati, neustrezne nastavitve šifrirnih paketov),
- ponovna uporaba istih ključev za daljša obdobja ali za različne namene, kar poveča tveganje kompromitacije,
- neuporaba preverjanja celovitosti in istovetnosti (npr. z digitalnimi podpisi ali HMAC), ko je to potrebno,

- uporaba lastnih, nestandardnih šifrirnih rešitev namesto preverjenih in standardiziranih algoritmov,
- zanemarjanje posodobitev in nadgradnje šifrirnih komponent, zaradi česar sistemi uporabljajo ranljive različice,
- nepravilno upravljanje s šifrirnimi ključi (npr. pomanjkanje rotacije, slaba zaščita, pomanjkanje revizije),
- zanašanje izključno na šifriranje za zaščito podatkov brez dodatnih varnostnih mehanizmov (npr. nadzora dostopa, preverjanja istovetnosti).

4.5 Priporočila za varno uporabo

Šifriranje je učinkovito le, če je uporabljeno pravilno in skladno z uveljavljenimi varnostnimi standardi. Tudi najboljši algoritmi ne morejo zagotoviti zaščite, če so ključi slabo upravljani ali programska orodja zastarela. Zato je pomembno slediti preverjenim priporočilom, ki zmanjšujejo varnostna tveganja in povečujejo zanesljivost celotnega varnostnega sistema. Zato:

- uporabljaj preverjene in standardizirane algoritme (npr. AES-256, RSA-2048 ali bolje), ki so priznani v strokovni skupnosti,
- redno menjaj šifrirne ključe in digitalne certifikate, da zmanjšaš tveganje kompromitacije,
- ne razvijaj lastnih šifrirnih rešitev — uporabi preizkušene in preverjene metode,
- uporabljaj programska orodja, ki so redno posodobljena in vključujejo najnovejše varnostne popravke,

Za učenje in testiranje lahko uporabiš različna brezplačna orodja, kot so:

- **Notepad++** z vtičniki za preizkušanje osnovnih šifer,
- **OpenSSL** za delo s certifikati in preverjanje šifriranja in
- **Gpg4win** za šifriranje in podpisovanje elektronske pošte.

Pri izbiri algoritmov upoštevamo Kerckhoffsovo načelo, po katerem varnost ne sme temeljiti na tajnosti algoritma, temveč na tajnosti ključa (Petitcolas, 2011).

4.6 Dodatni mehanizmi za celovitost in istovetnost

Šifriranje samo po sebi zagotavlja predvsem **zaupnost podatkov** – to pomeni, da nepooblaščene osebe vsebine ne morejo prebrati. Vendar to **ne zadostuje**, če želimo imeti popolno varnost komunikacije in podatkov.

V praksi je enako pomembno zagotoviti tudi:

– da podatki med prenosom ali shranjevanjem **niso bili spremenjeni** (celovitost),
– da lahko **zanesljivo preverimo izvor sporočila** (istovetnost pošiljatelja),
– da pošiljatelj **ne more zanikati** sodelovanja pri komunikaciji (nezanikljivost).

Za to se uporabljajo **dodatni kriptografski mehanizmi**, ki dopolnjujejo šifriranje in skupaj tvorijo celovit varnostni okvir.

4.6.1 Zgoščevalne funkcije (hash funkcije)

Zgoščevalna funkcija (ang. *hash function*) je **matematični postopek**, ki pretvori vhodne podatke poljubne dolžine (besedilo, datoteko, geslo ipd.) v **izhod fiksne dolžine** – t. i. **zgoščeno vrednost** (ang. *hash value*, *digest*) (NIST, 2015).

Hash funkcije se ne uporabljajo za šifriranje, temveč za **preverjanje celovitosti podatkov** in v nekaterih primerih za preverjanje istovetnosti (npr. v digitalnih podpisih ali geselskih bazah).

Osnovni nameni zgoščevalnih funkcij so:

1. **Preverjanje celovitosti** – če se vhodni podatek spremeni (tudi za en sam znak), se izhodna vrednost popolnoma spremeni. Tako lahko hitro ugotovimo, ali je bil podatek spremenjen.
2. **Enosmernost** – iz hash vrednosti **ni mogoče rekonstruirati** izvirnega podatka, zato so primerne za preverjanje brez razkrivanja vsebine (npr. gesel).
3. **Učinkovito preverjanje** – hash ima vedno enako dolžino, ne glede na velikost podatkov, zato je preverjanje hitrejše od primerjave celotnih datotek.
4. **Uporaba v digitalnih podpisih** – namesto, da podpisujemo celoten dokument, podpišemo le hash vrednost, kar je hitrejše in učinkovitejše.

Primer delovanja

Vhodni podatek:
POZDRAV

Hash (SHA-256):
6b8e0c4ccfc7c6dc6eabf7a76f2d6c6768c0c8e9a7e2c1d3b0b9f0b3a5e19d02

Če v besedilo dodamo samo eno črko (»POZDRAVI«), je izhod popolnoma drugačen:
c3c63b69de56bfc03a48d94c391cb5724a4ce6b07cb306bf61f1795c0c2167f5

To vedenje se imenuje **učinek plazu (ang. *avalanche effect*)**, saj majhna sprememba vhoda povzroči popolnoma drugačen izhod.

Glavne lastnosti kriptografskih hash funkcij

Da lahko hash funkcijo uporabljamo v varnostne namene, mora izpolnjevati naslednje v tabeli prikazane zahteve.

Tabela 18

Lastnost	Opis
Deterističnost	Za isti vhod vedno vrne enak izhod.
Enosmernost (ang. *preimage resistance*)	Na podlagi hash vrednosti je praktično nemogoče ugotoviti izvirni vhodni podatek.
Odporna na drugi preimage	Če poznamo en vhod in njegov hash, je izjemno težko najti drug vhod, ki ima enak hash.
Odporna na trke (ang. *collision resistance*)	Izjemno težko je najti dva različna vhoda, ki ustvarita enak hash.
Učinek plazu	Majhna sprememba vhoda povzroči popolnoma drugačen izhod.
Učinkovitost	Funkcija mora biti hitra in primerna za uporabo v realnem času.

Praktične uporabe hash funkcij so naslednje:

1. **Preverjanje integritete datotek**
   - Ob prenosu programske opreme, ISO slik ali varnostnih posodobitev je pogosto priložen hash (npr. SHA-256).
   - Prejemnik izračuna hash svoje kopije in preveri, ali se ujema z objavljenim.
   - Če se vrednosti razlikujeta, je bila datoteka spremenjena ali poškodovana.

2. **Shranjevanje gesel**
   - V sistemih se gesla **ne shranjujejo v čisti obliki**, temveč kot hash vrednosti.
   - Ko se uporabnik prijavi, sistem izračuna hash vnesenega gesla in ga primerja s shranjenim hashom.
   - Tudi če napadalec pridobi bazo, iz hashov ne more enostavno razkriti gesel (še posebej, če je uporabljen sol).
3. **Digitalni podpisi in HMAC**
   - Hash funkcije so del postopka digitalnega podpisovanja: najprej se ustvari hash dokumenta, nato se ta hash podpiše z zasebnim ključem.
   - V HMAC postopku se hash združi s skrivnim ključem za preverjanje celovitosti in istovetnosti.
4. **Preverjanje integritete pri prenosu podatkov**
   - V komunikacijskih protokolih (npr. TLS) se hash funkcije uporabljajo za preverjanje, da podatki med prenosom niso bili spremenjeni.
5. **Digitalni odtisi (fingerprinting)**
   - Hash vrednosti se uporabljajo za identifikacijo datotek, brez primerjave celotne vsebine (npr. v forenzičnih preiskavah, antivirusnih bazah, deduplikaciji podatkov).

Možni napadi na hash funkcije

Če hash funkcija ni dovolj močna, jo lahko napadalci izkoristijo:

Tabela 19

Vrsta napada	Opis
Brute-force napad	Sistematično ugibanje vhoda, dokler se hash ne ujema.
Napad z mavričnimi tabelami (ang. *rainbow tables*)	Vnaprej izračunane tabele hashov za pogosta gesla. Rešitev je uporaba soljenja.
Collision napadi	Napadalec poišče dve različni besedili z enakim hashom. Znani primer: napad na SHA-1 (2005–2017).
Preimage napad	Poskus rekonstruiranja izvirnega vhoda iz hash vrednosti — praktično neizvedljiv za moderne algoritme, a teoretično pomemben.

Zato je izbira **ustrezne in sodobne hash funkcije** izjemno pomembna za varnost sistema.

Table 20: Pregled sodobnih hash funkcij

Ime	Dolžina izhoda	Status	Opombe
MD5	128 bitov	Nevaren, zastarel	Ranljiv na trke; uporaba odsvetovana.
SHA-1	160 bitov	Nevaren, zastarel	Trki dokazani (Google SHAttered, 2017).
SHA-2 (SHA-224/256/384/512)	224–512 bitov	Priporočen	Trenutni standard, široko uporabljen.
SHA-3	224–512 bitov	Priporočen	Naslednik SHA-2, odporen na številne napade.
BLAKE2/BLAKE3	256 ali 512 bitov	Priporočen	Zelo hitre in varne, primerne za mobilne naprave in IoT.

Dodatni varnostni postopki

- **Soljenje (ang. *salting*)** – naključni niz, dodan geslu pred zgoščevanjem. Preprečuje uporabo mavričnih tabel in podaljša čas napadov.
- **Popranje (angl. *peppering*)** – skrivni dodatek (shranjen ločeno od baze), ki dodatno oteži napade.
- **Streščenje (ang. *key stretching*)** – postopki kot PBKDF2, bcrypt ali Argon2 ponavljajo hashiranje več tisočkrat, da napadi postanejo počasnejši.

4.6.1.1 Primer – Izračun zgoščevalne vrednosti

Zgoščevalne funkcije so uporabno orodje za preverjanje, ali se je vsebina podatkov med prenosom ali shranjevanjem spremenila. Z njihovim izračunom dobimo enoličen niz znakov (t. i. zgoščeno vrednost), ki deluje kot digitalni prstni odtis datoteke ali besedila. Že najmanjša sprememba v vhodnih podatkih povzroči popolnoma drugačen rezultat, kar omogoča hitro odkrivanje sprememb.

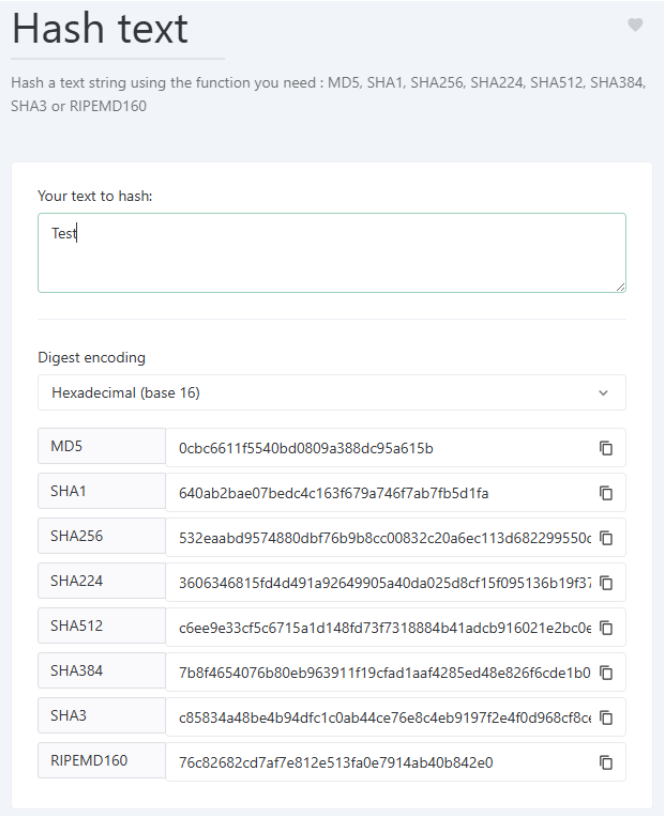
V nadaljevanju je prikazan preprost primer izračuna zgoščevalne vrednosti besedila s pomočjo spletnega orodja, dostopnega na povezavi https://it-tools.tech/hash-text. Takšna orodja so uporabna za hitre preizkuse algoritmov, učenje postopkov in preverjanje rezultatov brez nameščanja dodatnih programov.

Pomni!

Zgoščevalne vrednosti ne omogočajo povratnega pridobivanja izvirnih podatkov. Njihova naloga je preverjanje nespremenjenosti vsebine, ne pa šifriranje ali varovanje zasebnosti.

Primer v praksi

Predstavljaj si, da študent pošlje navodila za laboratorijsko vajo po e-pošti. Pred pošiljanjem izračuna SHA-256 zgoščevalno vrednost datoteke in jo navede v sporočilu. Prejemnik po prenosu datoteke sam izračuna SHA-256 in primerja vrednosti. Če se ujemata, datoteka med prenosom ni bila spremenjena. Če pa se razlikujeta, je datoteka bodisi poškodovana bodisi spremenjena – v tem primeru jo mora prejemnik ponovno pridobiti iz preverjenega vira.



Slika 3: Primer izračuna zgoščevalne vrednosti z orodjem IT-TOOLS.
Vir: lasten, zajem zaslona, IT-TOOLS, b. d.

4.6.1.2 Primer – Razbijanje zgoščevalne funkcije

Čeprav zgoščevalne funkcije same po sebi ne omogočajo povratnega pridobivanja izvirnih podatkov, jih je mogoče v določenih primerih razbiti s pomočjo vnaprej pripravljenih baz podatkov (t. i. »rainbow tables«) ali surovega ugibanja. Spletna orodja, kot je https://crackstation.net, omogočajo hitro preverjanje, ali zgoščena vrednost ustreza kateri izmed pogostih gesel ali znanih vzorcev. Takšna orodja so pogosto uporabljena pri varnostnih testih, saj pokažejo, kako enostavne zgoščevalne vrednosti lahko razkrijejo šibke zaščite.

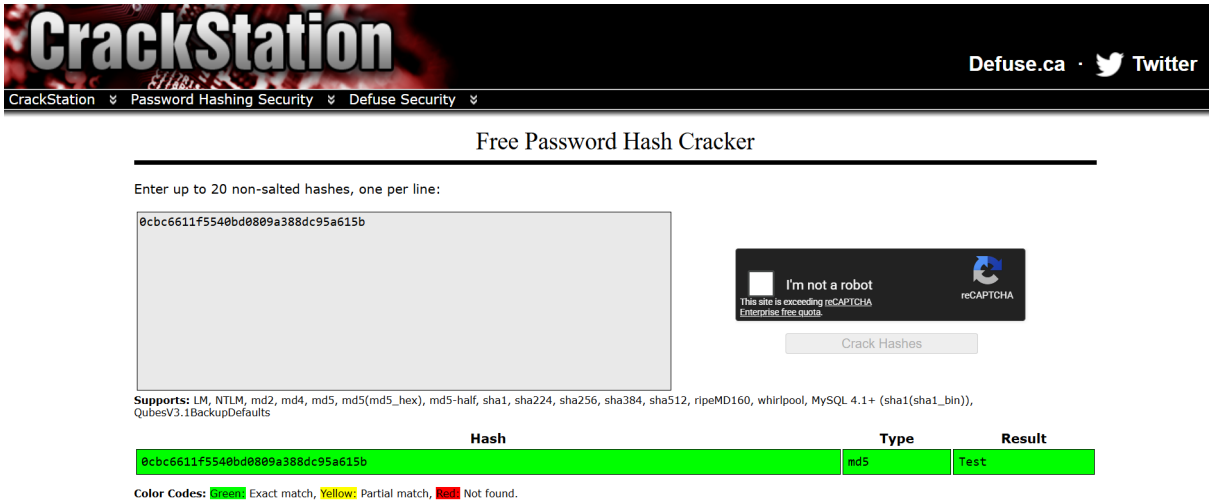
Namen tega primera je prikazati, kako napadalci v praksi razbijajo preproste zgoščene vrednosti, in zakaj je uporaba močnih gesel ter dodatnih zaščitnih mehanizmov (npr. soljenje – salting) nujna za varno shranjevanje podatkov.

Pomni!

Zgoščevalne funkcije niso neprepustna obramba. Če je vhodno geslo preprosto, lahko napadalec z uporabo javno dostopnih orodij hitro ugotovi izvirni niz. Zato morajo biti gesla dolga, nepredvidljiva in po možnosti kombinirana z naključnim dodatkom (soljo), da se prepreči razbijanje s pripravljenimi bazami.

Primer v praksi

Če zgoščeno vrednost npr. 5f4dcc3b5aa765d61d8327deb882cf99 vnesemo v CrackStation, bo orodje takoj prepoznalo, da ustreza geslu "password". To lepo ponazarja nevarnost uporabe enostavnih gesel in potrebo po dodatnih zaščitnih postopkih pri shranjevanju gesel.



Slika 4: Primer razbijanja zgoščevalne vrednosti z uporabo spletnega orodja CrackStation.
Vir: lasten, zajem zaslona, CrackStation, b. d.

4.6.1.3 Vaje

1. Avalanche učinek – praktični preizkus

– Napiši poljubno besedilo in izračunaj hash (SHA-256).

– Spremeni v njem **samo en znak** (npr. dodaj presledek ali piko).
– Ponovno izračunaj hash in primerjaj izhoda.
– Zapiši opazovanje: kako zelo se je hash spremenil?

Cilj vaje:

Praktično prikazati lastnost "avalanche effect" (učinek plazu) pri zgoščevalnih funkcijah – to pomeni, da že zelo majhna sprememba vhoda povzroči popolnoma drugačen hash izhod.

Študent se nauči:

– da so kriptografske hash funkcije zelo občutljive na spremembe,
– da je preverjanje integritete zanesljivo, ker tudi najmanjša sprememba podatka spremeni hash,
– zakaj je hash primeren za preverjanje nespremenjenosti datotek ali sporočil.

2. Prepoznavanje algoritmov po dolžini hash-a

Za naslednje zgoščene vrednosti določi, s katerim algoritmom so bile najverjetneje ustvarjene:

1. e99a18c428cb38d5f260853678922e03
2. f2ca1bb6c7e907d06dafe4687e579fce9
3. f1c9645dbc14efddc7d8a322685f26eb
4. 0beec7b5ea3f0fdbc95d0dd47f3c5bc275da8a33
5. 2c26b46b68ffc68ff99b453c1d30413413422d706483bfa0f98a5e886266e7ae

Namig: Poglej dolžino v heksadecimalnih znakih.

– MD5 = 32 znakov
– SHA-1 = 40 znakov
– SHA-256 = 64 znakov

Cilj vaje:

Naučiti študente prepoznati, kateri hash algoritem je bil najverjetneje uporabljen, na podlagi dolžine hash vrednosti.

Študent se nauči:

- razlikovati med pogostimi algoritmi (MD5, SHA-1, SHA-256) glede na dolžino izhoda,
- razumeti, da ima vsak algoritem značilno dolžino hash-a, ki omogoča hitro identifikacijo,
- uporabljati to znanje pri analizi datotek, preverjanju integritete ali forenzičnem delu, kjer pogosto poznamo le hash, ne pa tudi algoritma.

3. Hashiranje osebnih datotek

- Izberi eno osebno datoteko (npr. .pdf, .jpg, .docx).
- Izračunaj njen SHA-256 hash in ga shrani v .txt.
- Spremeni ime datoteke (vsebino pusti enako) in ponovno izračunaj hash.
- Primerjaj hashe — kaj opaziš?
- Nato spremeni vsebino in primerjaj znova.

Cilj vaje:

Praktično prikazati, da hash funkcije izračunavajo vrednost glede na vsebino datoteke, ne glede na ime, ter da vsaka sprememba vsebine povzroči popolnoma drugačen hash.

Študent se nauči:

- da sprememba imena datoteke ne vpliva na hash vrednost,
- da hash zanesljivo zazna tudi najmanjše spremembe v vsebini datoteke,
- razumeti pomen hashiranja za preverjanje celovitosti datotek pri prenosih, arhiviranju ali forenzični analizi.

4. Primerjava algoritmov

- Za isto kratko besedilo (npr. "Zgoščevalne funkcije") izračunaj MD5, SHA-1, SHA-256 in SHA-3.
- Primerjaj:
  - dolžino hash vrednosti,
  - čas izračuna (če želiš) in
  - razlike v izhodih.
- Zapiši, kateri algoritem je danes varen in zakaj starejših ne uporabljamo več.

Cilj vaje:

Študentom omogočiti primerjavo različnih hash algoritmov glede na dolžino izhoda, hitrost izračuna in varnost, ter razumeti, zakaj se starejši algoritmi danes ne uporabljajo več za varnostne namene.

Študent se nauči:

- prepoznati razlike med MD5, SHA-1, SHA-256 in SHA-3,
- razumeti, da se algoritmi razlikujejo po dolžini izhoda in stopnji varnosti,
- spoznati, da so starejši algoritmi (npr. MD5, SHA-1) ranljivi za napade (npr. trke) in zato neprimerni za sodobno rabo (Stevens idr., 2017; Weaver, 2017),
- znati izbrati primeren algoritem (npr. SHA-256 ali SHA-3) glede na zahteve varnosti in zmogljivosti.

5. Mini "hash forensic" vaja

- Predavatelj (ali sošolec) pripravi **2 skoraj enaki datoteki** (npr. dokumentA.pdf in dokumentB.pdf, ki se razlikujeta v eni besedi).
- Udeleženci izračunajo hash obeh in poskušajo **ugotoviti, katera je original** glede na priložen hash.
- Namen: razumeti, da hash **zanesljivo identificira vsebino**.

Cilj vaje:

Praktično pokazati, da hash vrednost deluje kot enoličen "digitalni prstni odtis" datoteke, ki omogoča zanesljivo prepoznavanje originala med skoraj enakimi različicami.

Študent se nauči:

- uporabiti hash za preverjanje pristnosti datoteke,
- prepoznati, da že najmanjša sprememba (npr. ena beseda) povzroči popolnoma drugačen hash,
- razumeti vlogo hashiranja pri forenzični analizi, preverjanju izvirnosti dokumentov in zaznavanju ponaredkov,
- kritično razmišljati o tem, kako lahko hash uporabimo v realnih primerih preverjanja datotek.

6. Soljenje hashov

– Vzemite enostavno geslo (npr. geslo123) in izračunajte njegov MD5.
– Dodajte na začetek neko "sol" (npr. X9k!) in izračunajte MD5 za X9k!geslo123.
– Primerjajte razlike v hashih in razložite, zakaj je to dobra zaščita pred rainbow tabelami.

Cilj vaje:

Praktično prikazati, kako dodajanje soli spremeni hash vrednost in s tem poveča varnost shranjenih gesel, saj preprečuje učinkovito uporabo vnaprej pripravljenih rainbow tabel.

Študent se nauči:

– da dodajanje soli povzroči popolnoma drugačen hash tudi pri istem geslu,
– razumeti, da sol preprečuje množično razbijanje gesel z enakimi hash vrednostmi,
– spoznati, zakaj je soljenje nujno pri varnem shranjevanju gesel v sistemih,
– razlikovati med "navadnim" hashiranjem gesel in varnim postopkom s soljo.

7. Iskanje izvirnega besedila iz hash vrednosti (rainbow tables)

– Uporabi spletno stran https://crackstation.net.
– V polje prilepi posamezne hash vrednosti:

f4020e91252aafd4b18d8acd17f883db
d3df2c31104b1adf58b31830f2245e5c225f32ff
86e1e99b79d8fe20116bac03b4b55c83
820466a89565d6288970a0a2d763c223
f49eb853c6ef9f8a3d1b2a0572b44b2b

– Klikni "**Crack Hashes**" in preveri, ali orodje razkrije izvirno besedilo (geslo).
– Za vsako hash vrednost zapiši:
  – Ali je bilo besedilo najdeno?
  – Katero besedilo je bilo razkrito?
  – Kateri algoritem je bil uporabljen?

Cilj vaje:

Praktično prikazati, kako enostavne in ne-soljene hash vrednosti lahko razbijemo s pomočjo javno dostopnih rainbow tabel in slovarjev gesel, kar poudari pomen varnih gesel in dodatnih zaščitnih postopkov.

Študent se nauči:

– uporabljati spletna orodja za razbijanje hashov (npr. CrackStation) za analizo enostavnih primerov,
– razumeti, da hash sam po sebi ne zagotavlja varnosti, če je geslo šibko ali predvidljivo,
– prepoznati razliko med hitro razbitimi in varnejšimi hash vrednostmi,
– povezati koncept rainbow tabel z realnimi tveganji pri neustreznem shranjevanju gesel.

8. Poročilo

Za vaje pripravite **poročilo**, ki vsebuje:

– opis postopka,
– uporabljene algoritme,
– rezultate (hash vrednosti) in
– kratek sklep (kaj vaja pokaže in zakaj je pomembna).

4.6.1.4 Povzetek

– Hash funkcije pretvarjajo podatke v fiksno dolg "digitalni odtis".
– Uporabljajo se za preverjanje celovitosti, zaščito gesel, digitalne podpise in preverjanje podatkov pri prenosu.
– Za varnost je bistveno, da funkcija zagotavlja enosmernost, odpornost na trke in dober učinek plazu.
– Starejši algoritmi, kot sta MD5 in SHA-1, niso več varni za sodobne potrebe.
– Kombinacija soljenja, sodobnih hash funkcij in postopkov za upočasnitev napadov zagotavlja zanesljivo zaščito.

4.6.2 HMAC (Hash-based Message Authentication Code)

HMAC je kriptografski mehanizem, ki združuje **zgoščevalno funkcijo (hash)** in **skrivni ključ**, da omogoči preverjanje **celovitosti** in **istovetnosti** sporočil (Krawczyk idr., 1997).

Ne šifrira vsebine, ampak zagotavlja, da:

– sporočilo ni bilo spremenjeno med prenosom,
– pošiljatelj dejansko pozna dogovorjeni skrivni ključ.

HMAC je opredeljen v standardu RFC 2104 in je danes sestavni del številnih varnostnih protokolov, kot so TLS, IPsec, SSH in različni sistemi za preverjanje pristnosti API-jev.

Osnovni namen HMAC

1. **Preverjanje celovitosti** – omogoča zaznavanje, če je bilo sporočilo med prenosom spremenjeno.
2. **Preverjanje istovetnosti pošiljatelja** – samo nekdo, ki pozna skrivni ključ, lahko ustvari veljavno HMAC vrednost.
3. **Zaščita pred ponarejanjem** – preprečuje, da bi napadalec brez poznavanja ključa ustvaril pravilno kodo.
4. **Učinkovitost in prenosljivost** – HMAC uporablja standardne hash funkcije (npr. SHA-256), zato je hiter in ga je mogoče enostavno implementirati na različnih platformah.

Primer delovanja

Predpostavimo:

– sporočilo: HELLO,
– ključ: SECRET in
– hash funkcija: SHA-256.

Postopek izračuna HMAC poteka v dveh fazah: notranje in zunanje hashiranje. Rezultat je 64-znakovna heksadecimalna vrednost, ki se pošlje skupaj s sporočilom.
Pošiljatelj:

1. kombinira ključ in sporočilo,

2. izračuna notranji hash (notranji sloj),
3. nato izračuna zunanji hash (zunanji sloj) in ustvari končno HMAC vrednost in
4. pošlje sporočilo in HMAC prejemniku.

Prejemnik:

1. prejme sporočilo in HMAC,
2. s svojim ključem ponovno izračuna HMAC in
3. če se vrednosti ujemata, je sporočilo pristno in nespremenjeno.

Glavne lastnosti HMAC

Da HMAC deluje varno in zanesljivo, mora imeti naslednje značilnosti prikazane v tabeli spodaj.

Tabela 21

Lastnost	Opis
Preverljivost	Prejemnik lahko preveri, ali je sporočilo prišlo od nekoga, ki pozna skrivni ključ, in ali ni bilo spremenjeno.
Enosmernost	Brez skrivnega ključa ni mogoče ustvariti pravilnega HMAC.
Odporen na napade podaljševanja (ang. *length extension*)	Dvojno hashiranje preprečuje zlorabe, ki so možne pri goli uporabi hash funkcij.
Hitrost	Temelji na standardnih hash funkcijah, zato je izračun hiter in primeren za veliko količino podatkov.
Prilagodljivost	Lahko se uporablja z različnimi hash algoritmi (SHA-2, SHA-3, BLAKE2 …).

Praktične uporabe HMAC

1. **Omrežni protokoli (TLS, IPsec, SSH)**
   - HMAC zagotavlja, da paketi niso bili spremenjeni na poti in da izvirajo od preverjene strani.
   - Preprečuje napade "man-in-the-middle".
2. **API-ji in spletne storitve**
   - Pri komunikaciji med odjemalci in strežniki se HMAC uporablja za preverjanje pristnosti zahtevkov.
   - Primer: podpisovanje HTTP glave z HMAC-SHA256.
3. **VPN povezave**
   - HMAC zagotavlja, da tunelski paketi niso bili spremenjeni in da prihajajo od veljavnega vira.

4. **Shranjevanje in prenos podatkov**
   - HMAC preverja, ali so datoteke (npr. varnostne kopije, posodobitve) pristne in nespremenjene.

Možni napadi na HMAC

Čeprav je HMAC, če je pravilno implementiran, varen, obstajajo možni napadi ali zlorabe v primeru slabih praks, ki so navedene spodaj.

Tabela 22

Vrsta napada	Opis
Kraja skrivnega ključa	Če napadalec pridobi ključ, lahko generira veljavne HMAC vrednosti.
Slab ključ	Kratek ali predvidljiv ključ poveča možnost uspešnega ugibanja ali napada z izčrpavanjem.
Ponovni napadi (ang. *replay attacks*)	Napadalec lahko prestreže sporočilo z veljavnim HMAC in ga kasneje pošlje znova. Rešitev: uporaba časovnih žigov ali nonce vrednosti.
Zastareli hash algoritmi	Uporaba HMAC z MD5 ali SHA-1 oslabi celoten mehanizem.

Tabela 23: Pregled priporočenih hash funkcij za HMAC

Funkcija	Status	Opombe
HMAC-MD5	Nevaren, zastarel	Napadi na MD5 so znani; uporaba ni več priporočena.
HMAC-SHA1	Nepriporočljivo	Obstajajo znani trki; postopno opuščanje.
HMAC-SHA256	Priporočljivo	Trenutno najpogosteje uporabljena varna možnost.
HMAC-SHA3	Priporočljivo	Moderna alternativa, odporna na napade, dobra izbira za nove sisteme.
HMAC-BLAKE2/BLAKE3	Priporočljivo	Zelo hitro in varno, primerno za moderne aplikacije in naprave.

Povzetek

- HMAC zagotavlja preverjanje celovitosti in istovetnosti sporočil s pomočjo kombinacije hash funkcije in skrivnega ključa.
- Je temeljni mehanizem v številnih varnostnih protokolih, kjer je treba preveriti, ali podatki niso bili spremenjeni in ali prihajajo iz zaupanja vrednega vira.
- Za varno uporabo je treba zagotoviti:
  - zadostno dolžino in naključnost skrivnega ključa,
  - sodobno hash funkcijo (SHA-256 ali novejšo) in
  - zaščito pred ponovnimi napadi (npr. s časovnimi žigi).
- HMAC ne zagotavlja zaupnosti, zato se pogosto uporablja skupaj s šifriranjem.

– Zaradi hitrosti in preprostosti je primeren za široko uporabo v omrežjih, aplikacijah in sistemih za zaščito podatkov.

4.6.2.1 Vaja: Prepoznavanje prave datoteke z uporabo HMAC

– Delo v paru: **oseba A** in **oseba B**.
– Vsak par si izbere **skupno skrivno geslo (ključ)**.
– **A pripravi dve datoteki**:
  – eno **originalno**,
  – eno **rahlo spremenjeno** (npr. popravi eno črko ali doda presledek).
– A izračuna **HMAC vrednost samo za originalno datoteko**.
– A pošlje B-ju:
  – obe datoteki,
  – pripadajočo HMAC vrednost originala.
– B z istim geslom preveri obe datoteki in mora **ugotoviti, katera je original** (tista, katere HMAC se ujema).
– Nato zamenjata vlogi — B pripravi primer, A preverja.
– Rezultat: vsak zna preveriti, katera datoteka je pristna, ne da bi mu to povedal sošolec.

Opcijsko: tretja oseba lahko poskuša "ponarediti" datoteko → vaja postane še bolj dinamična.

Pripravite poročilo, ki vsebuje:

– katero geslo sta uporabila,
– kratek opis obeh primerov datotek,
– izračunane HMAC vrednosti,
– kateri primer je bil pravi in kako sta to preverila,
– sklep: kaj sta se naučila o HMAC in preverjanju integritete.

Cilj vaje:

Praktično prikazati uporabo HMAC za preverjanje pristnosti in integritete datotek, kjer študenti v paru preverjajo, katera od dveh datotek je prava, na podlagi skupnega skrivnega ključa in izračunanih HMAC vrednosti.

Študent se nauči:

- uporabljati HMAC za zanesljivo preverjanje, ali je datoteka originalna ali spremenjena,
- razumeti pomen skupnega skrivnega ključa pri preverjanju pristnosti,
- prepoznati, da HMAC zazna tudi najmanjše spremembe v datoteki,
- izvesti preverjanje integritete neodvisno, brez razkrivanja vsebine,
- pripraviti kratko poročilo, ki povzema postopek, rezultate in spoznanja o uporabi HMAC v praksi.

4.6.2.2 Povzetek

HMAC združuje zgoščevalno funkcijo in skrivni ključ ter omogoča preverjanje celovitosti in istovetnosti sporočil.

- Ne zagotavlja zaupnosti podatkov, temveč preverja, ali je sporočilo nespremenjeno in prihaja od zaupanja vrednega vira.
- Uporablja se v številnih varnostnih protokolih (TLS, IPsec, SSH), pri API-jih, VPN povezavah ter preverjanju pristnosti datotek in posodobitev.
- Za varno uporabo je pomembno:
  - uporabiti dovolj dolg in naključen skrivni ključ,
  - izbrati sodobno hash funkcijo (SHA-256 ali novejšo),
  - zaščititi ključ pred razkritjem,
  - uporabiti časovne žige za zaščito pred ponovnimi napadi.
- Priporočene hash funkcije za HMAC so SHA-256, SHA-3 in BLAKE2/BLAKE3. MD5 in SHA-1 se, zaradi znanih ranljivosti, ne uporabljata več.
- HMAC je hiter, prenosljiv in odporen na napade podaljševanja, zato je primeren za široko uporabo v komunikacijskih omrežjih, aplikacijah in sistemih za zaščito podatkov.

4.6.3 Digitalni certifikati in infrastruktura javnih ključev (PKI)

Digitalni certifikat je elektronski dokument, ki dokazuje **povezavo med javnim ključem in identiteto** osebe, naprave ali organizacije (Chamishka, 2019).

Vloga certifikata je podobna osebni izkaznici ali potnemu listu v fizičnem svetu — potrjuje, **komu določen javni ključ pripada**.

Infrastruktura javnih ključev (PKI – ang. *Public Key Infrastructure*) je organiziran sistem tehnologij, pravil in postopkov, ki omogoča:

– ustvarjanje, izdajanje in upravljanje digitalnih certifikatov,
– preverjanje istovetnosti imetnikov certifikatov,
– varno izmenjavo informacij prek interneta.

PKI je temelj sodobne digitalne varnosti — uporablja se pri HTTPS, e-identiteti, elektronskem podpisovanju, VPN povezavah in številnih drugih varnostnih storitvah.

Osnovni namen digitalnih certifikatov in PKI

1. **Povezava javnega ključa z identiteto** – certifikat dokazuje, da javni ključ pripada določeni osebi, napravi ali organizaciji.
2. **Preverjanje istovetnosti** – omogoča preverjanje, ali je strežnik, oseba ali storitev res tista, za katero se izdaja.
3. **Vzpostavljanje zaupanja** – PKI omogoča zanesljivo delovanje varnostnih mehanizmov na globalni ravni (npr. HTTPS).
4. **Upravljanje življenjskega cikla ključev** – PKI skrbi za izdajo, obnovo, preklic in nadzor certifikatov skozi celotno življenjsko dobo.

Primer delovanja digitalnega certifikata

Predpostavimo, da uporabnik obišče spletno stran banke (https://www.banka.si). Kaj se ob tem dogaja?

1. Spletni strežnik pošlje uporabniku **digitalni certifikat**.
2. Brskalnik preveri, ali je certifikat:
   – izdan s strani zaupanja vrednega overitelja (CA),
   – veljaven (ni potekel, ni preklican),
   – izdan za točno to domeno.
3. Če je vse v redu, brskalnik vzpostavi varno povezavo (TLS).
4. Uporabnik lahko zaupa, da komunicira z resnično banko, ne z napadalcem.

Tabela 24: Glavne lastnosti digitalnih certifikatov

Lastnost	Opis
Povezava ključa in identitete	Certifikat potrjuje, da javni ključ pripada določeni osebi ali organizaciji.
Overjenost	Certifikat je podpisan s strani zaupanja vrednega overitelja.
Preverljivost	Vsakdo lahko preveri veljavnost certifikata z javnim ključem overitelja.
Časovna veljavnost	Certifikati imajo določen rok veljavnosti in jih je treba redno obnavljati.
Možnost preklica	Če je zasebni ključ kompromitiran, se certifikat lahko prekliče (CRL, OCSP).
Standardiziran format	Najpogosteje uporabljeni formati so X.509 (TLS/HTTPS) in PGP za elektronsko pošto.

Sestavni deli digitalnega certifikata (X.509)

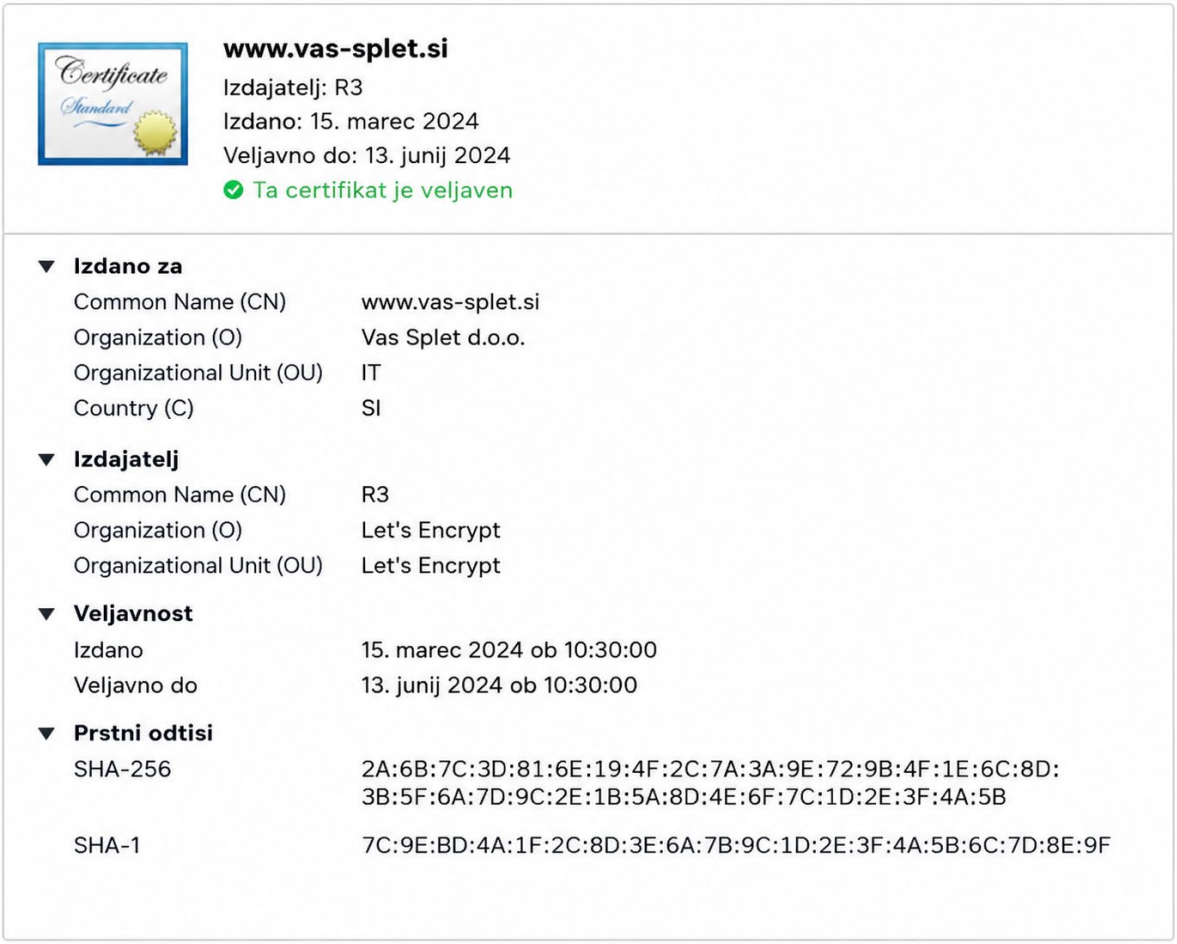
Tabela 25: Tipičen digitalni certifikat vsebuje (Cooper idr., 2008)

Element	Opis
Serijska številka	Enolična identifikacija certifikata.
Podatki o lastniku	Ime, organizacija, domena, e-pošta ali druge identifikacijske informacije.
Javni ključ	Ključ, ki ga certifikat povezuje z identiteto lastnika.
Podatki o izdajatelju (CA)	Informacije o overitelju, ki je izdal certifikat.
Veljavnost	Datum začetka in poteka veljavnosti certifikata.
Podpis izdajatelja	Digitalni podpis CA, ki potrjuje veljavnost certifikata.
Razširitve	Dodatni podatki (npr. dovoljena uporaba certifikata, informacije o preklicu).

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 04:9f:56:a1:3d:22:11:4e:b3:8c
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: CN=Let's Encrypt Authority X3, O=Let's Encrypt, C=US
        Validity
            Not Before: May 10 08:00:00 2025 GMT
            Not After : Aug  8 08:00:00 2025 GMT
        Subject: CN=www.example.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (2048 bit)
                Modulus: [redacted for brevity]
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Subject Alternative Name:
                DNS:www.example.com
            X509v3 Key Usage:
                Digital Signature, Key Encipherment
            X509v3 Extended Key Usage:
                TLS Web Server Authentication
    Signature Algorithm: sha256WithRSAEncryption
         Signature: [redacted for brevity]
```

Slika 5: Primer sestavnih delov digitalnega certifikata (X.509).

Vir: OpenSSL, b. d.



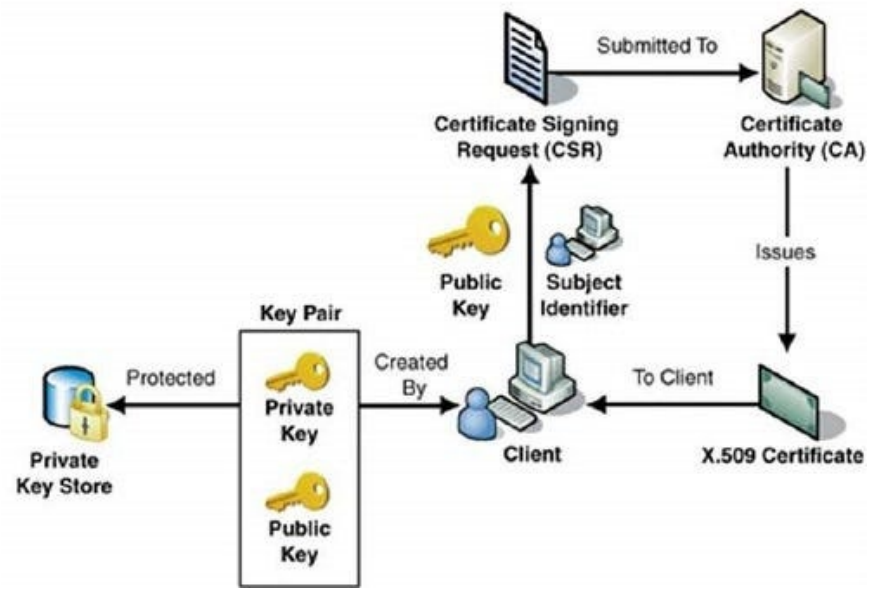
Slika 6: Primer SSL certifikata.
Vir: Namecheap, b. d.

Vloga infrastrukture javnih ključev (PKI)

PKI omogoča celovit sistem zaupanja in sestavlja naslednje elemente:

1. **Overitelji (ang. *CA – Certificate Authorities*)**
   - Izdajajo digitalne certifikate.
   - Podpisujejo certifikate s svojim zasebnim ključem.
   - Delujejo kot zaupanja vredne tretje osebe.
   - Primeri: DigiCert, Let's Encrypt, SI-TRUST (Slovenija) (SI-TRUST – Državni center za storitve zaupanja, 2025).
2. **Registracijski organi (ang. *RA – Registration Authorities*)**
   - Preverjajo identiteto prosilcev pred izdajo certifikata.
   - Delujejo kot vmesni člen med uporabniki in CA.
3. **Shrambe in repozitoriji**
   - Hramba in distribucija javnih ključev in certifikatov (npr. LDAP strežniki).

4. **Seznami preklicanih certifikatov (CRL) in OCSP**
   - Omogočajo preverjanje, ali je certifikat preklican ali še veljaven.
5. **Politike in postopki (CP/CPS)**
   - Dokumenti, ki določajo pravila za izdajo, uporabo in preklic certifikatov.



Slika 7: Shematski prikaz delovanja infrastrukture javnih ključev (PKI).
Vir: Chamishka, 2019.

Praktične uporabe PKI

1. **HTTPS in TLS**
   - Strežniški certifikati potrjujejo identiteto spletnih strani in omogočajo varno komunikacijo.
   - Brskalniki zaupajo le certifikatom, ki jih izdajo priznani CA.
2. **Elektronska identiteta in podpisovanje**
   - Državni sistemi e-identitete (npr. SIGEN-CA v Sloveniji) uporabljajo certifikate za preverjanje uporabnikov in podpisovanje dokumentov.
3. **VPN povezave in notranja omrežja**
   - Certifikati omogočajo varno avtentikacijo naprav in uporabnikov znotraj organizacij.
4. **Podpisovanje programske opreme**
   - Podjetja podpišejo aplikacije in posodobitve, da uporabniki preverijo izvor.
5. **E-pošta (S/MIME, PGP)**
   - Certifikati se uporabljajo za digitalno podpisovanje in šifriranje elektronske pošte.

Tabela 25: Možne ranljivosti in napadi

Težava / napad	Opis
Kraja zasebnega ključa lastnika	Če je zasebni ključ ogrožen, je treba certifikat takoj preklicati.
Napadi na CA	Če je napadena ustanova CA, lahko napadalci izdajo lažne certifikate.
Zloraba zaupanja	Če brskalnik zaupa kompromitiranemu CA, je ogrožena celotna veriga zaupanja.
Phishing z lažnimi certifikati	Napadalci lahko uporabijo certifikate za lažne strani, če uporabniki ne preverijo podrobnosti.
Zastareli algoritmi	MD5, SHA-1 in kratki RSA ključi so ranljivi za napade. Danes so obvezni sodobni algoritmi (SHA-2, SHA-3, ECC).

4.6.3.1 Povzetek

- **Digitalni certifikati** povezujejo javne ključe z identitetami in omogočajo preverjanje istovetnosti v digitalnem okolju.
- **PKI** predstavlja celoten sistem, ki omogoča izdajo, preverjanje, upravljanje in preklic teh certifikatov.
- Uporablja se v spletnih varnostnih protokolih (HTTPS), e-identiteti, elektronskih podpisih, VPN povezavah in drugih storitvah.
- Zaupanje v PKI temelji na **verigi zaupanja**, ki vključuje korenske in vmesne overitelje.
- Za varno uporabo je nujno:
  - zaščititi zasebne ključe,
  - uporabljati sodobne algoritme (SHA-256, ECC),
  - preverjati veljavnost certifikatov,
  - zaupati le preverjenim izdajateljem.
- Napadi na CA, slabo upravljanje ključev ali uporaba zastarelih algoritmov lahko ogrozijo celoten sistem zaupanja.

4.6.4 Digitalni podpis

Digitalni podpis je kriptografski mehanizem, ki omogoča preverjanje **istovetnosti pošiljatelja**, **celovitosti podatkov** in zagotavlja **nezanikljivost** (ang. *non-repudiation*) – pošiljatelj ne more zanikati, da je podpisal dokument ali sporočilo.

Deluje podobno kot ročni podpis, vendar je matematično veliko bolj zanesljiv in ni ponaredljiv.

Digitalni podpis temelji na **asimetrični kriptografiji**, pri kateri obstajata **zasebni ključ** (uporablja se za podpisovanje) in **javni ključ** (uporablja se za preverjanje podpisa).

Da bi bil postopek hiter in varen, se ne podpisuje celotno sporočilo, temveč najprej izračunamo **zgoščeno vrednost (hash)** sporočila.

Osnovni namen digitalnega podpisa

1. **Preverjanje istovetnosti pošiljatelja** – digitalni podpis potrjuje, da sporočilo izvira od imetnika zasebnega ključa.
2. **Zagotavljanje celovitosti** – podpis vključuje hash vsebine, zato vsaka sprememba sporočila povzroči neujemanje podpisa.
3. **Nezanikljivost** – pošiljatelj ne more kasneje zanikati, da je podpis ustvaril, ker je ta vezan na njegov zasebni ključ.
4. **Učinkovitost postopka** – zgoščevalne funkcije omogočajo, da se podpiše le kratek "digitalni odtis" sporočila, ne celotna vsebina.

Primer delovanja

Predpostavimo, da želi **Ana** poslati **Borutu** digitalno podpisano sporočilo.

Ana ima svoj par ključev (zasebni in javni), Borut pa pozna njen javni ključ.

Koraki:

1. Ana izračuna hash svojega sporočila.
2. Ta hash podpiše s **svojim zasebnim ključem** (npr. RSA ali ECDSA).
3. Podpis skupaj s sporočilom pošlje Borutu.
4. Borut z uporabo **Aninega javnega ključa** preveri, ali podpis ustreza sporočilu.
   - Če se podpis ujema, to pomeni, da je sporočilo pristno in nespremenjeno.
   - Če se ne ujema, to pomeni, da je bilo sporočilo spremenjeno ali podpis ni veljaven.

Tabela 26: Glavne lastnosti digitalnih podpisov

Lastnost	Opis
Avtentikacija	Prejemnik lahko preveri, kdo je podpisal sporočilo (na podlagi javnega ključa).
Celovitost	Vsaka sprememba vsebine povzroči neujemanje hash vrednosti in s tem neveljaven podpis.
Nezanikljivost	Pošiljatelj ne more zanikati podpisa, saj je vezan na njegov zasebni ključ.
Učinkovitost	Hashiranje omogoča hitro podpisovanje velikih dokumentov.
Zanesljivost	Temelji na preverjenih matematičnih problemih (faktorizacija, eliptične krivulje ipd.).

Praktične uporabe digitalnih podpisov

1. **Podpisovanje dokumentov**
   - Digitalni podpisi se uporabljajo v elektronskem poslovanju, pravnih postopkih, javni upravi in podjetjih.
   - Primer: elektronski podpis PDF dokumentov (npr. Adobe Acrobat, SI-PASS).
2. **Elektronska pošta**
   - Protokoli, kot sta S/MIME ali PGP, omogočajo podpisovanje e-pošte za preverjanje izvora in celovitosti vsebine.
3. **Posodabljanje programske opreme**
   - Podjetja digitalno podpišejo programske pakete in posodobitve, da uporabniki preverijo, ali so izvirne in nespremenjene.
4. **Digitalne transakcije**
   - V finančnih sistemih in blockchain tehnologijah se digitalni podpisi uporabljajo za avtorizacijo transakcij in preprečevanje ponarejanja.
5. **Avtentikacija strežnikov in storitev**
   - Digitalni podpisi so sestavni del TLS certifikatov, ki potrjujejo identiteto spletnih strani in omogočajo varne povezave (HTTPS).

Tabela 27: Možni napadi in varnostni izzivi

Vrsta napada / težava	Opis
Kraja zasebnega ključa	Če napadalec pridobi zasebni ključ, lahko ponareja podpise.
Slab upravljalski nadzor nad ključi	Nevarno hranjenje zasebnih ključev (npr. nezaščiteni diski) omogoča zlorabe.
Uporaba zastarelih algoritmov	MD5, SHA-1 ali prekratki RSA ključi zmanjšujejo varnost digitalnih podpisov.
Socialni in infrastrukturni napadi	Napadalci lahko napadejo overitelje certifikatov ali uporabnike, ne nujno algoritme.
Napadi z izsiljevanjem (ang. *replay*)	Če sistem ne preverja časovnih žigov, lahko podpisana sporočila ponovno uporabijo.

Tabela 29: Pregled pogosto uporabljenih algoritmov za digitalne podpise

Algoritem	Tip	Status	Opombe
RSA	Asimetrični	Priporočen	Široko uporabljen, podpira podpisovanje in preverjanje.
DSA (ang. *Digital Signature Algorithm*)	Asimetrični	Priporočen, vendar počasnejši	Standardiziran v ZDA, primeren za nekatere aplikacije.
ECDSA (ang. *Elliptic Curve Digital Signature Algorithm*)	Asimetrični	Priporočen	Zagotavlja visoko varnost z manjšimi ključi; dobra izbira za mobilne naprave.
EdDSA (ang. *Edwards-curve DSA*)	Asimetrični	Priporočen (novejši)	Moderna alternativa, hitra in varna; primerna za sodobne aplikacije.

Aktivnosti ob podpisu datoteke (v datoteko se shrani):

- zgoščeno vrednost vsebine,
- digitalni podpis (hash, šifriran z zasebnim ključem),
- tvoj javni ključ in podatke o potrdilu (X.509),
- časovni žig, če je bil uporabljen.

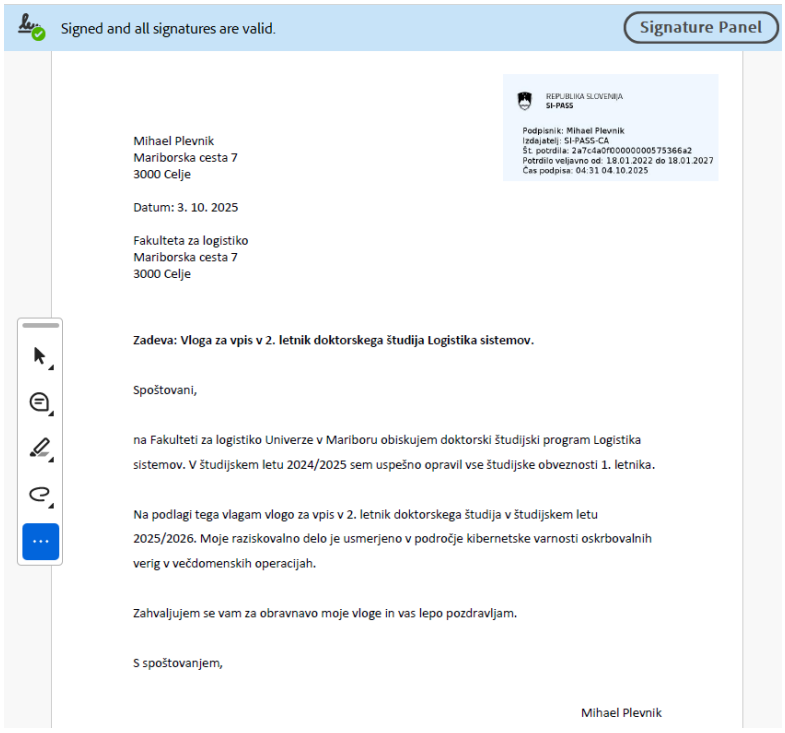
Preverjanje digitalnega podpisa

Ko prejemnik odpre PDF, obstajata dve spodaj opisani možnosti:

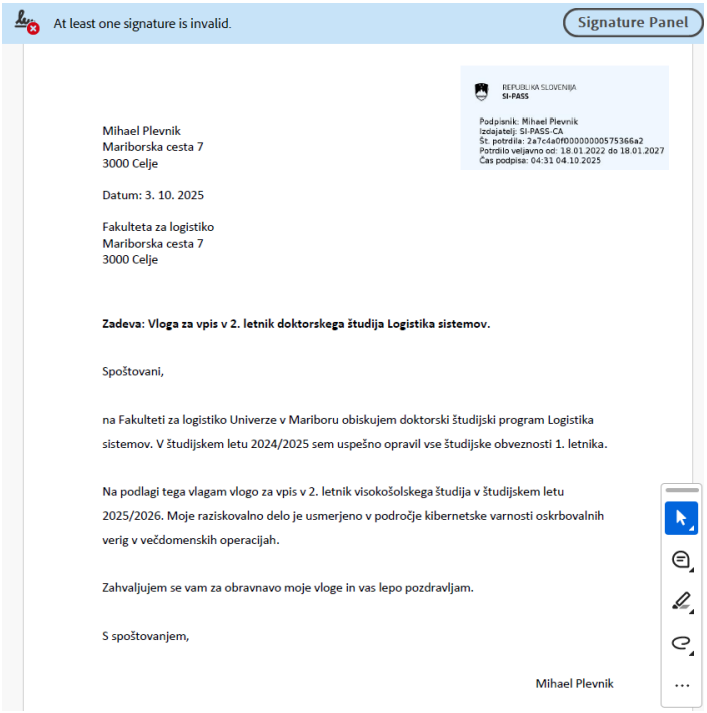
(A) Preverjanje samodejno (če ima ustrezno programsko opremo)

Če prejemnik uporablja npr. **Adobe Acrobat Reader** ali drug program, ki podpira evropska kvalificirana potrdila (eIDAS), bo podpis samodejno preverjen:

- program preveri **integriteto vsebine** (če je bila datoteka po podpisu spremenjena),
- program preveri **verigo zaupanja** (SIGEN-CA → nacionalna korenska overitelja → zaupanja vredni seznam EU) in
- prejemnik vidi status podpisa (zelen kljukica ali rdeč križ).



Slika 8: Primer preverjanja digitalnega podpisa.
Vir: lasten, zajem zaslona.



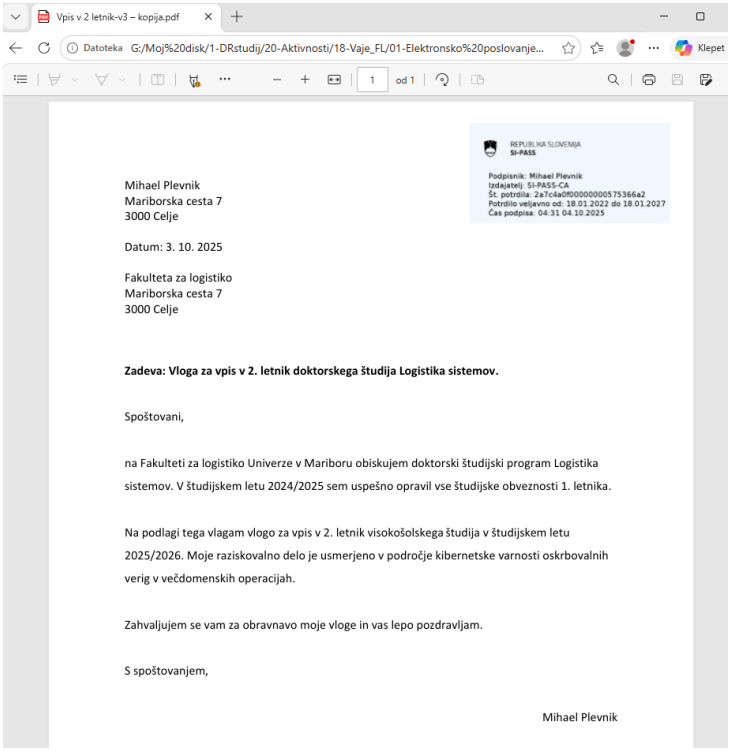
Slika 9: Primer neveljavnega digitalnega podpisa.
Vir: lasten, zajem zaslona.

(B) Preverjanje NI izvedeno samodejno

Če uporablja drug ali starejši pregledovalnik PDF ali če ima onemogočeno preverjanje zaupanja, bo:

- dokument **videti kot podpisan**,
- a program **ne preveri verodostojnosti** podpisa, ne opozori na potek veljavnosti, ne preveri revokacije itd.

Prejemnik torej "ima podpisan dokument", a brez preverjanja **ne ve**, ali je podpis veljaven, kdo je dejansko podpisal, ali je bil dokument po podpisu spremenjen in ali je certifikat morda preklican.



Slika 10: Primer ko digitalni podpis ni preverjen.
Vir: lasten.

4.6.4.1 Demonstracija

- E-podpis dokumenta s SI-PASS;
- Primerjava različnih podpisanih dokumentov:
  - dokument v MS Wordu,
  - dokument v pdf,
  - lastnoročno podpisan dokument v pdf,
  - e-podpisan dokument v pdf,
  - spremenjen e-podpisan dokument.

4.6.4.2 Vaja SI-PASS Podpis

Cilj vaje

Študente naučiti samostojne uporabe storitve SI-PASS za preverjanje identitete, pridobitev podatkov o lastnem digitalnem potrdilu ter izvedbo elektronskega podpisa dokumenta.

Študent se nauči:

- poiskati in preveriti veljavnost svojega digitalnega potrdila (npr. SIGEN-CA) v sistemu ali brskalniku,
- razumeti osnovne podatke digitalnega potrdila (nosilec, serijska številka, obdobje veljavnosti, namen),
- uporabljati spletni portal SI-PASS za elektronsko podpisovanje dokumentov,
- pripraviti poročilo in ga lastnoročno digitalno podpisati z izbranim načinom prijave (npr. kvalificirano potrdilo ali smsPASS),
- razumeti pomen kvalificiranega e-podpisa za varnost in pravno veljavnost dokumentov.

Naloga:

- Pridobite svoje digitalno potrdilo SIGENCA.
- Pridobite informacije o vašem digitalnem potrdilu:
  - serijska številka,
  - veljavnost digitalnega potrdila in
  - nosilec.
- Izdelajte poročilo o izdelavi naloge in ga digitalno podpišite.

Uvod

Storitev SI-PASS je enotna točka za preverjanje identitete različnih uporabnikov (državljanov, poslovnih subjektov, javnih uslužbencev) ter elektronsko podpisovanje vlog in ostalih dokumentov. SI-PASS se praviloma uporablja v okviru opravljanja posameznih elektronskih storitev (npr. eUprava, SPOT).

SI-PASS omogoča uporabo različnih sredstev za prijavo:

- digitalna potrdila (kvalificirana in ostala),
- mobilna identiteta smsPASS,
- uporabniško ime SI-PASS in geslo,
- uporabniški račun Google, Facebook in Microsoft,
- čezmejna prijava v EU.

Nabor dovoljenih sredstev za prijavo določi ponudnik posamezne elektronske storitve. Pri elektronskih storitvah, ki zahtevajo zanesljivo identifikacijo uporabnika, se običajno zahteva prijava s kvalificiranim digitalnim potrdilom ali mobilno identiteto smsPASS.

Državljani drugih članic Evropske Unije lahko uporabijo tista sredstva za prijavo, ki jih za potrebe čezmejne prijave dovoljuje njihova država.

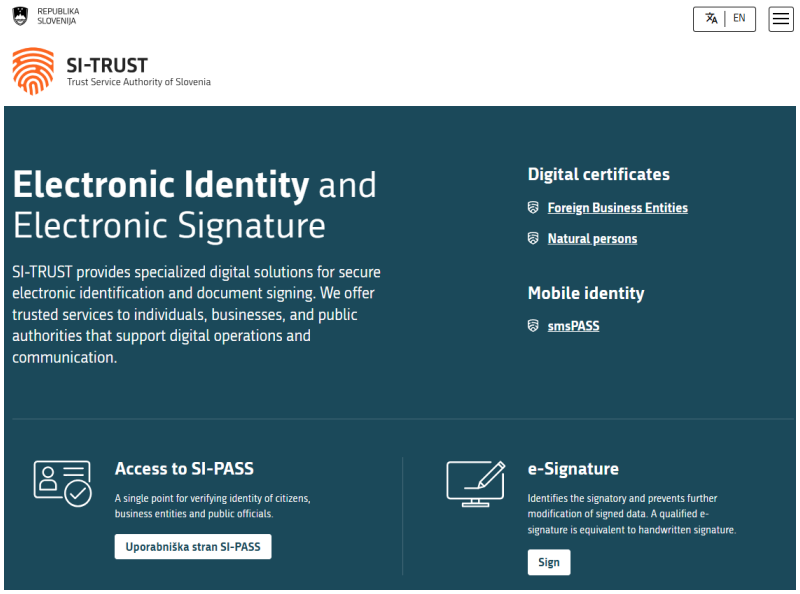
SI-PASS podpira enotno prijavo in odjavo uporabnika, zato je zanj hkratna uporaba več elektronskih storitev (npr. eUprave, SPOT, eDavkov) enostavnejša. Zahteva le enkratno prijavo in odjavo uporabnika iz SI-PASS in posledično tudi iz vseh storitev, ki jih je uporabljal.

V postopku prijave lahko SI-PASS drugi elektronski storitvi posreduje tudi določene podatke o uporabniku (npr. ime, priimek, EMŠO, naslov itd.), vendar vedno le z njegovim soglasjem.

E-podpis, predstavlja podpisnika in prepreči nadaljnje spreminjanje podpisanih podatkov. Lastnoročnemu podpisu enakovreden je kvalificirani elektronski podpis.

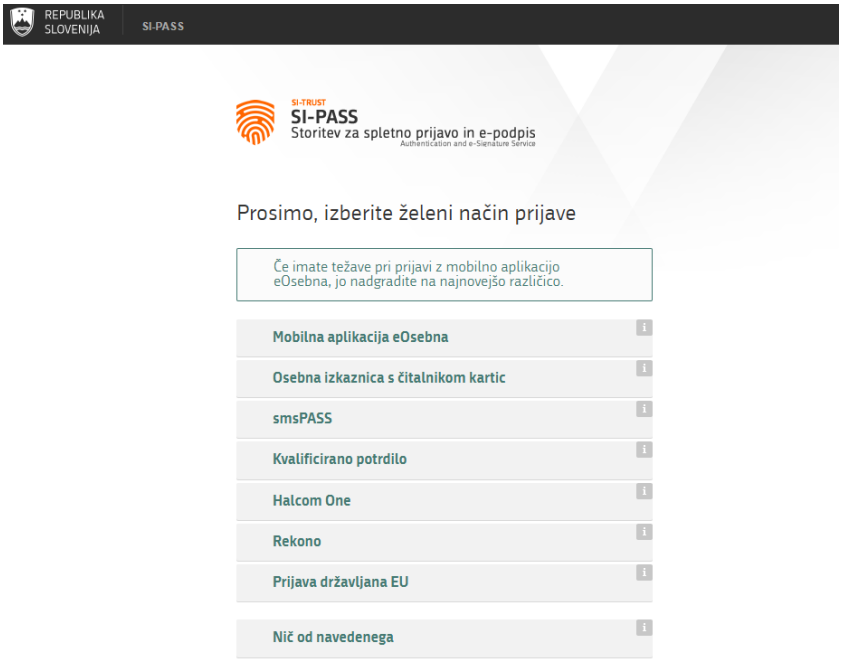
Navodilo za podpisovanje dokumentov

– Podpisovanje se izvaja na spletni strani: https://www.si-trust.gov.si/



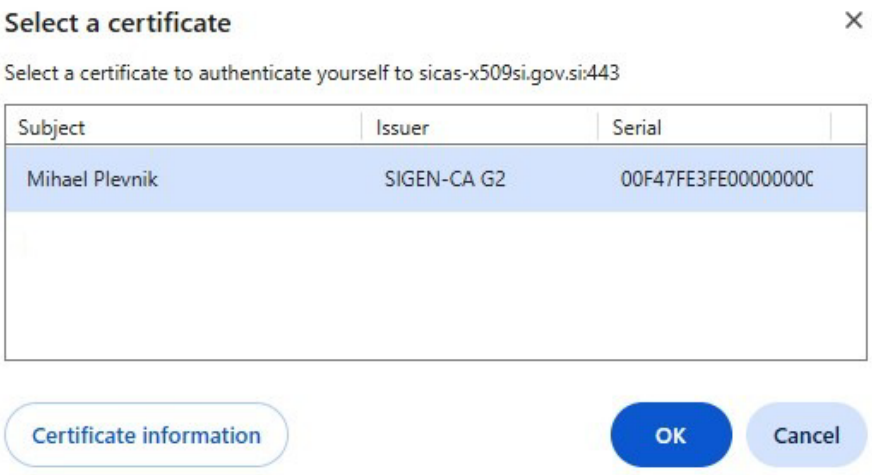
Slika 11: Portalna stran SI-TRUST.
Vir: lasten, zajem zaslona, SI-TRUST – Državni center za storitve zaupanja, 2025.

– uporabite funkcijo E-podpis in kliknite na gumb »Podpišite«
– izberite eno od možnosti prijave



Slika 12: Prijava v storitev.
Vir: lasten, zajem zaslona, SI-TRUST – Državni center za storitve zaupanja, 2025.

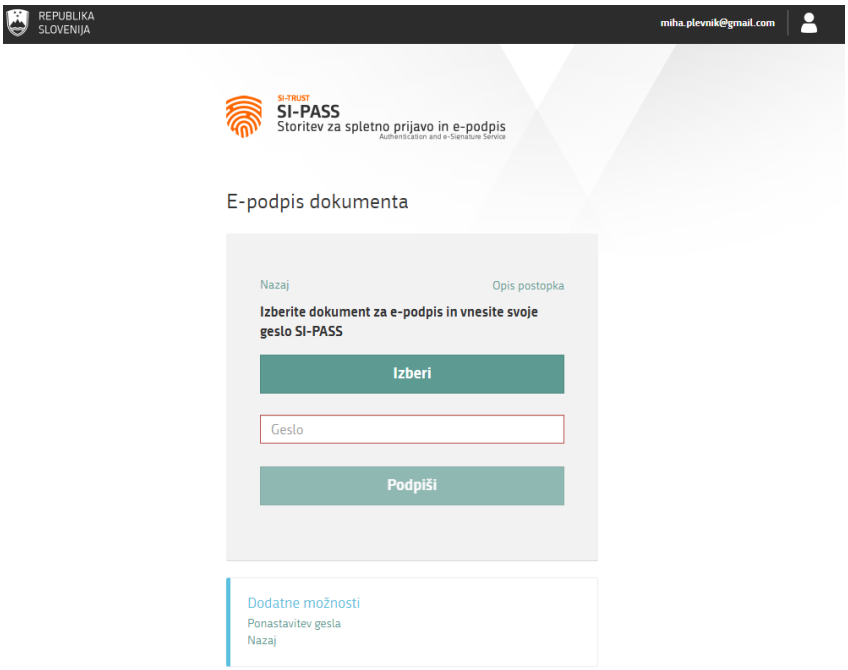
– Primer podpisa z izbiro »Kvalificiranega potrdila«



Slika 13: Prijava z digitalnim potrdilom.
Vir: lasten, zajem zaslona, SI-TRUST – Državni center za storitve zaupanja, 2025.

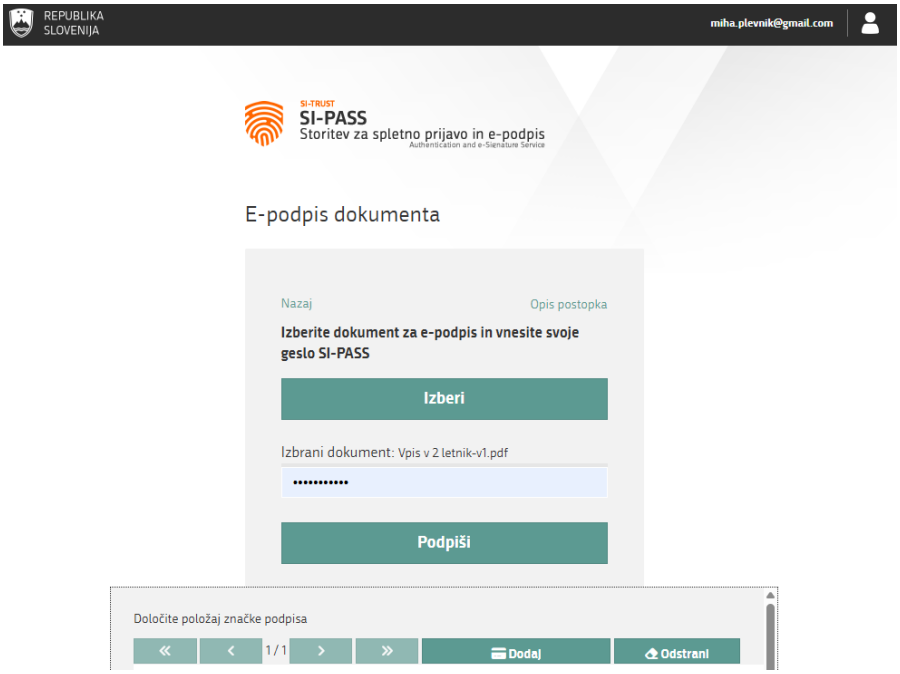
* Dobite informacijo, da imate veljavno digitalno potrdilo SI-PASS.

– Izberite možnost E-podpis dokumenta.



Slika 14: E-podpis dokumenta.
Vir: lasten, zajem zaslona. SI-TRUST – Državni center za storitve zaupanja, 2025.

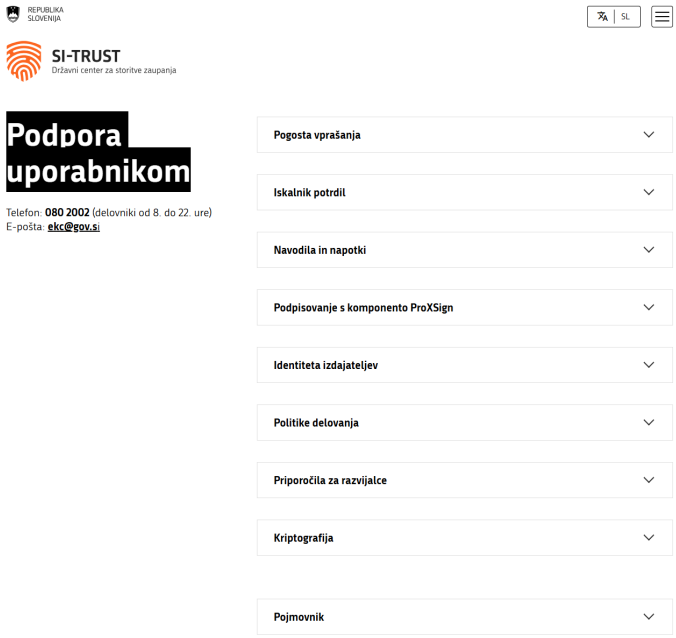
– Izberite pdf dokument, vnesite geslo, določite mesto značke podpisa in izberite možnost »Podpiši«.



Slika 15: Izvedba podpisa dokumenta.
Vir: lasten, zajem zaslona. SI-TRUST – Državni center za storitve zaupanja, 2025.

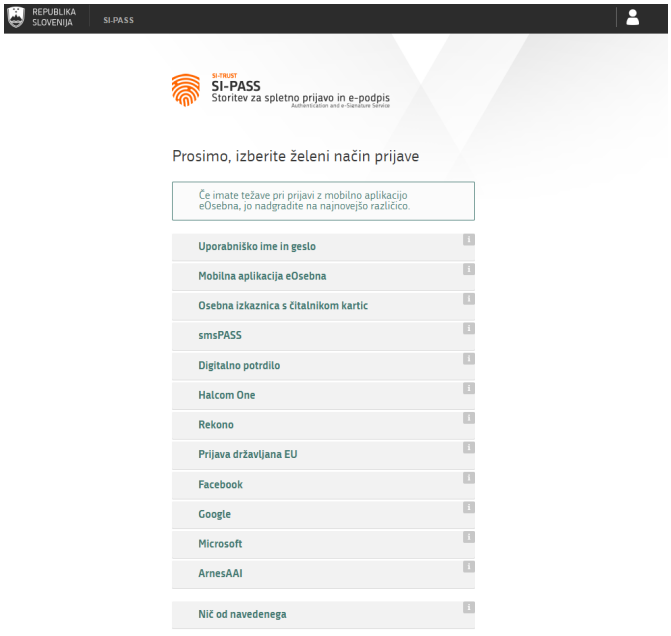
* Izvede se podpis pdf dokumenta, ki se avtomatsko shrani na vaš računalnik.

– Podpora uporabnikom se izvaja na spletnem naslovu: https://www.si-trust.gov.si/sl/podpora-uporabnikom.



Slika 16: Podpora uporabnikom SI-TRUST.
Vir: lasten, zajem zaslona. SI-TRUST – Državni center za storitve zaupanja, 2025.

Uporabniška stran SI-PASS



Slika 17: Nabor možnih prijav za urejanje uporabniških nastavitev.
Vir: lasten, zajem zaslona. SI-TRUST – Državni center za storitve zaupanja, 2025.

Urejanje uporabniških nastavitev:

– Urejanje nastavitev računa,
– Urejanje registriranih načinov prijave,
– Urejanje digitalnih potrdil izdajatelja SI-PASS,
– Urejanje obveščanja o prijavi,
– Urejanje soglasij za posredovanje podatkov,
– Pregled zadnjih prijav.

Digitalno potrdilo

Lokacija in način dostopa do informacij o **digitalnem potrdilu SIGEN-CA (sigenca)** na računalniku sta odvisna od:

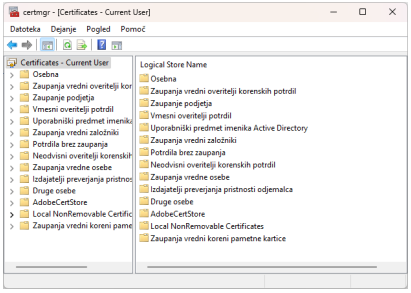
– operacijskega sistema (npr. Windows, macOS) in
– od tega, ali si potrdilo namestil v brskalnik ali v sistemski shrambi (Windows Certificate Store).

1. Preverjanje prek sistema Windows

Če si potrdilo namestil sistemsko (kar je privzeto za .p12/.pfx datoteke), ga najdeš tukaj:

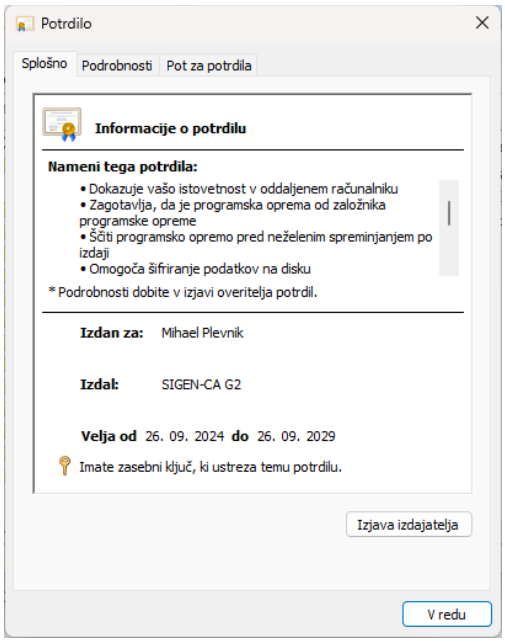
Koraki:

1. Pritisni tipko **Win + R.**
2. Vpiši:
   certmgr.msc
   in pritisni **Enter**.
3. Odpre se ti okno **Upravljalnik potrdil (Certificate Manager)**.



Slika 18: Urejanje digitalnih potrdil.
Vir: lasten zajem zaslona.

1. V levem meniju izberi: **Osebno (Personal)** → **Potrdila (Certificates)**
2. Na desni strani boš videl seznam svojih digitalnih potrdil.
3. Poišči tisto, kjer je v stolpcu *Izdano za (Issued To)* tvoje ime, v stolpcu *Izdano od (Issued By)* pa **SIGEN-CA**.
4. Dvoklik na potrdilo ti pokaže podrobnosti:
   - veljavnost,
   - namen uporabe,
   - javni ključ,
   - certifikatna pot itd.



Slika 19: Informacije o digitalnem potrdilu.
Vir: lasten, zajem zaslona.

2. Preverjanje v brskalniku (če je potrdilo nameščeno tam)

Google Chrome / Microsoft Edge (uporabljata Windows shrambo) uporablja isto pot kot zgoraj (Windows Certificate Store), zato ti ni treba iskati posebej v brskalniku.

Mozilla Firefox

Firefox ima **lastno** shrambo potrdil, ločeno od Windowsa.

1. Odpri Firefox.
2. Klikni ☰ → **Nastavitve (Settings)**.
3. Na levi izberi **Zasebnost in varnost (Privacy & Security)**.

4. Pomakni se dol do razdelka **Potrdila (Certificates)**.
5. Klikni **Poglej potrdila (View Certificates)**.
6. V zavihku **Vaša potrdila (Your Certificates)** boš videl osebno SIGEN-CA potrdilo.

3. Datoteka s potrdilom (.p12 ali .pfx)

Če iščeš **izvorno datoteko**, ki si jo prenesel iz portala SIGEN-CA, je običajno ime:

ime_priimek.p12
ali
ime_priimek.pfx
Najpogosteje je shranjena v:

– **Prenosi** (Downloads),
– **Namizje** (Desktop) ali
– v mapo, ki si jo izbral med postopkom namestitve.

Če ne veš, lahko uporabiš **Windows iskanje** in vpišeš:
*.p12
ali
*.pfx

DODATNO: Kaj lahko preveriš v potrdilu?

Ko odpreš potrdilo, so pomembni podatki:

- **izdano za (Issued to)**: tvoje ime in EMŠO (če gre za osebno potrdilo),
- **izdano od (Issued by)**: SIGEN-CA,
- **velja od – do**: časovna veljavnost,
- **namen**: digitalni podpis, šifriranje, prijava itd.

4.6.4.3 Povzetek

- Digitalni podpis omogoča preverjanje izvora sporočila, zagotavlja celovitost in preprečuje zanikanje avtorstva.
- Temelji na **asimetrični kriptografiji** in uporabi zgoščevalnih funkcij.
- Najprej se izračuna hash sporočila, nato se ta hash podpiše z zasebnim ključem.

- Prejemnik preveri podpis z javnim ključem pošiljatelja.
- Uporablja se v elektronskih dokumentih, e-pošti, programski opremi, varnih komunikacijah in transakcijah.
- Za varno uporabo so potrebni močni algoritmi (npr. RSA, ECDSA), dobra zaščita zasebnih ključev in sodobne hash funkcije (SHA-2, SHA-3).
- Slabi algoritmi, slabo upravljanje ključev ali kompromitacija infrastrukture lahko ogrozijo celoten sistem digitalnega podpisovanja.

4.6.5 PGP (ang. Pretty Good Privacy)

PGP (ang. *Pretty Good Privacy*) je kriptografski sistem, ki združuje **šifriranje, digitalno podpisovanje** in **upravljanje ključev** za zagotavljanje **zaupnosti, celovitosti** in **istovetnosti** pri izmenjavi podatkov — najpogosteje elektronske pošte in datotek (Raicea, 2017).

PGP je zasnoval **Philip Zimmermann** leta 1991 kot odprto rešitev za zaščito komunikacij brez odvisnosti od centralnih certifikacijskih oblasti. Danes obstajata dve glavni različici:

- **klasični PGP** (komercialni) in
- **OpenPGP** – odprt standard, ki ga uporabljajo številna orodja (npr. GnuPG/GPG).

PGP je postal eden najbolj razširjenih sistemov za varno elektronsko pošto, saj združuje **asimetrično kriptografijo, simetrično šifriranje** in **zgoščevalne funkcije** v en sam celovit postopek.

Osnovni namen PGP

1. **Zagotavljanje zaupnosti** – sporočilo je šifrirano tako, da ga lahko prebere le prejemnik z ustreznim zasebnim ključem.
2. **Zagotavljanje celovitosti** – zgoščevalne funkcije in digitalni podpisi omogočajo zaznavanje sprememb vsebine.
3. **Preverjanje istovetnosti pošiljatelja** – digitalni podpis dokazuje, kdo je sporočilo poslal.
4. **Decentralizirano upravljanje ključev** – namesto centralne infrastrukture zaupanja PGP uporablja t. i. **splet zaupanja (ang. *Web of Trust*)**.

Primer delovanja PGP

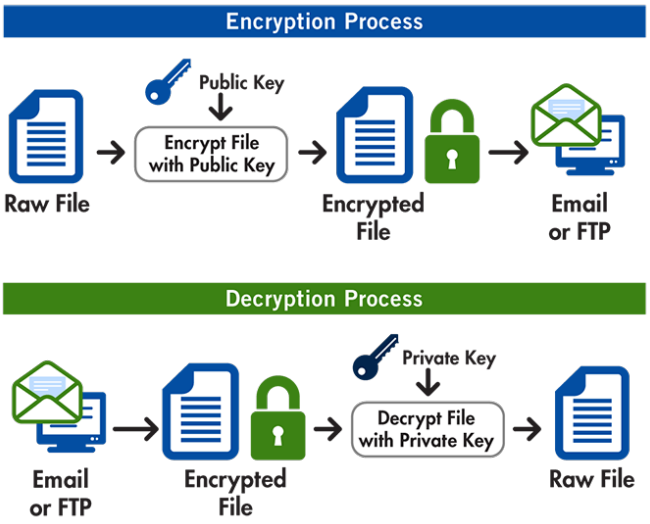
Predpostavimo, da želi **Ana** poslati **Borutu** varno e-pošto:

1. **Ana pripravi sporočilo** → npr. besedilo e-pošte.
2. **Ana ustvari hash sporočila** in ga podpiše s svojim **zasebnim ključem** → nastane digitalni podpis.
3. **Ana ustvari naključni simetrični ključ**, s katerim šifrira vsebino sporočila (hitro in učinkovito).
4. Ta simetrični ključ šifrira z **Borutovim javnim ključem**.
5. Ana Borutu pošlje:
   - šifrirano sporočilo,
   - šifriran simetrični ključ,
   - svoj digitalni podpis.

Borut:

1. S svojim zasebnim ključem najprej dešifrira simetrični ključ.
2. Nato s tem ključem dešifrira sporočilo.
3. Z Aninim javnim ključem preveri njen digitalni podpis.
4. Če je podpis veljaven, ve, da sporočilo res prihaja od Ane in da ni bilo spremenjeno.

Ta postopek združuje **šifriranje (zaupnost)**, **hashiranje in podpisovanje (celovitost + istovetnost)** ter **kombinacijo simetrične in asimetrične kriptografije**.



Slika 20: Shematski prikaz delovanja PGP.
Vir: Raicea, 2017.

Tabela 30: Glavne lastnosti PGP

Lastnost	Opis
Hibridno šifriranje	PGP združuje hitro simetrično šifriranje za vsebino in asimetrično šifriranje za varno izmenjavo ključev.
Digitalni podpisi	Zagotavljajo preverjanje izvora in celovitosti sporočil.
Decentraliziran sistem zaupanja	Temelji na spletni mreži uporabnikov, ne na centralnih certifikacijskih organih.
Močna varnost	Uporablja preverjene algoritme (npr. RSA, AES, SHA-2).
Prilagodljivost	Primeren za e-pošto, datoteke, arhive in druge oblike podatkov.

Praktične uporabe PGP

1. **Varna elektronska pošta**
   - PGP je najbolj znan po šifriranju in podpisovanju e-poštnih sporočil.
   - Uporabniki ustvarijo lastne pare ključev in jih izmenjujejo neposredno ali prek javnih ključnih strežnikov.
2. **Podpisovanje in preverjanje datotek**
   - PGP omogoča ustvarjanje digitalnih podpisov datotek, da se preveri, ali so avtentične in nespremenjene.
   - Pogosto se uporablja pri odprtokodnih programih (npr. podpisane distribucije Linuxa).
3. **Šifriranje datotek in arhivov**
   - PGP se uporablja tudi za varno shranjevanje občutljivih podatkov, npr. varnostnih kopij, poslovnih dokumentov.
4. **Avtentikacija identitet**
   - Splet zaupanja omogoča uporabnikom, da preverijo identiteto drugih preko medsebojnih podpisov ključev.

Splet zaupanja (ang. *Web of Trust*)

Za razliko od PKI, kjer zaupanje zagotavljajo centralni overitelji (CA), PGP uporablja **decentraliziran model** zaupanja, kjer:

- vsak uporabnik lahko **podpiše javne ključe drugih uporabnikov**, če jih osebno pozna ali zaupa njihovim ključem,
- sčasoma nastane **mreža medsebojno podpisanih ključev**, ki omogoča preverjanje identitet brez centralne avtoritete,
- stopnja zaupanja je odvisna od števila in kakovosti teh podpisov.

Ta pristop je bolj demokratičen in odporen proti kompromitaciji ene centralne točke, vendar zahteva aktivno sodelovanje uporabnikov.

Tabela 31: Možni napadi in težave pri uporabi PGP

Napad / težava	Opis
Kraja zasebnega ključa	Če napadalec pridobi zasebni ključ uporabnika, lahko bere njegova sporočila in ustvarja lažne podpise.
Slabo upravljanje ključev	Pozabljeni, izgubljeni ali nezaščiteni ključi povzročajo varnostna tveganja.
Manjkajoča preveritev ključev	Če uporabniki ne preverjajo javnih ključev, lahko pride do napadov »man-in-the-middle«.
Uporaba zastarelih algoritmov	MD5, SHA-1 in kratki RSA ključi zmanjšujejo varnost.
Uporabniška zahtevnost	PGP je za nekatere uporabnike zapleten, kar omejuje njegovo širšo uporabo v splošni populaciji.

Tabela 32: Pregled pogostih algoritmov v PGP

Namen	Algoritmi
Asimetrično šifriranje	RSA, ElGamal, ECC (novejši)
Simetrično šifriranje	AES, 3DES, CAST5
Zgoščevalne funkcije	SHA-2, SHA-3 (novejši), starejši MD5 (nevaren)
Digitalni podpisi	RSA, DSA, ECDSA

Tabela 33: Povzetek dodatnih mehanizmov

Mehanizem	Glavni namen	Primer uporabe
Hash funkcije	Preverjanje celovitosti podatkov	Kontrolne vsote, preverjanje gesel
HMAC	Preverjanje celovitosti in istovetnosti s skupnim ključem	TLS, VPN, API avtentikacija
Digitalni podpisi	Preverjanje izvora, celovitosti in nezanikljivost	E-podpis, programska oprema, dokumenti
Certifikati (PKI)	Povezava javnega ključa z identiteto	HTTPS, digitalna identiteta, pametne kartice



4.6.5.1 Vaja: Uporaba PGP (Mailvelope) za varno e-pošto v Gmailu

Cilj vaje

Študente naučiti, kako z uporabo Mailvelope in PGP v Gmailu praktično izvedejo ustvarjanje ključev, šifriranje, pošiljanje varne e-pošte in preverjanje digitalnega podpisa, brez uporabe centralnih overiteljev (PKI).

Študent se nauči:

- namestiti in aktivirati Mailvelope kot razširitev za brskalnik,
- ustvariti lasten PGP par ključev (javnega in zasebnega),
- izmenjati javne ključe z drugim uporabnikom,
- sestaviti in poslati šifrirano e-pošto z digitalnim podpisom,
- prejeti, dešifrirati in preveriti podpisano sporočilo,
- razumeti razliko med PGP in PKI (decentraliziran splet zaupanja).

Potrebščine:

- Gmail račun (vsak udeleženec),
- brskalnik (Chrome ali Firefox),
- razširitev Mailvelope (https://mailvelope.com),
- par študentov za delo v paru (pošiljatelj in prejemnik).

Navodila za izvedbo

1. Namestitev Mailvelope

- Odpri https://mailvelope.com in namesti razširitev za brskalnik.
- Po namestitvi se v brskalniku prikaže ikona ključavnice.

2. Ustvarjanje para ključev

- Klikni ikono Mailvelope → »Options« → »Generate Key«.
- Vpiši svoje ime in Gmail naslov ter izberi močno geslo za zaščito zasebnega ključa.
- Ustvari ključ.
- Zapiši **fingerprint** svojega javnega ključa (uporaben za preverjanje).

3. Izmenjava javnih ključev

- Izvozi javni ključ → pošlji ga sošolcu (npr. prek Gmaila ali USB).
- Uvozi prejeti javni ključ sošolca v Mailvelope → »Key Management« → »Import«.
- Preverita fingerprint, da se izognete MITM napadom.

Objava javnega ključa na strežnik

- Pojdi v **Key Management** → **Export** → **Upload to Key Server**.
- Izberi privzeti Mailvelope strežnik (keys.mailvelope.com) ali keys.openpgp.org.
- Potrdi objavo in (če je potrebno) preveri potrditveno e-pošto.
- Tako omogočiš, da lahko kdorkoli poišče tvoj javni ključ po e-pošti.

Pridobivanje javnih ključev druge osebe

- V Key Management klikni »Search on key server«.
- Vpiši e-poštni naslov svojega sošolca.
- Najdi in uvozi njegov javni ključ iz strežnika.
- (Opcijsko) preveri fingerprint prek drugega kanala (npr. osebno), da se izogneš MITM napadom.

4. Pošiljanje šifriranega in podpisanega sporočila

- V Gmailu odpri novo sporočilo.
- Klikni ikono Mailvelope v oknu za pisanje → odpre se varno okno.
- Napiši sporočilo (npr. »Pozdrav, to je šifrirano sporočilo.«).
- Označi **prejemnika**, izberi možnost **šifriranja in podpisovanja**, pošlji.

5. Prejem in dešifriranje

- Sošolec prejme šifrirano sporočilo v Gmailu.
- Klikne ikono Mailvelope → vpiše geslo za zasebni ključ → sporočilo se dešifrira.
- Preveri digitalni podpis (Mailvelope samodejno preveri javni ključ pošiljatelja).

6. Napaka ali sprememba

- Namenoma spremeni en znak v šifriranem sporočilu → prejemnik naj poskusi preveriti podpis.
- Opazuj, kako podpis ne velja več → prikaz zagotavljanja **celovitosti**.

Poročilo

Vsak par pripravi kratko poročilo, ki vsebuje:

- kdo je bil pošiljatelj in kdo prejemnik,
- fingerprint izmenjanih ključev,
- izpis poslanega in prejetega (dešifriranega) sporočila,
- ali je bil podpis uspešno preverjen,
- kratek sklep: kako PGP zagotavlja zaupnost, celovitost in istovetnost brez PKI.

Ključne točke vaje

- PGP uporablja **kombinacijo asimetričnega in simetričnega šifriranja**.
- **Javni ključ** za šifriranje, **zasebni ključ** za dešifriranje.
- Digitalni podpis se izdela z zasebnim ključem pošiljatelja, preveri z javnim ključem.
- V PGP zaupanju ne posreduje CA, ampak uporabniki prek **spleta zaupanja (ang. *Web of Trust*)**.

4.6.5.2 Povzetek

- **PGP** je celovit kriptografski sistem za šifriranje, podpisovanje in upravljanje ključev, posebej uporaben pri zaščiti elektronske pošte in datotek.
- Uporablja **kombinacijo simetričnega in asimetričnega šifriranja** ter **zgoščevalnih funkcij**.
- Omogoča preverjanje izvora, zaščito vsebine in zaznavanje sprememb.
- Namesto centralnih overiteljev uporablja **decentraliziran splet zaupanja**, kjer uporabniki medsebojno potrjujejo identitete.
- Za varno uporabo je nujno varno shranjevanje zasebnih ključev, preverjanje javnih ključev, uporaba sodobnih algoritmov (npr. RSA 2048+, AES, SHA-2) in redno vzdrževanje ključev.
- Glavna prednost PGP je visoka stopnja zasebnosti in neodvisnost od centralnih oblasti; glavna slabost pa je zahtevnejša uporaba za neizkušene uporabnike.

4.6.6 Primerjava med PKI in PGP

Tabela 34

Element	PKI – Public Key Infrastructure	PGP – Pretty Good Privacy
Namen	Centraliziran sistem za upravljanje digitalnih potrdil, identitet in zaupanja.	Decentraliziran sistem za šifriranje, digitalno podpisovanje in upravljanje javnih ter zasebnih ključev brez centralne avtoritete.
Nastanek	Razvit kot del X.509 standardov v začetku 1990-ih.	Philip Zimmermann, 1991.
Model zaupanja	Hierarhični model – na vrhu je korenska CA (Root CA), ki izdaja potrdila podrejenim CA in uporabnikom.	Mreža zaupanja (»web of trust«) – uporabniki med seboj preverjajo in podpisujejo javne ključe, brez osrednje avtoritete.
Izdajanje potrdil / ključev	Digitalna potrdila izdaja certifikacijski organ (CA). Uporabnik prejme certifikat, ki potrjuje istovetnost.	Uporabnik sam generira par ključev. Javni ključ lahko objavi na strežniku, njegova verodostojnost pa je odvisna od podpisov drugih uporabnikov.
Shranjevanje javnih ključev	Javni ključi in potrdila so v varnih repozitorijih (npr. LDAP, Active Directory).	Javni ključi so objavljeni na javnih PGP strežnikih (npr. keys.openpgp.org) ali izmenjani neposredno.
Digitalni podpisi	Podpis temelji na certifikatu, ki ga izda CA; preverjanje je vezano na verigo zaupanja znotraj PKI.	Podpis temelji na zasebnem ključu posameznika; preverjanje poteka na osnovi zaupanja v podpisnika ali verige uporabniških podpisov.
Preklic	Možen z objavo CRL (Certificate Revocation List) ali prek OCSP (Online Certificate Status Protocol).	Uporabnik ustvari preklicno potrdilo (ang. *revocation certificate*) in ga objavi na strežniku ali razpošlje drugim uporabnikom.
Upravljanje sistema	Zahteva celoten ekosistem (CA strežniki, RA, politiko izdajanja, varne naprave).	Upravljanje poteka lokalno pri uporabniku z namenskimi programi (npr. GnuPG).
Integracija	Običajna raba v podjetjih, državni upravi, bančnem sektorju in za HTTPS (TLS).	Pogosta raba pri posameznikih, za varno elektronsko pošto in zaščito datotek.
Obseg uporabe	Dobro deluje v velikih organizacijskih okoljih.	Učinkovit v manjših skupinah ali za osebno rabo; pri večjih sistemih je potrebnega več dela za vzpostavitev zaupanja.
Uporabniška izkušnja	Nastavitve in vzdrževanje zahtevajo strokovno podporo.	Preprostejši za osebno rabo, vendar koncept zaupanja ni vedno samoumeven za neizkušene uporabnike.
Primeri uporabe	eDavki, bančni certifikati, HTTPS spletne strani, S/MIME elektronska pošta, pametne kartice.	Šifriranje datotek in e-pošte, podpisovanje odprtokodne programske opreme, osebna varna komunikacija brez posrednikov.

Tabela 35 : Tipične razlike v praksi

Situacija	PKI	PGP
Prijava v varne sisteme	Uporaba digitalnega potrdila, ki ga izda CA.	Možna, vendar redkeje uporabljena.
Varna elektronska pošta	S/MIME s certifikati PKI infrastrukture.	PGP plug-in (npr. GPG + Thunderbird).
HTTPS spletne strani	SSL/TLS certifikati, izdani pri zaupanja vrednih CA.	PGP se tukaj ne uporablja.
Deljenje javnih ključev	Prek infrastrukture CA in imenikov.	Prek PGP strežnikov ali osebne izmenjave.

Oba sistema temeljita na asimetrični kriptografiji in digitalnih podpisih, vendar uporabljata **različna pristopa k zaupanju**:

- PKI gradi zaupanje od vrha navzdol (hierarhija CA),
- PGP gradi zaupanje med uporabniki (mreža zaupanja).

PKI je zato bolj primeren za uradne in množične primere uporabe, PGP pa za prilagodljivo, osebno ali skupinsko komunikacijo.

4.6.6.1 Povzetek

- **PKI** je formaliziran, centraliziran in primeren za **velike organizacijske sisteme**, kjer je potrebna pravna veljavnost in sistemsko upravljanje identitet.
- **PGP** je bolj decentraliziran, namenjen **posameznikom ali manjšim skupinam**, in omogoča visoko stopnjo zasebnosti brez zaupanja v centralni organ, vendar zahteva aktivno sodelovanje pri vzpostavljanju zaupanja.


MSKU
MAERSK
maerskline.com

Elektronsko poslovanje v logistiki

Gradivo za vaje

KODA

Navodila za računalnik.
Rešujemo probleme.

SINTAKSA
Pravila, po katerih je koda pravilno napisana.

LOGIKA
Zaporedje ukazov, ki vodi do rešitve.

STRUKTURA
Urejanje kode v funkcije, razrede in module.

NAPAKE
Iskanje in odpravljanje napak v kodi.

TESTIRANJE
Preverjanje pravilnosti in zanesljivosti kode.

```python
def izracunaj_skupno(kos, cena):
    skupaj = kos * cena
    popust = 0
    if skupaj > 1000:
        popust = skupaj * 0.1
    return skupaj - popust
kos = 15
cena = 78.50
rezultat = izracunaj_skupno(kos,cena)
print("Skupni znesek:", rezultat)
```

KODA

ANALIZA
Razumemo problem.

NAČRTOVANJE
Zasnova algoritma in poteka.

PISANJE KODE
Pretvorba načrta v kodo.

TESTIRANJE
Preverimo delovanje in odpravimo napake.

IZVEDBA
Rešitev uporabimo v praksi.



Univerzitetna založba
Univerze v Mariboru

5 Koda

Pri vajah s področja kriptografije pogosto uporabljamo praktična orodja, s katerimi preverjamo celovitost in avtentičnost datotek. PowerShell, ki je privzeto nameščen v okolju Windows, omogoča enostavno izvajanje takšnih postopkov brez dodatne programske opreme. V nadaljevanju so prikazane skripte za izračun zgoščevalnih vrednosti z algoritmi MD5, SHA-1 in SHA-256, ter za ustvarjanje in preverjanje HMAC-SHA256. Skripte so kratke, pregledne in primerne za prilagoditev, zato so zelo uporabne pri osnovnih varnostnih pregledih ali preverjanju datotek v učnem okolju.

Pomni!

Zgoščevalne funkcije same po sebi ne zagotavljajo zaščite podatkov, temveč omogočajo preverjanje, ali je bila vsebina spremenjena. Če želimo preveriti tudi izvor podatkov, uporabimo kombinacijo zgoščevalne funkcije in skrivnega ključa – HMAC.

Primer v praksi

Predstavljaj si, da si prejel pomemben dokument po elektronski pošti. Z uporabo PowerShell skripte lahko hitro izračunaš zgoščevalno vrednost (npr. SHA-256) in jo primerjaš z vrednostjo, ki ti jo je poslal pošiljatelj. Če se vrednosti ujemata, veš, da datoteka med prenosom ni bila spremenjena.

5.1 Power Shel skripta za izračun MD5

```
# ============================================
# Skripto pripravil: Mihael Plevnik
# Datum: oktober 2025
# Skripta: Izračun MD5 hash vrednosti datoteke
```

```
# ==========================================

# Pot do datoteke (lahko jo spremeniš)
$filePath = "C:\Datoteka.pdf"

# Preveri, ali datoteka obstaja
if (-Not (Test-Path -Path $filePath)) {
    Write-Host "Datoteka ne obstaja: $filePath" -ForegroundColor Red
    exit
}

# Izračun MD5
$hash = Get-FileHash -Path $filePath -Algorithm MD5

# Izpis rezultata
Write-Host "Datoteka : $($hash.Path)"
Write-Host "MD5 hash: $($hash.Hash)"
```

5.2 Power Shel skripta za izračun SHA-1

```
# ============================================
# Skripto pripravil: Mihael Plevnik
# Datum: oktober 2025
# Skripta: Izračun SHA-1 hash vrednosti datoteke
# ============================================

# Pot do datoteke (lahko jo spremeniš)
$filePath = "C:\Datoteka.pdf"

# Preveri, ali datoteka obstaja
if (-Not (Test-Path -Path $filePath)) {
    Write-Host "Datoteka ne obstaja: $filePath" -ForegroundColor Red
    exit
}

# Izračun SHA-1 hash vrednosti
$hash = Get-FileHash -Path $filePath -Algorithm SHA1

# Izpis rezultata
Write-Host "Datoteka : $($hash.Path)"
Write-Host "SHA-1 hash: $($hash.Hash)"
```

5.3 Power Shel skripta za izračun SHA-256

```
# ==============================================
# Skripto pripravil: Mihael Plevnik
# Datum: oktober 2025
# Skripta: Izračun SHA-256 hash vrednosti datoteke
# ==============================================

# Pot do datoteke (lahko jo spremeniš)
$filePath = "C:\Datoteka.pdf"

# Preveri, ali datoteka obstaja
if (-Not (Test-Path -Path $filePath)) {
    Write-Host "Datoteka ne obstaja: $filePath" -ForegroundColor Red
    exit
}
```

```
# Izračun SHA-256 hash vrednosti
$hash = Get-FileHash -Path $filePath -Algorithm SHA256

# Izpis rezultata
Write-Host "Datoteka : $($hash.Path)"
Write-Host "SHA-256 hash: $($hash.Hash)"
```

5.4 Power Shel skripta za izračun HMAC

```
# ========================================
# Skripto pripravil: Mihael Plevnik
# Datum: oktober 2025
# Skripta: Kreiranje HMACSHA256 hash
# ========================================

# 1 Nastavi ključ in pot do datoteke
$key = [Text.Encoding]::UTF8.GetBytes("MojSkrivniKljuc")
$filePath = "C:\Datoteka.pdf"

# 2 Preberi vsebino datoteke kot bajte
$fileBytes = [System.IO.File]::ReadAllBytes($filePath)

# 3 Ustvari HMAC objekt (SHA256)
$hmac = New-Object System.Security.Cryptography.HMACSHA256
$hmac.Key = $key

# 4 Izračunaj HMAC
$hashBytes = $hmac.ComputeHash($fileBytes)

# 5 Pretvori bajte v hex niz
$hmacHex = [BitConverter]::ToString($hashBytes) -replace "-", ""

# 6 Prikaži rezultat
Write-Output "HMAC-SHA256 za datoteko '$filePath':"
Write-Output $hmacHex
```

5.5 Power Shel skripta za preverjanje HMAC-SHA256 Hash

```
# ========================================
# Skripto pripravil: Mihael Plevnik
# Datum: oktober 2025
# Skripta: Preverjanje HMACSHA256 hash
# ========================================

# 1 Nastavi ključ in pot do datoteke
$key = [Text.Encoding]::UTF8.GetBytes("MojSkrivniKljuc")
$filePath = "C:\Datoteka.pdf"

# 2 Originalni HMAC (hex), ki ga želiš preveriti
$expectedHmac =
"4376D77DA8A6E5C8277D25BD319918232FF0091953BDE55341218C153E698891"

# 3 Preberi datoteko in izračunaj nov HMAC
$fileBytes = [System.IO.File]::ReadAllBytes($filePath)
$hmac = New-Object System.Security.Cryptography.HMACSHA256
$hmac.Key = $key
$hashBytes = $hmac.ComputeHash($fileBytes)
$calculatedHmac = [BitConverter]::ToString($hashBytes) -replace "-", ""
```

```
# 4 Primerjaj vrednosti
if ($calculatedHmac -eq $expectedHmac) {
    Write-Host "Datoteka je avtentična in nespremenjena." -ForegroundColor Green
} else {
    Write-Host "Datoteka je bila spremenjena ali ključ ni pravilen!" -
ForegroundColor Red}
```


MSKU
MAERSK
maerskline.com

Elektronsko poslovanje v logistiki

Gradivo za vaje

ORODJA

Prava orodja.
Bolje rešitve.

UREDNIKI KODE
Pisanje in urejanje kode.
(VS Code, Notepad++, Sublime Text)

VALIDATORJI XML
Preverjanje pravilnosti XML dokumentov.
(XML Validator, xmlLint)

OGLEDOVALNIKI XML
Prikaz in raziskovanje XML podatkov.
(XMLTree, Oxygen XML)

BAZE PODATKOV
Shranjevanje in upravljanje podatkov.
(MySQL, SQLite)

TESTNA OKOLJA
Preizkušanje aplikacij in storitev.
(Postman, SoapUI)

SISTEMI ZA VERZIJE
Sledenje spremembam in sodelovanje.
(Git, GitHub)



OD KODE DO REŠITVE

RAZVOJ
Napišemo kodo in logiko.

PREVERJANJE
Validiramo in preverimo podatke.

TESTIRANJE
Preizkusimo delovanje in rezultate.

SHRANJEVANJE
Podatke varno shranimo.

UPORABA
Rešitve uporabimo v praksi.



Univerzitetna založba
Univerze v Mariboru

6 Uporaba orodij

Pri vajah s področja kibernetske varnosti in zaščite podatkov pogosto uporabljamo praktična orodja, ki omogočajo preverjanje celovitosti datotek, izvajanje šifriranja, delo z zgoščevalnimi funkcijami ter učinkovito upravljanje z datotekami. Njihova prednost je, da so večinoma brezplačna, enostavna za uporabo in ne zahtevajo kompleksne namestitve, zato so zelo primerna za učno okolje in samostojno delo študentov. Orodja, kot so Notepad++, Hash Tool, IT-TOOLS, CrackStation, Mailvelope, VeraCrypt, 7-Zip in Total Commander, pokrivajo različna področja dela, od urejanja skript in preverjanja hash vrednosti do šifriranja komunikacij ter zaščite podatkov na nosilcih.



Pomni!

Pravilna uporaba teh orodij zahteva razumevanje osnovnih varnostnih konceptov, saj lahko napačna nastavitev (npr. izbira šibkega gesla ali neustreznega algoritma) bistveno zmanjša stopnjo zaščite. Orodja so pomoč pri izvajanju postopkov, ne pa nadomestilo za premišljeno varnostno prakso.



Primer v praksi

Predstavljaj si, da želiš poslati zaupno datoteko prek elektronske pošte. S 7-Zip lahko datoteko najprej stisneš in zaščitiš z AES-256 šifriranjem, nato pa preveriš njeno zgoščevalno vrednost s Hash Tool, da zagotoviš celovitost. Če želiš dodatno zaščito, lahko uporabiš Mailvelope za šifriranje e-poštnega sporočila z javnim ključem prejemnika. Na ta način združiš več enostavnih orodij v učinkovit varnostni postopek.

6.1 Notepad ++ (https://notepad-plus-plus.org/)

Notepad++ je brezplačen in odprtokoden urejevalnik besedil, namenjen predvsem delu z izvorno kodo in konfiguracijskimi datotekami (Notepad++, b. d.). Deluje v okolju Windows in je priljubljen zaradi svoje hitrosti, enostavne uporabe ter širokega nabora funkcionalnosti. Uporablja ga veliko programerjev, sistemskih skrbnikov in študentov za pisanje skript, urejanje nastavitev in pregledovanje različnih vrst datotek.

Med pomembnejše zmogljivosti orodja spadajo:

- poudarjanje skladnje (syntax highlighting) za številne programske jezike, kar omogoča boljšo preglednost kode,
- več zavihkov za hkratno urejanje več dokumentov,
- iskanje in zamenjava z regularnimi izrazi, kar močno pospeši delo z večjimi besedili,
- samodejno zapiranje oklepajev in zamikanje kode, kar pomaga pri pisanju skript in programov,
- podpora za vtičnike (plug-ins), s katerimi lahko dodajamo nove funkcije, kot so FTP dostop, preverjanje slovnice ali pretvorbe formatov.

Notepad++ je izjemno uporabno orodje pri tehničnih vajah, saj omogoča hitro in pregledno urejanje PowerShell skript, XML datotek, HTML strani, konfiguracijskih datotek in drugih besedilnih formatov. Zaradi majhne velikosti in hitrega zagona je priročen tudi za uporabo na šolskih računalnikih ali prenosnikih z omejenimi viri.



Pomni!

Čeprav je Notepad++ naprednejši od osnovne Beležnice (ang. Notepad), ne deluje kot polnopravno razvojno okolje (IDE). Njegova prednost je preprostost in prilagodljivost, kar ga naredi odličnega za učenje in osnovno delo s kodo.



Primer v praksi

Pri pisanju PowerShell skripte za izračun SHA-256 lahko datoteko hitro odpremo v Notepad++, uredimo pot do datoteke, shranimo spremembe in skripto takoj zaženemo v PowerShell okolju. Pregledno barvno označevanje ukazov olajša razumevanje in odpravljanje napak.

6.2 Hash Tool 1.2 (https://hash-tool1.software.informer.com/)

Hash Tool 1.2 je preprosto, brezplačno namizno orodje za okolje Windows, namenjeno izračunu zgoščevalnih vrednosti (hash) za različne datoteke (Hash Tool 1.2, b. d.). Uporablja se predvsem za preverjanje celovitosti datotek po prenosu, varnostne preglede ali učenje osnov kriptografskih postopkov. Program je lahek, ne zahteva namestitve kompleksnih komponent in je primeren tudi za uporabo v učilniškem okolju.

Med njegovimi glavnimi značilnostmi so:

- podpora za različne algoritme, kot so MD5, SHA-1, SHA-256 in drugi,
- enostavna uporaba, saj uporabnik samo izbere datoteko, orodje pa samodejno izračuna izbrane zgoščevalne vrednosti,
- možnost primerjave, saj program omogoča, da vnesemo pričakovano zgoščevalno vrednost in preverimo ujemanje,
- kopiranje rezultatov, saj je izračunane vrednosti mogoče hitro kopirati v odložišče za nadaljnjo uporabo ali dokumentacijo.

Hash Tool 1.2 je posebej priročen za začetnike in študente, saj omogoča hitro razumevanje delovanja zgoščevalnih funkcij brez potrebe po uporabi ukazne vrstice ali programiranja. Prav tako je uporaben pri preverjanju prenosov programske opreme, kjer ponudniki pogosto objavijo pričakovane zgoščevalne vrednosti datotek (npr. SHA-256), da lahko uporabniki preverijo, ali datoteka ni bila spremenjena.



Pomni!

Če se izračunana zgoščevalna vrednost ne ujema z objavljeno, datoteka morda ni avtentična ali je bila med prenosom spremenjena. V takem primeru je prenos treba ponoviti ali datoteko preveriti iz drugega vira.



Primer v praksi

Študent prenese namestitveno datoteko odprtokodnega programa. Na spletni strani je objavljena tudi njena SHA-256 vrednost. Datoteko odpre v programu Hash Tool 1.2, izračuna vrednost in jo primerja z objavljeno. Ker se obe vrednosti ujemata, lahko zanesljivo potrdi, da je datoteka nepoškodovana in pristna.

6.3 IT – TOOLS (https://it-tools.tech/)

IT-TOOLS je brezplačno spletno orodje, ki združuje številne uporabne funkcije za delo z besedili, podatki in kriptografskimi postopki (IT-TOOLS, b. d.). Deluje neposredno v spletnem brskalniku, zato ni potrebna nobena namestitev ali prijava, kar ga naredi zelo priročnega za uporabo v učilnici, domačem okolju ali pri hitrem preverjanju različnih postopkov.

Spletna stran je pregledno razdeljena po tematskih sklopih, med katerimi so:

- hash funkcije – izračun zgoščevalnih vrednosti (MD5, SHA-1, SHA-256 in drugih) za poljubno besedilo ali datoteke,
- kodiranje in dekodiranje – pretvorbe med različnimi formati (Base64, URL, HTML, JSON ipd.),
- kriptografska orodja – enostavno šifriranje in dešifriranje besedila ter preverjanje digitalnih podpisov,
- različni pripomočki – generatorji gesel, UUID, pretvorniki številskih sistemov, analizatorji JSON in druge praktične funkcije.

Ena izmed največjih prednosti IT-TOOLS je, da deluje brez namestitve, takoj v brskalniku, in je zato odličen pripomoček za hitre preizkuse ali razlago konceptov med predavanji. Študentom omogoča takojšnje eksperimentiranje z algoritmi, kar olajša razumevanje delovanja postopkov, kot so zgoščevanje, kodiranje in osnovno šifriranje.



Pomni!

Ker gre za spletno orodje, ne shranjuj zaupnih podatkov ali gesel. Uporabno je predvsem za učenje, preizkušanje in preverjanje, ne pa za obdelavo občutljivih informacij.



Primer v praksi

Med vajami študent vnese besedilo »test123« v zavihek za hash funkcije in izračuna SHA-256 vrednost. Nato spremeni en znak v besedilu in znova izračuna rezultat. Tako takoj opazi, da se zgoščena vrednost popolnoma spremeni, kar nazorno ponazarja občutljivost zgoščevalnih funkcij na spremembe podatkov.

6.4 CrackStation (https://crackstation.net/)

CrackStation je brezplačno spletno orodje za razbijanje zgoščevalnih vrednosti z uporabo velikih baz podatkov pogostih gesel in njihovih zgoščenk (CrackStation, b. d.). Namenjeno je predvsem preverjanju moči gesel ter prikazu, kako enostavno je mogoče odkriti izvirno geslo, če je to šibko ali pogosto uporabljeno. Orodje temelji na uporabi t. i. »rainbow« tabel (vnaprej pripravljenih seznamov gesel in njihovih hash vrednosti), kar omogoča zelo hitro primerjanje in razkrivanje ujemanj.

Med glavnimi značilnostmi CrackStation so:

- hitra primerjava z ogromno zbirko znanih hash vrednosti za MD5, SHA-1 in SHA-256,
- podpora za več algoritmov – omogoča preverjanje različnih vrst hashov brez dodatnih nastavitev,
- takojšnji rezultati – če se hash nahaja v bazi, orodje v nekaj trenutkih prikaže izvirno besedilo (npr. geslo) in
- dostop prek brskalnika, brez namestitve ali registracije.

CrackStation je pogosto uporabljen pri izobraževanju in varnostnih preverjanjih, saj zelo nazorno pokaže, kako nevarna je uporaba preprostih gesel in neustrezno zaščitenih hashov. Uporablja se lahko tudi za preverjanje, ali je določena zgoščevalna vrednost že javno znana, kar je uporabno pri testiranju varnosti informacijskih sistemov.



Pomni!

CrackStation ne izvaja napada z ugibanjem v realnem času, temveč uporablja obsežne vnaprej pripravljene sezname hashov. Če gesla ni v bazi, rezultat ne bo najden, kar pa ne pomeni, da je varno, temveč le, da ni med najpogostejšimi.



Primer v praksi

Študent vnese hash 5f4dcc3b5aa765d61d8327deb882cf99 v CrackStation. Orodje takoj prikaže, da ta vrednost ustreza geslu “password”. Ta primer jasno pokaže, kako hitro je mogoče razbiti preprosta gesla in zakaj je priporočljiva uporaba dolgih, naključnih gesel skupaj s postopkom soljenja (ang. *salting*).

6.5 Mailvelope (https://mailvelope.com/en)

Mailvelope je brezplačna in odprtokodna razširitev za spletne brskalnike (npr. Chrome, Firefox), ki omogoča enostavno šifriranje in podpisovanje elektronske pošte z uporabo standarda OpenPGP (Mailvelope, b. d.). Namenjena je predvsem uporabnikom spletnih e-poštnih storitev, kot so Gmail, Outlook.com ali Yahoo, ki želijo izboljšati zasebnost in varnost svojih sporočil brez zapletenih namestitev dodatne programske opreme.

Glavne značilnosti orodja Mailvelope vključujejo:

- podporo za OpenPGP – omogoča ustvarjanje, uvoz in upravljanje parov javnega in zasebnega ključa neposredno v brskalniku,
- šifriranje in podpisovanje e-pošte – uporabnik lahko pred pošiljanjem sporočila enostavno zašifrira vsebino ali jo digitalno podpiše, kar zagotavlja zaupnost in preverjanje pristnosti,
- integracijo s spletnimi poštnimi vmesniki – Mailvelope doda varnostna orodja neposredno v uporabniški vmesnik znanih spletnih poštnih storitev,
- enostavno uporabo – intuitiven vmesnik omogoča delo tudi začetnikom, brez potrebe po ročnem delu z ukazno vrstico ali zunanjimi programi.

Mailvelope je še posebej uporaben pri učnih vajah o šifriranju komunikacij, saj omogoča praktičen prikaz delovanja javno-zasebnih ključev, digitalnih podpisov in varne elektronske pošte v realnem okolju. Zaradi razširitvene zasnove deluje neodvisno od strežnikov posameznih ponudnikov e-pošte, kar povečuje zasebnost uporabnika.



Pomni!

Za uspešno šifriranje mora prejemnik imeti svoj javni ključ, ki ga pošiljatelj uvozi v Mailvelope. Brez tega sporočila ni mogoče zašifrirati. Pomembno je tudi, da uporabniki ustrezno zaščitijo svoj zasebni ključ z varnim geslom.



Primer v praksi

Študent želi poslati sošolcu zaupno datoteko prek Gmaila. Oba imata v Mailvelope ustvarjena OpenPGP para ključev in izmenjana javna ključa. Študent v Mailvelope napiše sporočilo, izbere šifriranje in ga pošlje. Sošolec lahko sporočilo odpre in dešifrira le s svojim zasebnim ključem, kar zagotavlja, da nihče drug med prenosom ne more prebrati vsebine.

6.6 VeraCrypt (https://veracrypt.jp/en/Home.html)

VeraCrypt je brezplačno in odprtokodno orodje za šifriranje podatkov, ki omogoča zaščito posameznih datotek, celih particij ali celo celotnih diskov z naprednimi šifrirnimi algoritmi (npr. AES, Serpent, Twofish) (VeraCrypt, b. d.). Namenjeno je uporabnikom, ki želijo zagotoviti visoko stopnjo zaupnosti in zaščite svojih podatkov pred nepooblaščenim dostopom — tako na osebnih računalnikih kot na prenosnih USB-ključih ali zunanjih diskih.

Glavne značilnosti orodja VeraCrypt vključujejo:

- močno šifriranje podatkov – VeraCrypt podpira več uveljavljenih šifrirnih algoritmov in njihove kombinacije (npr. AES-Twofish-Serpent), kar zagotavlja visoko stopnjo varnosti,
- ustvarjanje šifriranih vsebnikov – uporabnik lahko ustvari šifrirano datoteko (t. i. volume), ki se v sistemu prikaže kot navidezni disk. Vanjo lahko shranjuje datoteke kot na običajen pogon,
- šifriranje particij in celotnega diska – VeraCrypt omogoča tudi šifriranje sistemskega diska, kar zagotavlja zaščito podatkov že ob zagonu računalnika,
- skrivni (hidden) volume – napredna možnost, ki omogoča ustvarjanje "volumna znotraj volumna" za dodatno plast zasebnosti v primerih prisile,
- večplatformsko podporo – na voljo za Windows, macOS in Linux, kar omogoča uporabo šifriranih vsebnikov na različnih napravah.

VeraCrypt je izjemno uporaben pri učnih vajah, kjer želimo prikazati praktične načine zaščite podatkov pri shranjevanju ali prenosu na zunanjih nosilcih. Uporabnikom omogoča, da zavarujejo občutljive datoteke brez potrebe po dostopu do interneta ali zunanjih strežnikov, saj se šifriranje izvaja lokalno na napravi.



Pomni!

Za dostop do šifriranih podatkov je potrebno poznati geslo (ali uporabiti ključno datoteko). Če uporabnik pozabi geslo, dostop do podatkov ni več mogoč — ni »stranskih vrat«. Pomembno je izbrati močno in edinstveno geslo ter ga varno hraniti. Pri šifriranju sistemskega diska je priporočljivo predhodno narediti varnostne kopije.

Primer v praksi

Študent želi zaščititi projektno dokumentacijo na USB-ključku. Z VeraCrypt ustvari šifriran vsebnik velikosti 2 GB, nastavi močno geslo in vanj shrani vse datoteke. Ob izgubi USB-ključka nihče brez gesla ne more dostopati do vsebine, saj je celotna vsebina šifrirana. V šoli lahko vsebnik "odklene" z VeraCrypt in do datotek dostopa kot do običajnega pogona.

6.7 7-Zip (https://www.7-zip.org/)

7-Zip je brezplačen in odprtokoden pripomoček za stiskanje in stiskanje-razstiskanje datotek, ki poleg tega podpira tudi šifriranje stisnjenih arhivov z močnim algoritmom AES-256 (7-Zip, b. d.). Namenjen je uporabnikom, ki želijo učinkovito zmanjšati velikost datotek, hkrati pa po potrebi zagotoviti zaščito podatkov z geslom.

Glavne značilnosti orodja 7-Zip vključujejo:

- stiskanje in razstiskanje – podpira svojo lastno .7z obliko (ki običajno ponudi boljšo stiskanje) ter številne druge arhivske formate, kot so ZIP, TAR, GZIP, BZIP2, RAR (le razstiskanje) in drugi,
- šifriranje arhivov – pri ustvarjanju .7z ali .zip arhiva lahko uporabnik nastavi geslo in uporabi AES-256 (pri .7z) za zaščito vsebine,
- visoko stiskalno učinkovitost – 7-Zip pogosto dosega boljše razmerje stiskanja kot mnoga druga orodji, zlasti pri velikih in homogeno strukturiranih datotekah,
- integracijo v sistem – v operacijskem sistemu Windows se doda v kontekstni meni (desni klik) za hitro kompresijo / dekompresijo,
- večplatformsko alternativo (p7zip) – za Linux / macOS obstaja programska različica p7zip, ki omogoča uporabo funkcionalnosti 7-Zip tudi na teh operacijskih sistemih.

7-Zip je zelo uporaben pri učnih vajah, ko želimo prikazati združevanje, stiskanje in zaščito datotek — z nastavitvijo gesla študenti lahko naučijo, kako stisnjeni arhivi (npr. .7z) ponujajo varnost, kot dodatni sloj zaščite.

Pomni!

Za dešifriranje zaščitenega arhiva je potrebno poznati geslo. Če ga uporabnik izgubi, ni mogoče dostopati do vsebine arhiva. Šifriranje v .zip arhivih, ki jih ustvari 7-Zip, uporablja šibkejše varnostne metode kot šifriranje AES-256 v .7z arhivih. Zato je priporočljivo, da kadar je mogoče, uporabimo format .7z z AES-256 šifriranjem.

Primer v praksi

Študent želi poslati zbirko dokumentov preko e-pošte, vendar želi zagotoviti, da jih vmesni strežniki ne bodo mogli prebrati. Ustvari arhiv .7z, nastavi geslo in šifriranje z AES-256 ter pošlje datoteko. Prejemnik, ki pozna geslo, arhiv odpre z 7-Zip (ali združljivim programom) in si ogleda vsebino.

6.8 Total Commander (https://www.ghisler.com/)

Total Commander je zmogljiv in prilagodljiv upravljalnik datotek za operacijski sistem Windows, ki omogoča učinkovito delo z datotekami in mapami ter vključuje številna napredna orodja, kot so stiskanje, šifriranje, sinhronizacija map in povezovanje z omrežnimi strežniki. Namenjen je uporabnikom, ki želijo imeti večji nadzor nad datotečnim sistemom in dodatne funkcionalnosti, ki jih standardni Windows Explorer ne ponuja (Total Commander, b. d.).

Glavne značilnosti orodja Total Commander vključujejo:

- dvookenski vmesnik – omogoča hkratni prikaz dveh map, kar olajša kopiranje, premikanje in primerjanje vsebin,
- vgrajeno podporo za arhive – uporabnik lahko odpira in ustvarja različne arhive (.zip, .rar, .7z ipd.) neposredno v programu, brez potrebe po zunanjem orodju,
- šifriranje arhivov – pri ustvarjanju .zip arhivov omogoča nastavitev gesla in šifriranje vsebine, kar poveča varnost pri pošiljanju ali shranjevanju podatkov,
- napredna iskalna orodja – vključuje zmogljiv iskalnik z možnostjo uporabe filtrov, regularnih izrazov in iskanja po vsebini datotek,
- vtičnike in razširitve – uporabnik lahko funkcionalnosti razširi z dodatnimi moduli (npr. za FTP/SFTP povezave, obdelavo slik, branje posebnih formatov ipd.).
- sinhronizacijo map in delo v omrežjih – omogoča primerjanje map, sinhronizacijo datotek ter povezovanje z FTP/SFTP strežniki za prenos podatkov.

Total Commander je še posebej uporaben pri učnih vajah, kjer študenti spoznavajo delo z datotečnimi sistemi, arhiviranje, šifriranje in varno izmenjavo podatkov. Zaradi jasne strukture in podpore vtičnikom omogoča fleksibilno uporabo tako za začetnike kot za napredne uporabnike.



Pomni!

Čeprav Total Commander omogoča osnovno šifriranje arhivov (npr. .zip z geslom), to ne zagotavlja enake stopnje zaščite kot specializirana orodja, kot je VeraCrypt. Gesla naj bodo močna, arhivi pa naj bodo po možnosti dodatno zaščiteni (npr. z uporabo .7z formata z AES-256, ustvarjenega z 7-Zip).



Primer v praksi

Študent želi hitro pripraviti zbirko datotek za prenos na zunanji disk. V Total Commanderju izbere mapo, ustvari .zip arhiv z geslom in datoteko shrani na USB-ključek. Tako je vsebina arhiva zaščitena pred nepooblaščenim odpiranjem, postopek pa je opravljen neposredno v programu brez dodatnih orodij.


MSKU
MAERSK
maerskline.com

Elektronsko poslovanje v logistiki

Gradivo za vaje

VIRI

Zanesljivi viri.
Znanje v praksi.

UČBENIKI
Temeljna literatura za razumevanje konceptov.

ZNANSTVENI ČLANKI
Aktualne raziskave in strokovne analize.

SPLETNI VIRI
Strokovne spletne strani, portali in baze znanja.

STANDARDI IN SMERNICE
Mednarodni standardi in najboljše prakse.

MULTIMEDIJSKA GRADIVA
Videi, predavanja in interaktivne vsebine.



PREDNOSTI UPORABE KAKOVOSTNIH VIROV

- Poglobljeno razumevanje.
- Točnejše rešitve.
- Boljše odločitve v praksi.
- Večja zanesljivost in varnost.
- Razvoj strokovnih kompetenc.



Univerzitetna založba
Univerze v Mariboru

Viri in literatura

7-Zip. (b. d.). 7-Zip. https://www.7-zip.org/

Blažič, A. J., Bjelica, B., Tomažič, B., Tovornik, D., Zupančič, D., Janc, I., Kafadar, M., Švarc, M., Grčman, S., Saljić, S. in Smrekar, Z. (2010). Priporočila za uporabo standarda GZS e-SLOG 1.5 za enostavni račun (verzija 2010/1_0). Gospodarska zbornica Slovenije. http://e-slog.gzs.si

Chamishka, S. (2019, 13. oktober). Public key infrastructure. Medium. https://sadilchamishka.medium.com/public-key-infrastructure-230621274a82

Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R. in Polk, W. (2008). RFC 5280: Internet X.509 public key infrastructure certificate and certificate revocation list (CRL) profile. Internet Engineering Task Force. https://datatracker.ietf.org/doc/html/rfc5280

CrackStation. (b. d.). CrackStation. https://crackstation.net/

Diffie, W. in Hellman, M. (1976). New directions in cryptography. IEEE Transactions on Information Theory, 22(6), 644–654. https://ee.stanford.edu/~hellman/publications/24.pdf

Dykes, L. in Tittel, E. (2005). XML for dummies (4. izd.). Wiley Publishing.

Encyclopaedia Britannica. (2025a). Caesar cipher. https://www.britannica.com/topic/Caesar-cipher

Encyclopaedia Britannica. (2025b). Enigma (German code device). https://www.britannica.com/topic/Enigma-German-code-device

Encyclopaedia Britannica. (2025c). Vigenère cipher. https://www.britannica.com/topic/Vigenere-cipher

Gospodarska zbornica Slovenije. (2020a). e-SLOG 2.0 – Elektronski račun – 1. del: Splošna navodila. Gospodarska zbornica Slovenije. https://www.epos.si

Gospodarska zbornica Slovenije. (2020b). e-SLOG 2.0 – Elektronski račun – 2. del: Sestava dokumenta. Gospodarska zbornica Slovenije. https://www.epos.si

Gospodarska zbornica Slovenije. (2020c). eSLOG 2.0 – Splošna navodila za elektronsko naročilo, elektronsko potrditev naročila in elektronsko dobavnico. Gospodarska zbornica Slovenije. https://www.epos.si

Hash Tool 1.2. (b. d.). Software Informer. https://hash-tool1.software.informer.com/

IT-TOOLS. (b. d.). IT-TOOLS. https://it-tools.tech/

Jereb, B. (2019). Informatika in informacijska varnost: repetitorij. Univerzitetna založba Univerze v Mariboru. https://doi.org/10.18690/978-961-286-251-0

Krawczyk, H., Bellare, M. in Canetti, R. (1997). RFC 2104: HMAC – keyed-hashing for message authentication. Internet Engineering Task Force. https://www.rfc-editor.org/rfc/rfc2104

Mailvelope. (b. d.). Mailvelope. https://mailvelope.com/en

Mucklow, T. J. (2015). The SIGABA/ECM II cipher machine: »A beautiful idea«. Center for Cryptologic History, National Security Agency. https://www.nsa.gov/portals/75/documents/about/cryptologic-heritage/historical-figures-publications/publications/technology/The_SIGABA_ECM_Cipher_Machine_A_Beautiful_Idea3.pdf

Namecheap. (b. d.). What is an SSL certificate and what is it used for? https://www.namecheap.com/support/knowledgebase/article.aspx/786/33/what-is-an-ssl-certificate-and-what-is-it-used-for/

National Institute of Standards and Technology. (2001). FIPS PUB 197-upd1: Advanced Encryption Standard (AES). https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197-upd1.pdf

National Institute of Standards and Technology. (2015). FIPS PUB 180-4: Secure hash standard (SHS). https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf

National Institute of Standards and Technology. (2017). NIST Special Publication 800-12 Rev. 1: An introduction to information security. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-12r1.pdf

Notepad++. (b. d.). Notepad++. https://notepad-plus-plus.org/

OpenSSL. (b. d.). OpenSSL: Cryptography and SSL/TLS toolkit. https://www.openssl.org/
Petitcolas, F. A. P. (2011). Kerckhoffs' principle. V H. C. A. van Tilborg in S. Jajodia (ur.), Encyclopedia of cryptography and security (str. 675). Springer. https://link.springer.com/rwe/10.1007/978-1-4419-5906-5_487
Raicea, R. (2017, 8. oktober). How Pretty Good Privacy works, and how you can use it for secure communication. freeCodeCamp. https://www.freecodecamp.org/news/how-does-pretty-good-privacy-work-3f5f75ecea97/
Rescorla, E. (2018). RFC 8446: The Transport Layer Security (TLS) protocol version 1.3. Internet Engineering Task Force. https://datatracker.ietf.org/doc/html/rfc8446
Shannon, C. E. (1949). Communication theory of secrecy systems. Bell System Technical Journal, 28(4), 656–715. https://www.cs.virginia.edu/~evans/greatworks/shannon1949.pdf
SI-TRUST – Državni center za storitve zaupanja. (2025). Elektronska identiteta in elektronski podpis. https://www.si-trust.gov.si
Stevens, M., Bursztein, E., Karpman, P., Albertini, A. in Markov, Y. (2017). The first collision for full SHA-1. CWI Amsterdam in Google Research. https://shattered.io/static/shattered.pdf
Total Commander. (b. d.). Total Commander. https://www.ghisler.com/
VeraCrypt. (b. d.). VeraCrypt. https://veracrypt.jp/en/Home.html
W3Schools. (b. d.). XML tutorial. https://www.w3schools.com/xml/default.asp
Weaver, C. (2017). 160 bits … SHAttered!!! IT Dojo. https://www.itdojo.com/160-bits-shattered/

ELEKTRONSKO POSLOVANJE V LOGISTIKI: GRADIVO ZA VAJE

MIHAEL PLEVNIK,[1,2] BORUT JEREB[1]

[1] Univerza v Mariboru, Fakulteta za logistiko, Celje, Slovenija
mihael.plevnik2@um.si, borut.jereb@um.si
[2] Slovenska vojska, Ljubljana, Slovenija
mihael.plevnik@mors.si

"Elektronsko poslovanje v logistiki" je gradivo, ki sta ga pripravila avtorja Mihael Plevnik in Borut Jereb za predmet Elektronsko poslovanje v logistiki na Fakulteti za logistiko Univerze v Mariboru. Učbenik povezuje teoretični uvod s praktičnimi vajami s področja elektronskega poslovanja, standardiziranih podatkovnih formatov (npr. XML, XPath) ter digitalne varnosti, vključno z osnovnimi kriptografskimi mehanizmi za varno izmenjavo informacij, kot so šifriranje, izračun zgoščevalnih vrednosti in digitalno podpisovanje. Namenjen je postopnemu pridobivanju znanja in praktičnih spretnosti, ki jih bodoči strokovnjaki potrebujejo v sodobnem, digitaliziranem logističnem okolju, pridobljeno znanje pa jih usposobi za samostojno izvajanje osnovnih postopkov zaščite podatkov ter uporabo izbranih programskih orodij. Gradivo je namenjeno predvsem podpori pedagoškemu procesu in samostojnemu učenju, hkrati pa lahko služi tudi kot referenčni priročnik pri kasnejšem strokovnem delu na področju informacijske varnosti v logistiki.

DOI
https://doi.org/10.18690/um.fl.9.2026

ISBN
978-961-299-216-3

Ključne besede:
elektronsko poslovanje, logistika, informacijska varnost, kriptografija, digitalni podpis, XML

Univerz tetna založba Univerze v Mariboru

DOI
https://doi.org/
10.18690/um.fl.9.2026

ISBN
978-961-299-216-3

Electronic Business in Logistics: Exercise Materials

Mihael Plevnik,[1,2] Borut Jereb[1]

[1] University of Maribor, Faculty of Logistics, Celje, Slovenia
mihael.plevnik2@um.si, borut.jereb@um.si
[2] Slovenian Armed Forces, Ljubljana, Slovenia
mihael.plevnik@mors.si

Keywords:
electronic business,
logistics,
information security,
cryptography,
digital signature,
XML

"Electronic Business in Logistics" is a set of course materials by Mihael Plevnik and Borut Jereb for the course Electronic Business in Logistics at the Faculty of Logistics, University of Maribor. The textbook combines a theoretical introduction with hands-on exercises covering electronic business, standardized data formats (e.g., XML, XPath), and digital security, including basic cryptographic mechanisms for secure information exchange such as encryption, hash calculation, and digital signatures. It is designed to help students progressively acquire the knowledge and practical skills needed in today's digitalized logistics environment, enabling them to independently perform basic data-protection procedures and use selected software tools. The material primarily supports classroom and self-directed learning, while also serving as a reference guide for later professional work in information security within logistics.

University of Maribor Press

Univerza v Mariboru

Fakulteta za logistiko

