

Otvoreno računarstvo

9. Sigurnost u otvorenim sustavima

- Autentikacija i autorizacija
- HTTP sheme autentikacije
- Tokeni (JWT)
- Sigurni komunikacijski kanal (TLS/SSL, HTTPS)
- Delegiranje prava (OAuth2)
- Autentikacija (OpenIDConnect, SAML)

Creative Commons



[Otvoreno računarstvo 2020/21](#) by Ivana Bosnić & Igor Čavrak, FER
is licensed under [CC BY-NC-SA 4.0](#)

Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0)

This license requires that reusers give credit to the creator.

It allows reusers to distribute, remix, adapt, and build upon the material in any medium or format, for noncommercial purposes only.

If others modify or adapt the material, they must license the modified material under identical terms.

BY: Credit must be given to you, the creator.

NC: Only noncommercial use of your work is permitted.

SA: Adaptations must be shared under the same terms.

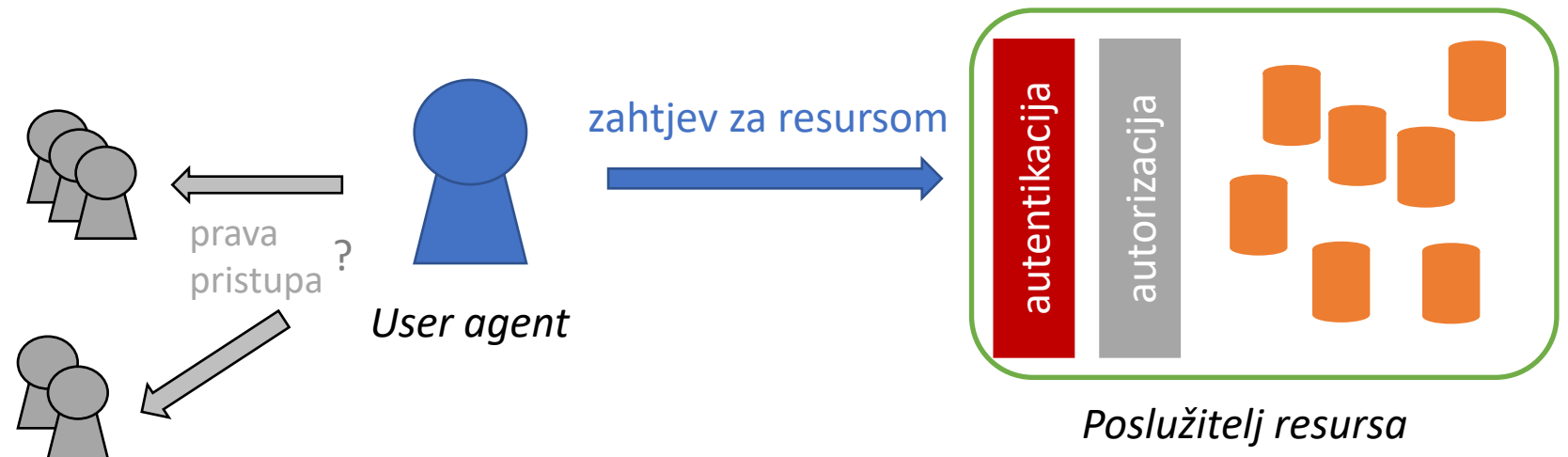
Otvoreno računarstvo

9. Sigurnost u otvorenim sustavima

- Autentikacija i autorizacija
- HTTP sheme autentikacije
- Tokeni (JWT)
- Sigurni komunikacijski kanal (TLS/SSL, HTTPS)
- Delegiranje prava (OAuth2)
- Autentikacija (OpenIDConnect)

Autentikacija i autorizacija (I)

- **Autentikacija** – postupak dokazivanje identiteta
- **Autorizacija** – postupak određivanje prava pristupa resursu
- Klasičan pristup autentikaciji i autorizaciji
 - Svaki *poslužitelj resursa* ima svoju implementaciju i autentikacije i autorizacije
 - Problem broja vjerodajnica (npr. parova *username-password*) koje klijent mora koristiti



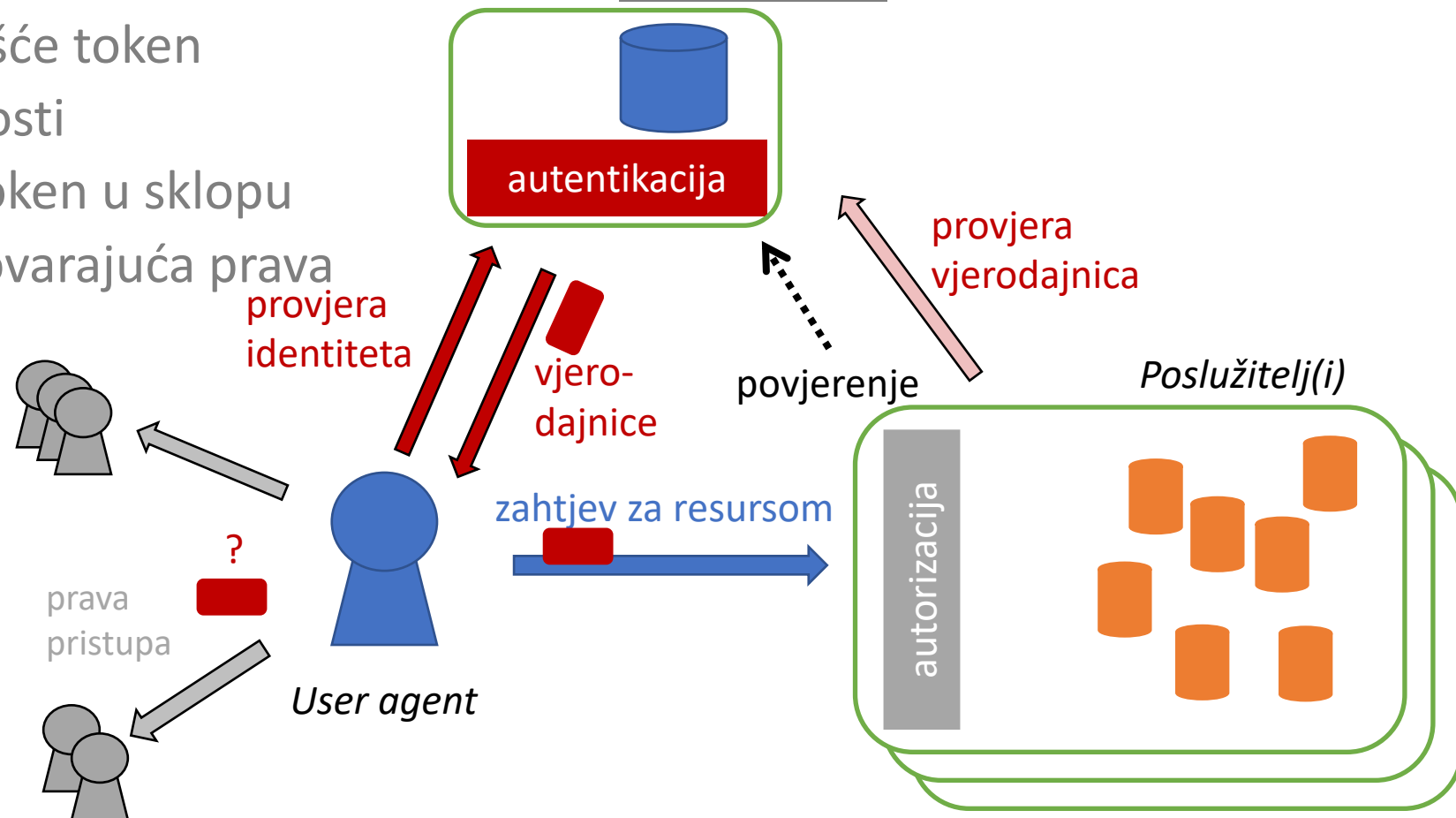
Autentikacija i autorizacija (II)

- Problemi s dijeljenjem autentikacijskih podataka vlasnika resursa trećoj strani:
 - Povjerljivost podataka, rizik krađe podataka (kopije podataka na više mjesta ...)
 - Upravljanje dostupnošću resursa u vremenu (kako povući prava pristupa trećoj strani?)
 - Upravljanje dostupnošću podskupu resursa
 - Upravljanje ovlastima nad resursom (kako upravljati dozvolama za određene akcije nad resursom ili skupom resursa trećoj strani?)

Autentikacija i autorizacija (III)

- Autentikacija na zasebnoj komponenti sustava – pružatelju identiteta (identity provider)

- Uspješna provjera identiteta – izdavanje vjerodajnica o identitetu
- Oblik vjerodajnica – najčešće token ograničenog vijeka valjanosti
- Svatko tko priloži valjani token u sklopu pristupa resursu ima odgovarajuća prava
- Poslužitelji resursa vjeruju pružatelju identiteta
 - potpisane vjerodajnice
 - provjera vjerodajnica kod pružatelja identiteta



Autentikacija i autorizacija (IV)

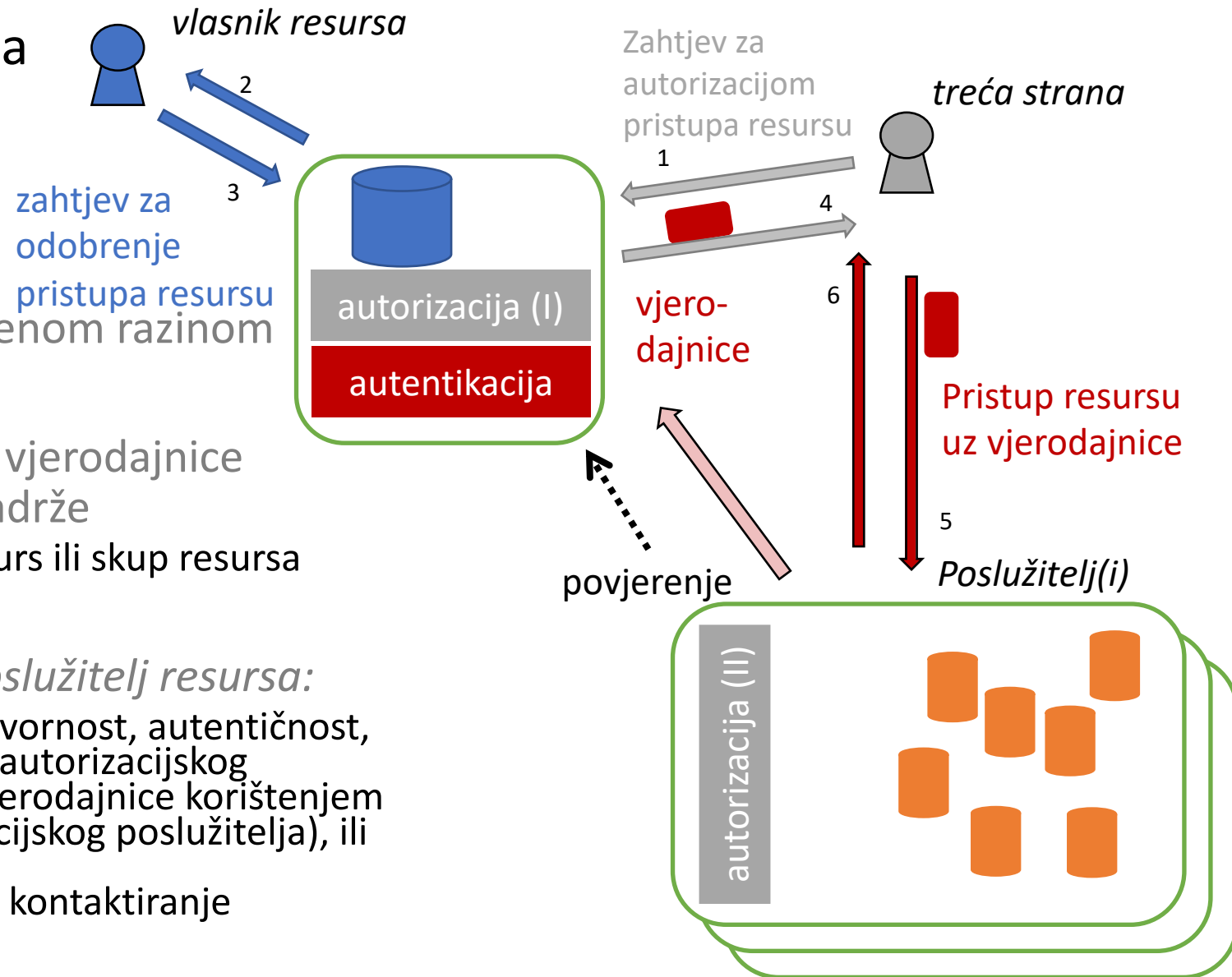
- Prethodna metoda omogućuje

- Korištenje istog identiteta na više poslužitelja resursa v
- Dijeljenje (privremeno) prava pristupa resursima, ali s punim ovlastima korisnika koji dijeli pravo (npr. vlasnik resursa ima sva prava nad resursom) – što ako to nije poželjno?
- Identifikacija, vjerodajnice, ... -> sigurnosno kritične informacije, kako ih zaštititi tijekom prenošenja komunikacijskim kanalom?

Autentikacija i autorizacija (V)

- Treća strana traži dozvolu pristupa resursu ili grupi resursa

- autorizacijski poslužitelj posreduje zahtjev prema vlasniku resursa koji odobrava ili odbija (engl. *consent*) pristup s traženom razinom prava
- *Autorizacijski poslužitelj* (ne) izdaje vjerodajnice o autorizaciji trećoj strani, a koje sadrže
 - razinu prava korisnika za traženi resurs ili skup resursa
 - vrijeme valjanosti vjerodajnice
- Na dospijeće zahtjeva za pristup *poslužitelj resursa*:
 - provjerava valjanost vjerodajnice (izvornost, autentičnost, period valjanosti) bez kontaktiranja autorizacijskog poslužitelja (npr. provjera potpisa vjerodajnice korištenjem certificiranog javnog ključa autorizacijskog poslužitelja), ili
 - provjerava valjanost vjerodajnice uz kontaktiranje autorizacijskog poslužitelja

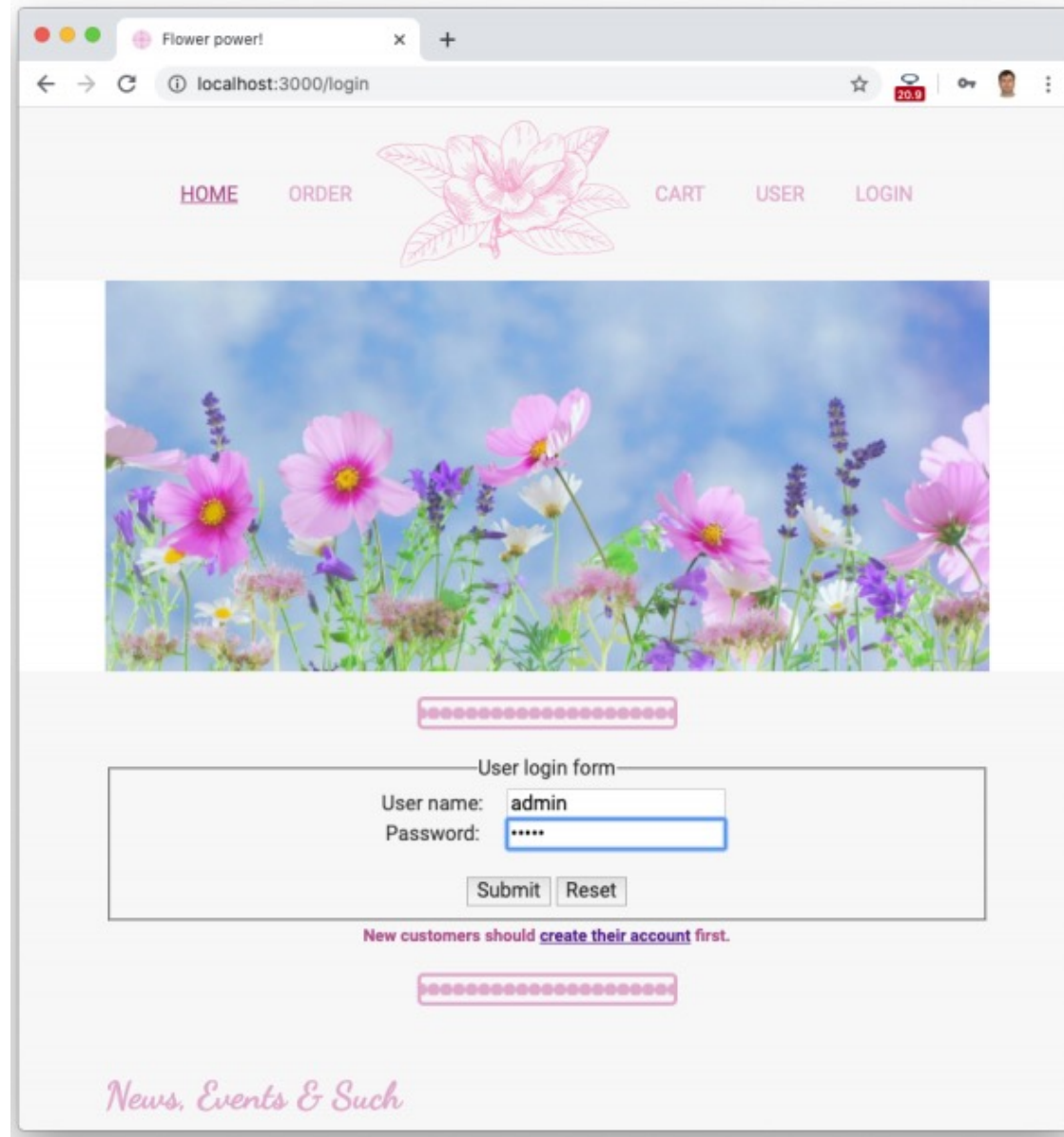


Otvoreno računarstvo

9. Sigurnost u otvorenim sustavima

- Autentikacija i autorizacija
- HTTP sheme autentikacije
- Tokeni (JWT)
- Sigurni komunikacijski kanal (TLS/SSL, HTTPS)
- Delegiranje prava (OAuth2)
- Autentikacija (OpenIDConnect)

HTTP + HTML obrasci (I)



Flower power!

localhost:3000/login

HOME ORDER CART USER LOGIN

User login form

User name: admin

Password:

Submit Reset

New customers should [create their account first](#).

News, Events & Such

HTTP + HTML obrasci (II)

- **Postupak**

- Nakon neuspjelog pristupa štićenom resursu poslužitelj vraća klijentu *login HTML* stranicu s obrascem
- Korisnik popunjava obrazac, podaci se šalju poslužitelju
- Poslužitelj provjerava identitet korisnika (dijeljena tajna) i prava pristupa resursu, stvara zapis o sjednici, vraća identifikacijski podatak korisniku (kolačić)
- Klijent zaprima odgovor poslužitelja zajedno s identifikacijskim podatkom (kolačić)
- Klijent koristi identifikacijski podatak (kolačić) kod svakog sljedećeg pristupa poslužitelju

- **Prednosti**

- aplikacijski specifična implementacija
- implementacija autentikacija i autorizacija korisnika na istom mjestu

- **Nedostaci:**

- aplikacijski specifična (*ad hoc*) implementacija
- podrazumijeva akciju korisnika na strani programa klijenta
- Uvodi se čuvanje stanja između HTTP transakcija (čuvanje podataka o stanju na poslužitelju)
- podložno *phishing* napadima (glumljenje login stranice web sjedišta)
- Zahtijeva korištenje sigurnog komunikacijskog kanala (HTTPS) radi očuvanja povjerljivosti dijeljene tajne

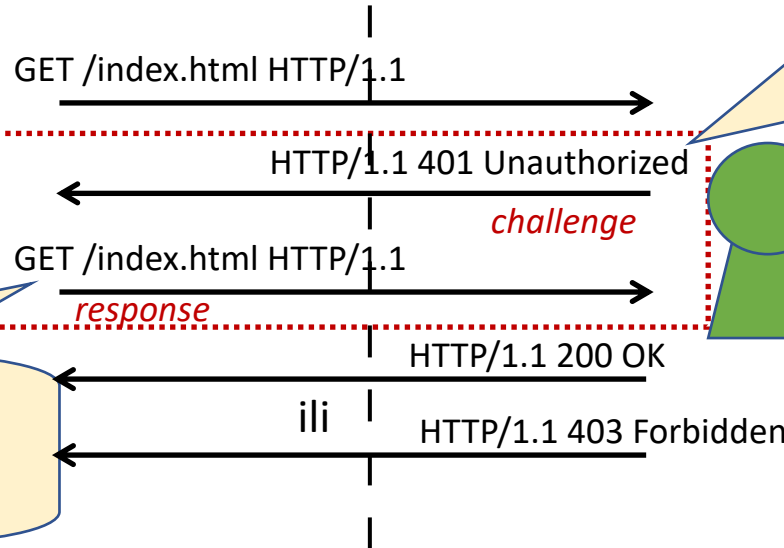
Autentikacija protokolom HTTP (I)

RFC 7235

User agent

Web server

GET /index.html HTTP/1.1
Authorization: <type> <credentials>
...



HTTP/1.1 401 Unauthorized
WWW-Authenticate: <type> [parovi
ime=vrijednost odvojeni zarezom] |
[base64 kodiran niz znakova]
...

- Opći radni okvir za upravljanje pristupom resursima i autentikaciju klijenta
 - *challenge-response* transakcije korištenjem odabrane autentikacijske sheme
- Zahtjev zaštićenim resursom
 - Poslužitelj odbija zahtjev i u zaglavlju odgovora **WWW-Authenticate:** pruža upute o
 - <type> - traženoj shemi autorizacije, npr. Basic, Digest, Bearer ...
 - parametri sheme priloženi kao niz parova ime=vrijednost odvojeni zarezom ili niz base64 kodiranih znakova
 - Klijent šalje zahtjev s autorizacijskim podacima u zaglavlju zahtjeva **Authorization:**
 - <type> - korištena shema autorizacije
 - <credentials> - parametri odgovora na izazov (oblik ovisan o korištenoj shemi autentikacije – parovi vrijednosti ...)

Autentikacija protokolom HTTP (II)

- *Challenge-response* transakcija:
 - odvija se samo kod prvog, neautoriziranog pristupa resursu unutar skupa štićenih resursa
 - kod sljedećih zahtjeva za resurse prema istom skupu štićenih resursa klijent (*user agent*) automatski prilaže autorizacijske podatke unutar zaglavlja – nema *challenge-response* transakcije (jer nema *401* odgovora)
 - Skup štićenih resursa definiran s kanoničkim URI poslužitelja i parametrom `realm`, moguće i s dodatnim parametrima ovisnim o korištenoj shemi
 - kanonički URI: URI komponente shema i autoritet (npr. **https://www.example.com/home/index.html**)
 - parametar `realm` – niz znakova koji logički (korisniku) opisuju skup štićenih resursa
- Poslužitelj može vratiti više `WWW-Authenticate`: zaglavlja s različitim shemama autorizacije
 - Klijent bi trebao odabrati najsigurniju shemu iz skupa od poslužitelja ponuđenih shema, a čiju implementaciju podržava

Sheme autentikacije

- Popis postojećih shema autentikacije:
 - <https://www.iana.org/assignments/http-authschemes/http-authschemes.xhtml>
- Postojeće metode autentikacije (2021.g.):
 - Basic
 - Digest
 - Bearer
 - OAuth (verzija 1 !!!)
 - HOBA
 - Mutual
 - ~~Negotiate~~
 - SCRAM-SHA-1
 - SCRAM-SHA-256
 - vapid

Shema autentikacije Basic (I)

- Klijent u polju zaglavlja HTTP zahtjeva prilaže dijeljenu tajnu:
 - tajna je niz znakova koji se sastoji od *korisničkog imena* i *zaporke* odvojenih dvotočkom, te kodiranih algoritmom base64URL
 - znakovi tajne se uobičajeno kodiraju lokalno korištenim kodiranjem znakova (nedefinirano normom), ako nije drugačije zatraženo parametrom *charset* u polju odgovora WWW-Authenticate (dozvoljena vrijednost polja samo "UTF-8")

Authorization: Basic <credentials>

- za kombinaciju korisničkog imena i zaporke *Otvoreno:Računarstvo*

GET /index.html HTTP/1.1

Host: or.fer.unizg.hr

Authorization: Basic T3R2b3JlbnM86UmHEjXVuYXJzdHZv

...

Provjerite na <https://www.base64decode.org/>

Shema autentikacije Basic (II)

- Interakcija klijenta (preglednika weba) s korisnikom
 - kod prvog zahtjeva za autorizacijom od strane poslužitelja, klijent od korisnika zahtijeva unos korisničkog imena i zaporke, najčešće korištenjem generičkog dijaloga
 - Kod ostalih zahtjeva prema istom poslužitelju, klijent može koristiti lokalno pohranjeno korisničko ime i zaporku te automatski postavljati Authorization polje zaglavlja zahtjeva (izbjegava *challenge-response* transakcije)
- Prednosti sheme:
 - Jednostavnost
 - Svi potrebni autorizacijski podaci se šalju u sklopu svakog zahtjeva – očuvana neovisnost HTTP transakcija – **RESTful** filozofija
- Nedostaci sheme:
 - Dijeljena tajna se prenosi komunikacijskim kanalom između klijenta i poslužitelja
 - Dijeljena tajna je kodirana algoritmom base64 – reverzibilno kodiranje – ne štiti se povjerljivost dijeljene tajne tijekom prijenosa komunikacijskim kanalom – potrebno koristiti siguran komunikacijski kanal (HTTPS)
 - Nije definiran mehanizam prestanka slanja autorizacijskih podatka prema poslužitelju (kao npr. mehanizam poništavanja valjanosti kolačića)
- Shema definirana u [RFC7617](#)

Base64URL

- Base64:
 - znakovi A-Z a-z 0-9 + /
 - Popuna (*padding*) znakovima =
- Problem korištenja originalnog base64 za ime datoteke ili dijela URL-a
 - Znakovi / + i = imaju rezervirano značenje u strukturi URL-a
- Promjene u base64URL
 - + --> -
 - / --> _
 - Nema karaktera popune (=)
 - Zabranjeni separatori redaka
- [RFC4648](#) - definicija kodiranja base64, base32 i base16
- Korisna stranica za (de)kodiranje: <https://base64.guru/standards/base64url>

Shema autentikacije Digest (I)

- Izbjegavanje slanja dijeljene tajne nesigurnim komunikacijskim kanalom
- Tijekom challenge-response transakcije (odbijanje pristupa resursu) poslužitelj klijentu u sadržaju WWW-Authenticate polja zaglavlja odgovora prosljeđuje:
 - shemu autentikacije: `Digest`
 - opis štićenog prostora: `realm="ime ili opis štićenog prostora"`
 - definiciju liste URI-ja koji pripadaju štićenom prostoru: `domain="URI1 URI2 ... "`
 - razine kvalitete zaštite („auth“, „auth-int“ ...): `qop="auth,auth-int"`
 - jednokratni podatak: `nonce="niz okteta u ASCII hex ili base64 formatu"`
 - proizvoljan podatak: `opaque="niz okteta u ASCII hex ili base64 formatu"`
 - *opcijska* zastavica *stale* (istekao nonce): `stale=true`
 - *opcijska* zastavica korištenja sažetka umjesto *plaintext* korisničkog imena: `userhash="true"`
 - *opcijski* algoritam sažetka (MD5, SHA-256, SHA-512/256, običan ili session):
`algorithm=SHA-256` ili `algorithm=SHA-256-sess`

HTTP/1.1 401 Unauthorized

WWW-Authenticate: `Digest realm="http-auth@example.org",
qop="auth, auth-int", algorithm=SHA-256,
nonce="7ypf/xlj9XXwFDPEoM4URrv/xwf94BcCAzFZH4GiTo0v",
opaque="FQhe/qaU925kfzjCev0ciny7QMkPqMAFRtzCUYo5tdS"`

...

Shema autentikacije Digest (II)

- Klijent dobavlja korisničko ime i zaporku (dijeljena tajna s poslužiteljem) i računa sažetke algoritmom MD5 (rezultirajući sažetak duljine 16 okteta) –
 - Za parametar qop=„auth”, algorithm=H

$HA1 = H(\text{korisničko_ime}:\text{realm}:\text{zaporka})$

$HA2 = H(\text{metoda}:\text{URI_put})$

$\text{response} = H(HA1:\text{nonce}:\text{nc}:\text{cnonce}:\text{qop}:HA2)$

Veže sažetak uz korisničko ime, lozinku i
štićeni skup resursa

Veže sažetak uz HTTP metodu i
resurs (put do njega)

Veže sažetak uz:

- poslužitelj *nonce* – poslužitelj vodi računa o ovoj dozvoli, upravlja životnim vijekom
- klijentov *nonceCount* – brojač korištenja *nonce*-a (prevencija *replay* napada)
- klijentov *cnonce* – varijacija rezultirajućeg sažetka
- klijentov *qop* – veže sažetak na korištenu kvalitetu zaštite (korištene postupke)

Shema autentikacije Digest (III)

- Sljedeći upit klijenta za željeni resurs na poslužitelju:

```
GET /dir/index.html HTTP/1.1
```

```
Host: www.example.org
```

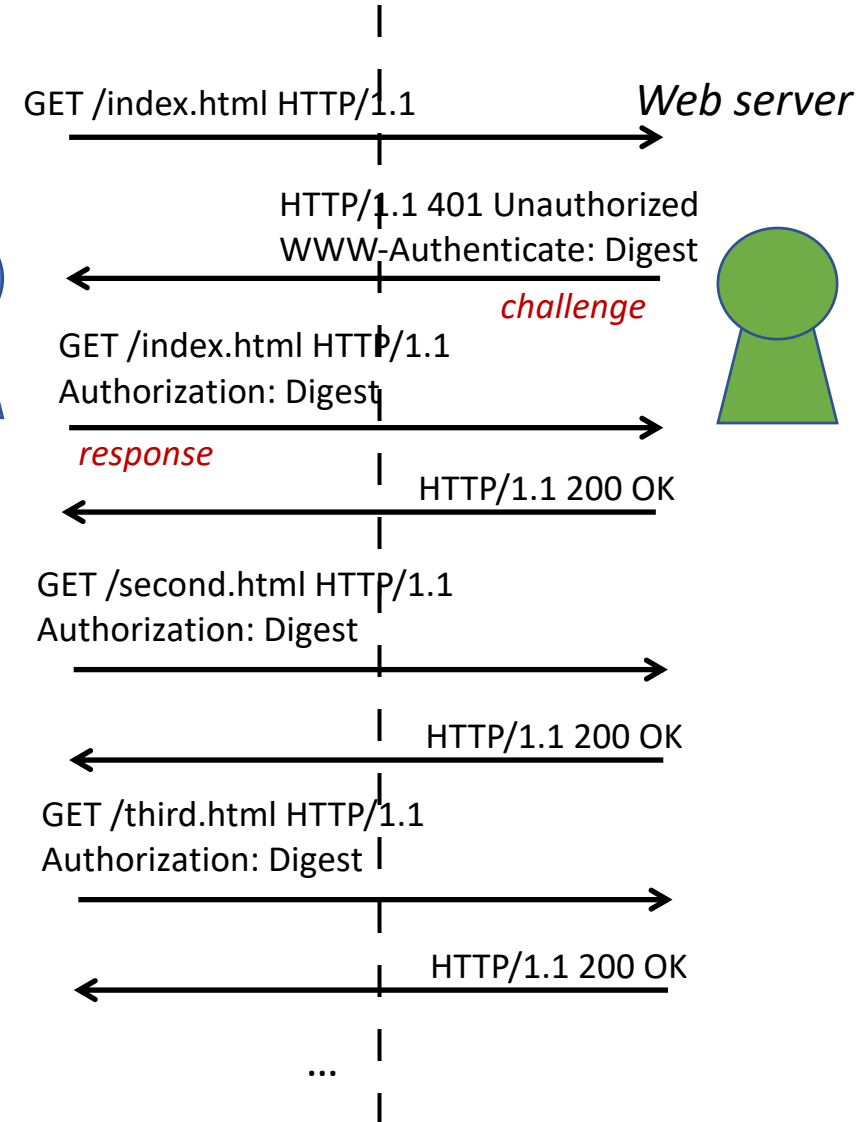
```
Authorization: Digest username="Mufasa", realm="http-auth@example.org",  
    uri="/dir/index.html", algorithm=MD5,  
    nonce="7ypf/xlj9XXwfdPEoM4URrv/xwf94BcCAzFZH4GiTo0v", nc=00000001,  
    cnonce="f2/wE4q74E6zIJEtWaHKaf5wv/H5QzzpXusqGemxURZJ", qop=auth,  
    response="8ca523f5e9506fed4657c9700eebdbec",  
    opaque="FQhe/qaU925kfzjCev0ciny7QMkPqMAFRtzCUYo5tdS"
```

- **username** – korisničko ime ili njegov sažetak (ovisno o vrijednosti zastavice *userhash*)
- **cnonce** – jednokratni ASCII niz znakova stvoren od klijenta
- **nc** – *nounce count* – broj zahtjeva poslanih od strane klijenta s istom vrijednošću *nounce* (definirane od poslužitelja)
- **response** – glavni dio odgovora transakcije challenge (server) nonce – (client) response

Shema autentikacije Digest (IV)

- Interakcija klijenta (preglednika weba) s korisnikom
 - poslužitelj prosljeđuje parametre (realm, qop, nonce, opaque) samo tijekom *challenge-response* transakcije (odgovora 401), klijent koristi iste vrijednosti parametara prosljeđenih od poslužitelja u svim sljedećim transakcijama*
- Prednosti sheme:
 - Nema slanja dijeljene tajne komunikacijskim kanalom
 - Svi potrebni autorizacijski podaci se šalju u sklopu svakog zahtjeva – očuvana neovisnost HTTP transakcija – **RESTful ??** filozofija (a što s opaque podatkom?)
- Nedostaci sheme:
 - Ranjivost na razne vrste sigurnosnih napada (chosen plaintext, brute force, man-in-the-middle, ...)
 - Nema provjere autentičnosti klijenta i poslužitelja
 - Nema mogućnosti autentikacije strana u komunikaciji
 - Preporučeno korištenje preko sigurnog kom. kanala
- Shema definirana u [RFC7616](#)

User agent

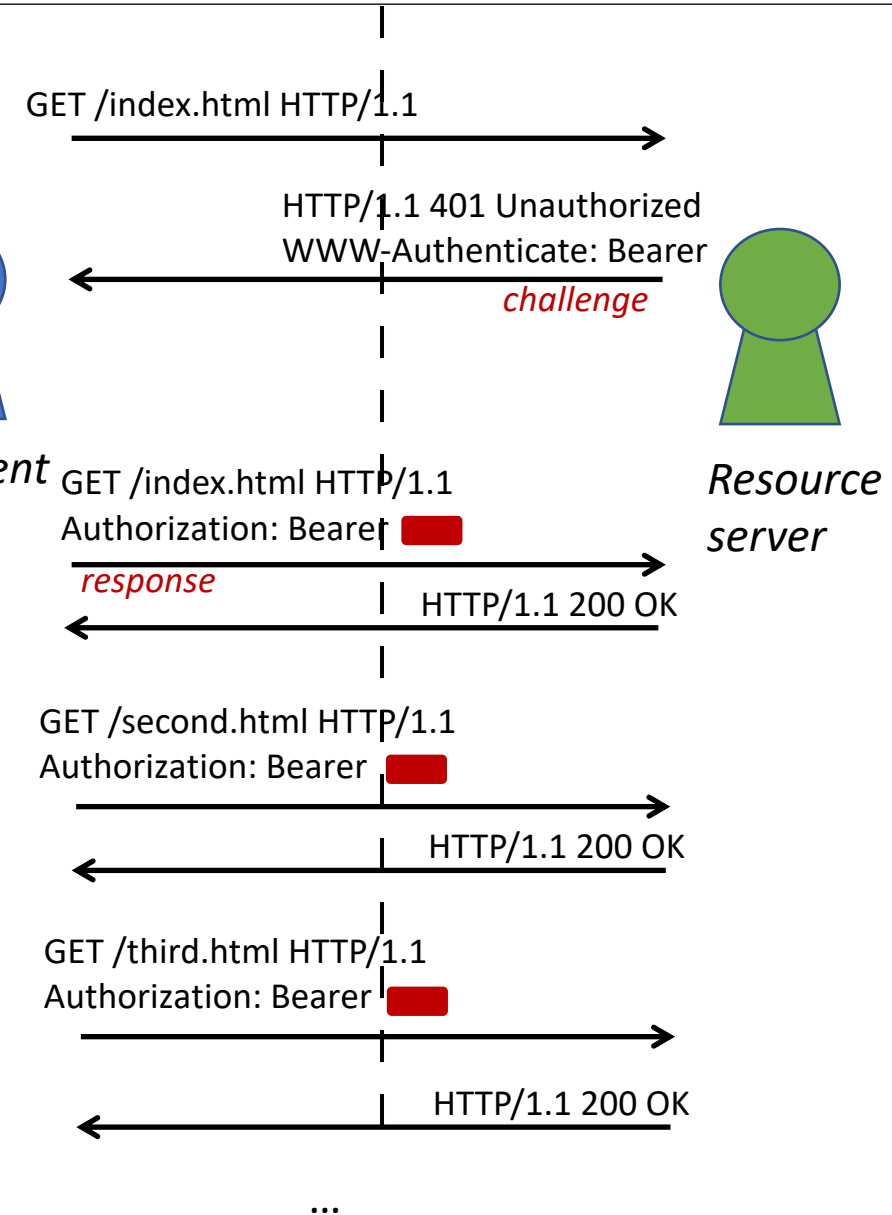
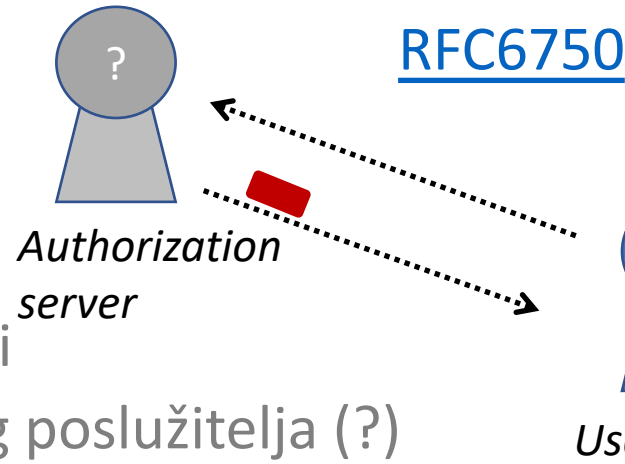


* - za isti skup štićenih resursa

Shema autentikacije Bearer (I)

■ Bearer token

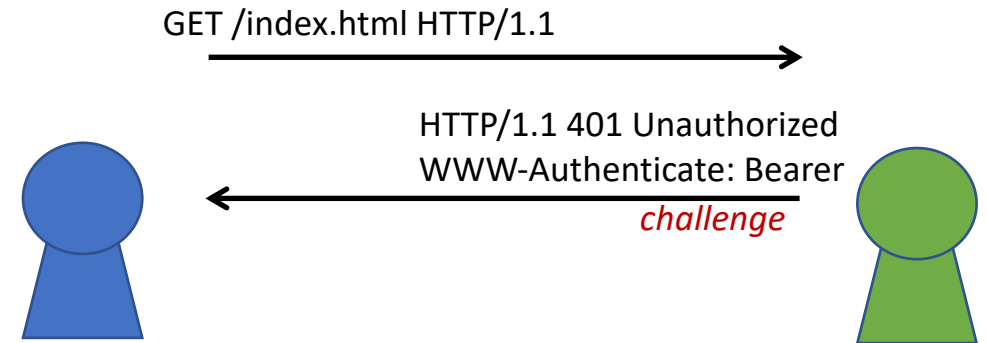
- Niz znakova koji služe kao dokaz jedne ili skupa tvrdnji
- Izdan od strane sigurnosnog poslužitelja (?)
- Nositelj (entitet u posjedu bearer tokena) prezentira token *poslužitelju resursa* kako bi dobio pristup štićenom resursu
- Mora biti štićen pri pohrani i tijekom prijenosa komunikacijskim kanalom (TLS - HTTPS)



Shema autentikacije Bearer (II)

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Bearer realm="example",
                  scope="scope1 scope2 scopeN"
```

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Bearer realm="example",
                  error="invalid_token",
                  error_description="The access token expired"
```

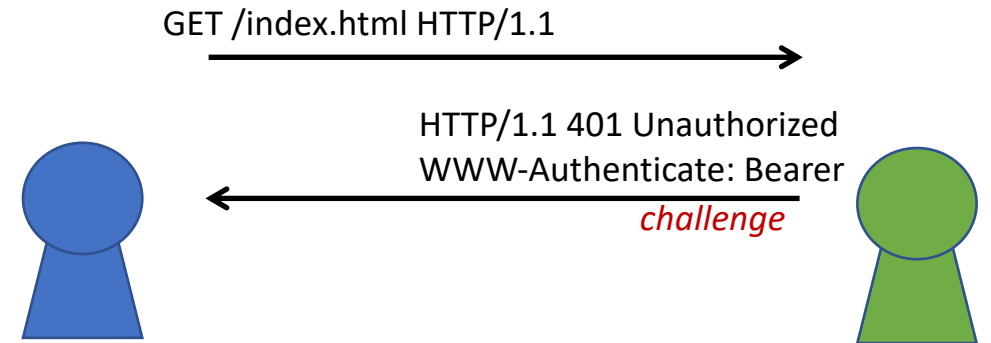


- Poslužitelj resursa vraća *challenge* (401) odgovor ako:
 - klijent ne priloži zahtjevu nikakve vjerodajnice (token)
 - klijent priloži zahtjevu nevaljale vjerodajnice
- Parametar opseg (*scope*):
 - opcionalan (kao i parametar `realm`)
 - jedna ili više vrijednosti (riječi) odvojenih razmacima koje definiraju opseg prava nad resursima
 - moguće vrijednosti i značenja implementacijski ovisna - definirana ovisno o pojedinom autorizacijskom poslužitelju

Shema autentikacije Bearer (III)

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Bearer realm="example",
                  scope="scope1 scope2 scopeN"
```

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Bearer realm="example",
                  error="invalid_token",
                  error_description="The access token expired"
```



■ Parametri opisa greške:

- navode se ukoliko je zahtjev klijenta sadržavao neispravne vjerodajnice
- *error: invalid_request (400), invalid_token (401), insufficient_scope (403)*
- *error_description*: opis greške (za developere)
- *error_uri*: poveznica na stranicu s detaljnijim opisom greške

Shema autentikacije Bearer (IV)

```
GET /resource HTTP/1.1  
Host: server.example.com  
Authorization: Bearer mF_9.B5f-4.1JqM
```

Base64 znakovi



- Token zapisan u base64 obliku
- Sadržaj tokena nije čitljiv nositelju tokena
 - sadržaj je bitan entitetu izdavatelju tokena i entitetu poslužitelju štićenog resursa
 - u pojedinim primjenama sadržaj tokena mora biti nedostupan nositelju (šifriran)
 - token se ne smije prenositi nesigurnim komunikacijskim kanalima
 - svatko tko posjeduje odgovarajući token može pristupiti povezanom resursu ili skupu resursa
 - Ograničavanje roka trajanja tokena (uključivanje informacije o roku valjanosti unutar samog tokena)
 - Ograničavanje skupa resursa kojima se može pristupiti i odgovarajućih akcija za pojedinog klijenta nositelja tokena (parametar `scope`)
 - ...

Shema autentikacije Bearer (V)

- Prednosti:

- odvajanje autentikacije i autorizacije od posluživanja resursa
- delegiranje prava akcija nad resursima trećoj strani, u ime vlasnika resursa

- Nedostaci:

- token je sigurnosno kritična informacija, mora se štititi od neovlaštenog pristupa
- Vrste napada na tokene:
 - Krivotvorenje ili neovlaštena izmjena sadržaja tokena
 - Razotkrivanje sadržaja tokena (može sadržavati sigurnosno osjetljive informacije)
 - Preusmjerenje tokena - token namijenjen za jedan sustav preusmjerava se drugom sustavu i ostvaruju nepripadajuća prava nadštićenim resursima
 - *Token replay* – neovlaštena ponovna uporaba tokena

- Preporučeni postupci:

- korištenje isključivo sigurnih komunikacijskih kanala
- kratki period valjanosti tokena
- ograničavanje opsega prava tokena
- osiguravanje integriteta (dijela) sadržaja tokena
- osiguravanje povjerljivosti (dijela) sadržaja tokena

API ključevi (I)

- API ključ:

- Služi za identifikaciju korisnika API-ja (korisnik \approx projekt)
- Ne služi za identifikaciju korisnika u užem smislu (tj. autentikaciju korisnika)
- Može služiti za autorizaciju korištenja (dijelova) API-ja
 - Definira prava pristupa krajnjim točkama i moguće akcije

- Nedostaci:

- Svatko tko posjeduje valjani ključ ima pristup API-ju
- Za očuvanje povjerljivosti mora se koristiti sigurni komunikacijski kanal
- Ne sprječava maliciozno korištenje API-ja

API ključevi (II)

- Načini prosljeđivanja API ključeva:

Upit segment URL-a

```
GET /something?api_key=abcdef12345 HTTP/1.1
```

```
GET /something HTTP/1.1  
Host: www.target.host.org  
X-API-Key: abcdef12345
```

Custom polje zaglavlja zahtjeva

Kolačić

```
GET /something HTTP/1.1  
Cookie: X-API-KEY=abcdef12345
```

Otvoreno računarstvo

9. Sigurnost u otvorenim sustavima

- Autentikacija i autorizacija
- HTTP sheme autentikacije
- Tokeni (JWT)
- Sigurni komunikacijski kanal (TLS/SSL, HTTPS)
- Delegiranje prava (OAuth2)
- Autentikacija (OpenID, OpenIDConnect)

Json Web Tokens (JWT)

- (sigurnosni) tokeni – oblik i sadržaj aplikacijski specifični
- JWT (jot) – pokušaj standardizacije (uniformizacije) tokena – [RFC 7519](#)
 - formata zapisa (JSON)
 - strukture (zaglavlje, tijelo, potpis)
 - sigurnosnih mehanizama
 - sadržaja - parovi (ime, vrijednost), imenima pridružena dijeljena semantika
- Kompaktan zapis tokena, pogodan za prijenos
 - unutar zaglavlja HTTP zahtjeva ili odgovora (*Bearer shema*)
 - unutar URL-a ili tijela HTTP zahtjeva ili odgovora
- Informacije unutar JWT su
 - jamčenog integriteta i izvora (potpisani tokeni)
 - *jamčene povjerljivosti (šifrirani tokeni)*
 - *samo-dostatne – samo na osnovu sadržaja tokena, bez kontaktiranja drugih izvora informacija – usluga ili lokalne pohrane podataka) moguće je dobiti potpunu informaciju (prenose potpuno stanje – REST princip!)*

Struktura JWT

- Token se sastoji od tri elementa:
 - Zaglavlja – definira tip tokena i algoritam korišten za potpisivanje tokena
 - Tijelo – sadrži skup tvrdnji (parova ime-vrijednost)
 - Potpisa – sadrži potpis tokena štiteći njegov integritet

<https://jwt.io/>

- **Zapis tokena**
 - Sastoji se od tri polja - niza znakova dobivenih kodiranjem elemenata tokena, odvojenih znakom točke (.)
 - Svako od polja je kodirano metodom base64URL

```
token = base64url(zaglavlje) + '.' + base64url(sadržaj) + '.' + base64url(potpis)
```

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjMONTY3ODkwIiwibmFtZSI6Ikcwcm1sc2tpIGtvcGHEjWnEhYlsmIhdCI6MTYxNzIzOTAyMiwiwGFqbmEiOiJzYXJtYSBvZCB0dW5lIiwicG9ydWthIjoidGtlIHBydmkgamF2aSB0YWpudSBpIGlhdCBuYXN0YXZuaWNpbWEsIGRvYmIqZSBqYXZudSBwb2h2YWx1In0.7naNmMpE1m5LNUucQJBF-OVd9hnZiQI_1f2jQqWZwE

Zaglavlje JWT

```
{  
  "alg": "HS256",  
  "typ": "JWT"  
}
```

← rfc7518

tip tokena - JWT

- Zaglavlje sadrži informacije o:
 - Algoritmu korišteno za potpisivanje (i enkripciju) zaglavlja i tijela tokena
 - HS256 ([RFC2104](#)) koristi dijeljenu tajnu i SHA-256 za stvaranje sažetka poruke
 - RS256 ([RFC3447](#)) koristi RSA (tajni i javni ključ) i SHA-256 za potpisivanje i provjeru potpisa sažetka poruke
 - "none" – za neosigurane tokene (u tom slučaju ne postoji ni segment potpisa tokena)
 - ...
 - Tipu tokena: niz znakova "JWT"

Polja zaglavlja

- `typ` – tip tokena (JWT)
 - `alg` – izdavalatelj tokena deklarira algoritam za provjeru potpisa tokena
 - `kid` – ID ključa s kojim je stvoren potpis
 - `x5c` – lanac certifikata u formatu RFC4945 za specificiranje privatnog ključa kojim je stvoren potpis tokena, verifikacija se obavlja javnim ključem
 - ...
-
- Povezane norme: JWS, JWK, JWA !

Sadržaj JWT

```
{  
  "jti": "WRtaW4iLCJpYXQiOiE0MjI3Nzk",  
  "iat": 1422779638,  
  "iss": "or:authServer:fer:hr",  
  "nickname": "bongo",  
  "course": "opencomputing"  
}
```

- Sadrži tvrdnje – parove ime: vrijednost
 - **Registrirane tvrdnje** – sedam registriranih imena, jedinstvena semantika
 - Javne tvrdnje – definirane od strane nekog korisnika JWT, registrirane kod [IANA](#)
 - **Privatne tvrdnje** – sadrže podatke specifične za konkretnu primjenu tokena

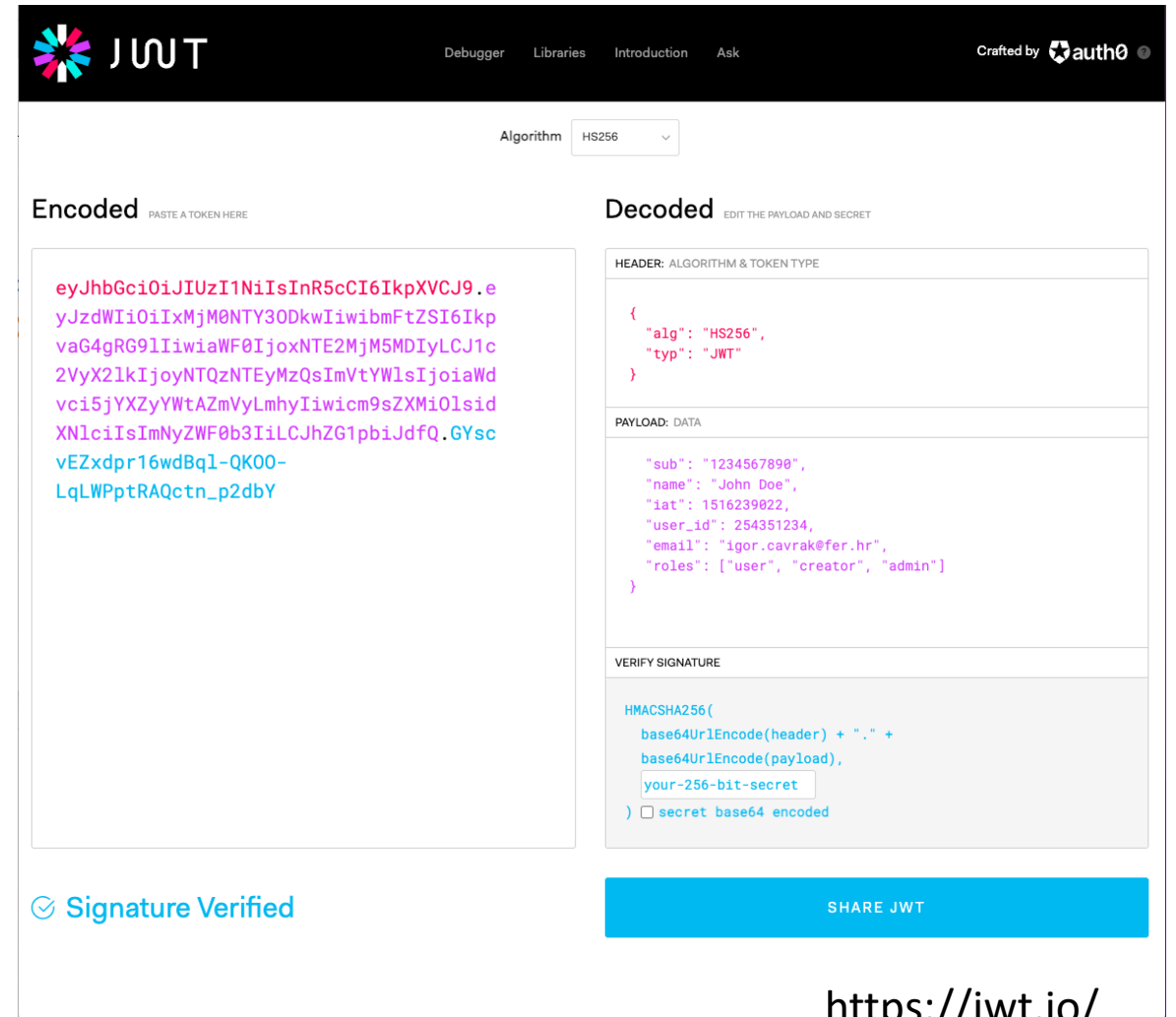
Registrirane tvrdnje

- `jti` – JWT ID – globalno jedinstveni identifikator tokena
- `iss` – izdavatelj tokena (npr. ime autorizacijskog poslužitelja)
- `iat` – vrijeme izdavanja tokena format: sekunde od 01.01.1970. 00:00:00)
- `sub` – predmet tokena (korisnik, resurs ...)
- `aud` – publika – kome je namijenjen token
- `exp` – vrijeme isteka valjanosti (format – isti kao kod polja `iat`)
- `nbf` – nije valjan prije (format – isti kao kod polja `iat`)

Potpis JWT

- Postupak stvaranja polja potpisa tokena:

```
potpis = base64url(ALG(tajna,  
base64url(zaglavljje) + '.' +  
base64url(sadržaj)))
```



The screenshot shows the JWT.io website interface. At the top, there's a navigation bar with links: Debugger, Libraries, Introduction, Ask, and a 'Crafted by auth0' logo. Below the navigation bar, there's a dropdown menu for 'Algorithm' set to 'HS256'. The main content area is split into two columns: 'Encoded' and 'Decoded'. The 'Encoded' column has a text area containing a long string of characters. The 'Decoded' column shows the decoded token structure, including the header, payload, and signature verification details. The header is 'ALGORITHM & TOKEN TYPE' and the payload is 'DATA'. The signature verification section shows the HMACSHA256 algorithm and a checkbox for 'secret base64 encoded'.

JWT

Debugger Libraries Introduction Ask

Crafted by auth0

Algorithm HS256

Encoded PASTE A TOKEN HERE

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{  
  "alg": "HS256",  
  "typ": "JWT"  
}
```

PAYLOAD: DATA

```
{  
  "sub": "1234567890",  
  "name": "John Doe",  
  "iat": 1516239022,  
  "user_id": 254351234,  
  "email": "igor.cavrak@fer.hr",  
  "roles": ["user", "creator", "admin"]  
}
```

VERIFY SIGNATURE

HMACSHA256(
 base64UrlEncode(header) + "." +
 base64UrlEncode(payload),
 your-256-bit-secret
) ☐ secret base64 encoded

Signature Verified

SHARE JWT

<https://jwt.io/>

Korištenje JWT (I)

▪ Prijava na poslužitelj

▪ Zamjena za kolačić

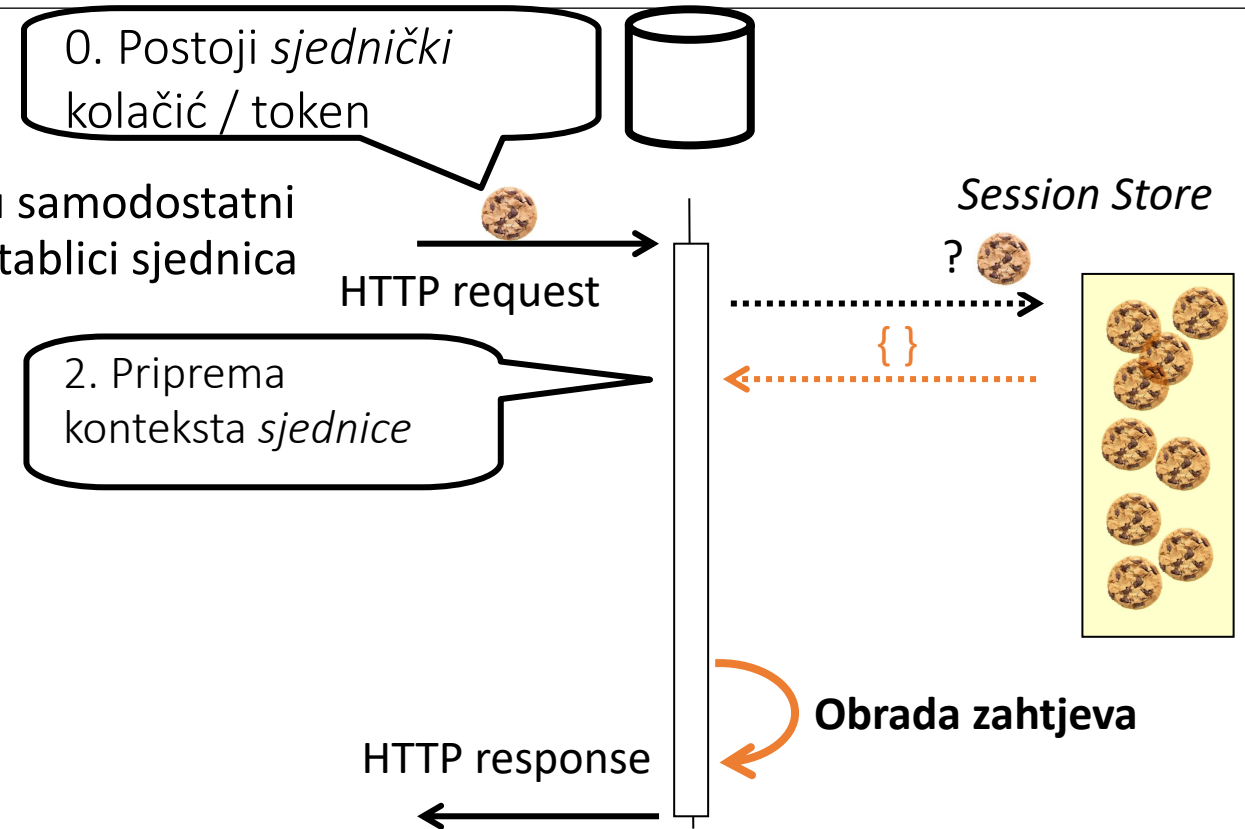
- kolačić sadrži identifikator sjednice, podaci nisu samodostatni
- referenca na zapis o stanju sjednice nalazi se u tablici sjednica
- problem skalabilnosti
- lakše upravljanje životnim vijekom sjednice

▪ RESTful prosljeđivanje stanje sjednice

- podaci u JWT su samodostatni
- stanje sjednice u potpunosti sadržano u tokenu
- nema problema skalabilnosti
- teže upravljanje životnim vijekom sjednice

▪ Prosljeđivanje JWT

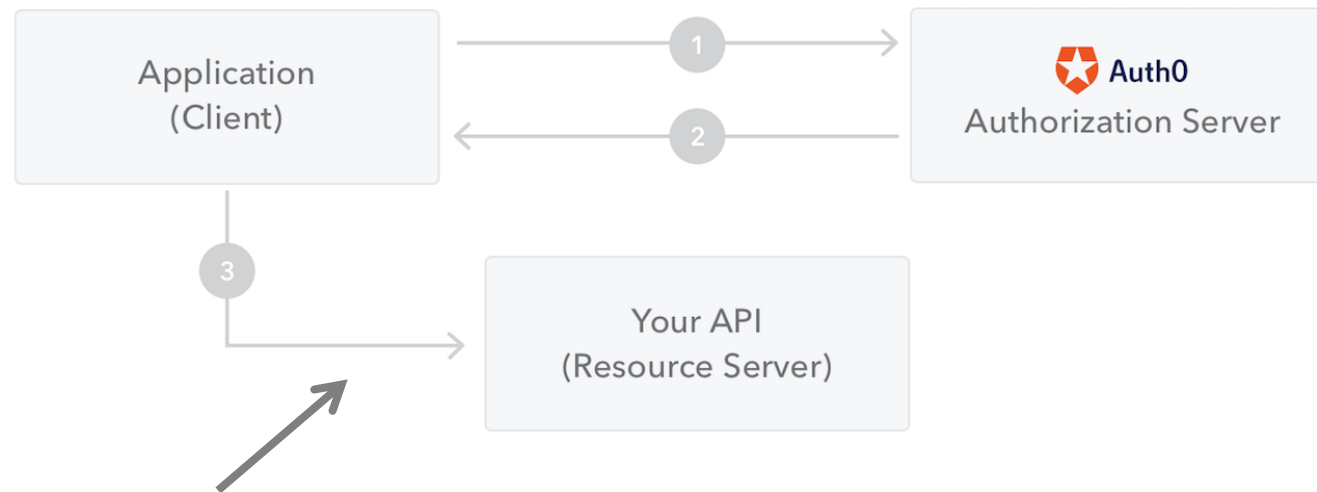
- kolačić - automatsko prosljeđivanje od strane preglednika, zaštićenost pristupa tokenu
- zaglavlja – *WWW-Authenticate* ili *posebna (x-...)* zaglavlja, nema automatizma postavljanja zaglavlja od strane preglednika tijekom formiranja upita
- query string – jedino primjenljivo rješenje kod automatske redirekcije zahtjeva



Korištenje JWT (II)

- Delegirana autentikacija i/ili autorizacija

- Autentikacija i/ili autorizacija obavlja se na zasebnom entitetu sustava
- Informacije o identitetu i dozvolama zapisane unutar zaštićenog tokena
- Svi entiteti unutar sustava vjeruju jedan drugom
- Provjera integriteta tokena korištenjem dijeljenih tajni ili javnih ključeva izdavatelja



1 Zahtjev za autorizaciju

2 Odobrena autorizacija – vraća se JWT token

3 Prosljeđivanje tokena unutar svakog zahtjeva prema štićenom resursu

GET /resource HTTP/1.1
Authorization: Bearer <JWT token>

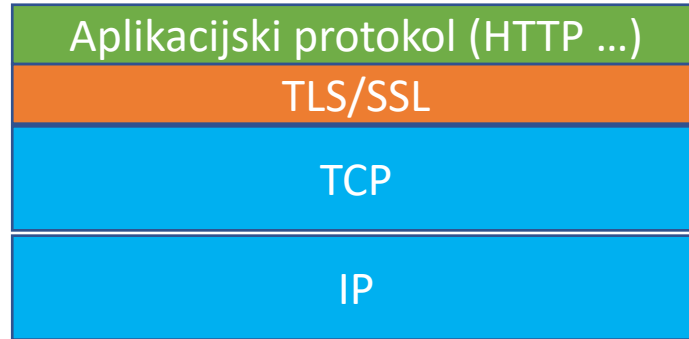
preuzeto s <https://jwt.io/introduction/>

Otvoreno računarstvo

9. Sigurnost u otvorenim sustavima

- Autentikacija i autorizacija
- HTTP sheme autentikacije
- Tokeni (JWT)
- Sigurni komunikacijski kanal (TLS/SSL, HTTPS)
- Delegiranje prava (OAuth2)
- Autentikacija (OpenIDConnect)

TLS/SSL



Aplikacijski sloj

[RFC 8446](#)

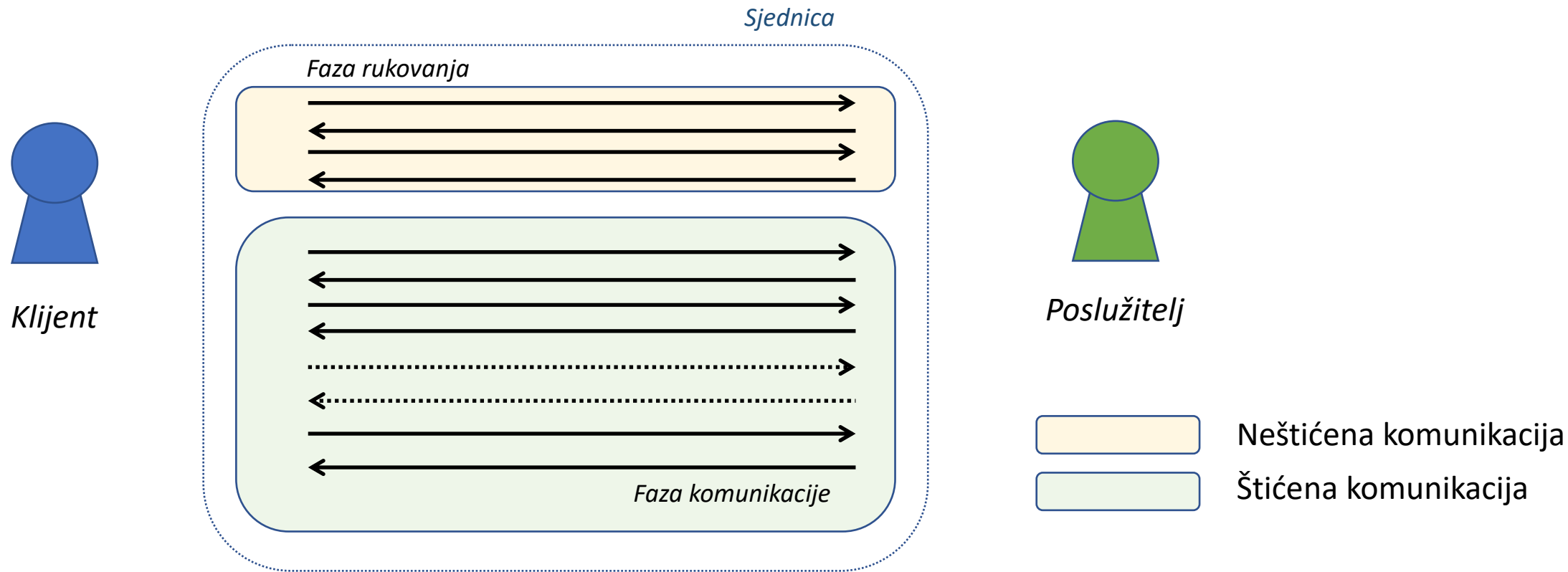
https://www.fer.unizg.hr

- Služi za enkripciju komunikacije na razini aplikacijskih protokola
 - TCP: HTTP, FTP, XMPP, SMTP ...
 - UDP: Datagram Transport Layer Security (DTLS): WebRTC, razne implementacije VPN ...
- Ostvarivanje veze TLS-om
 - Podrazumijevano – TLS-temeljen protokol na odvojenom portu (HTTPS – port 443)
 - Opcija aplikacijskog protokola – naredba za početak uspostave TLS štićenog kanala
- Trenutna verzija TLS 1.3 (kolovoz 2018.)

Svojstva veze ostvarene TLS-om

- Povjerljivost prenošenih podataka
 - Simetrična kriptografija za šifriranje prometa
 - Jednokratni simetrični ključ za svaku sjednicu
- Autentičnost strana u komunikaciji
 - Temeljena na javnom i tajnom ključu te certifikatima
 - Nužna za poslužitelj
 - Opcionalna za klijenta
- Integritet prenošenih podataka
 - MAC za štíćenje integriteta prenošenih podataka
- *Forward secrecy*
 - Moguće - ovisno o odabiru protokola, metoda i algoritama

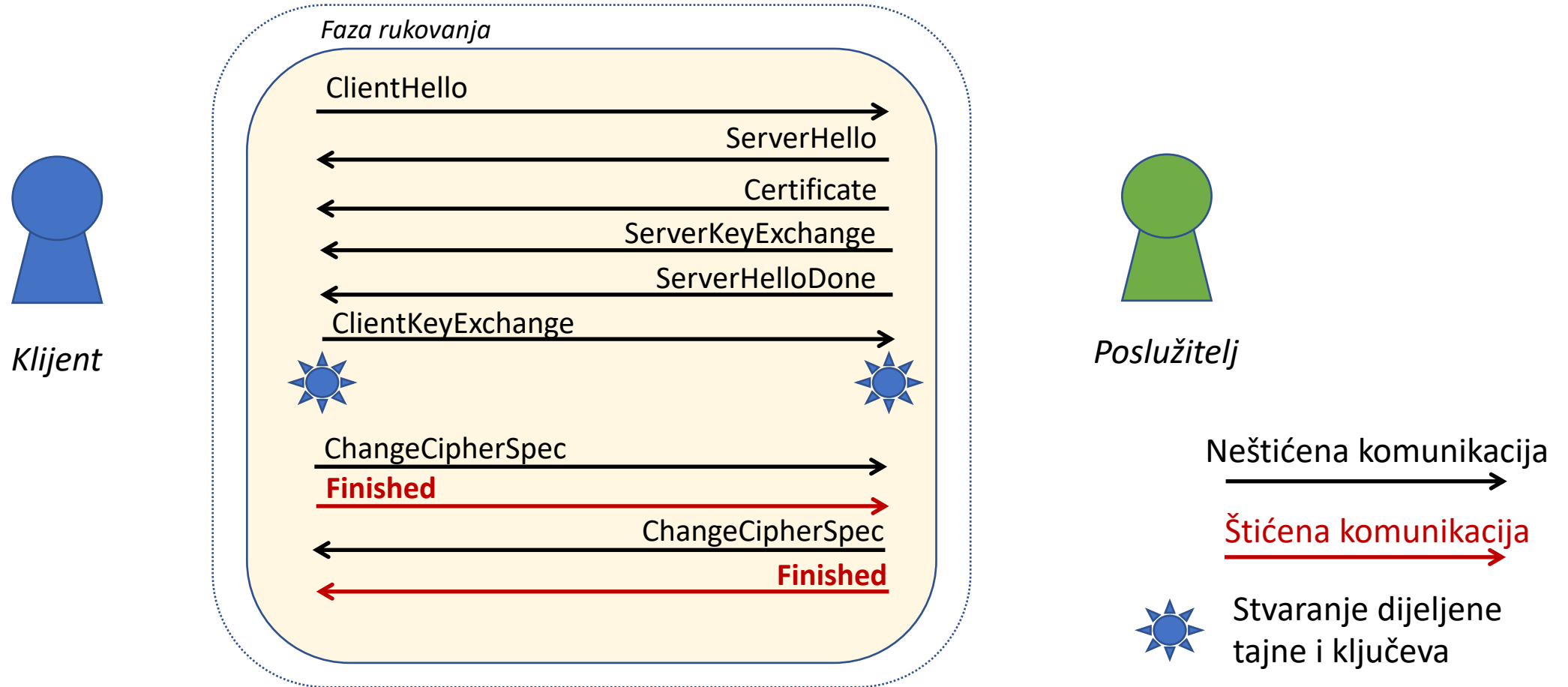
Uspostava sigurnog kanala u TLS-u



▪ Dvije osnovne faze:

- Faza rukovanja – dogovora oko parametara sigurnog kanala
- Faza komunikacije – dvosmjerna komunikacija kroz siguran komunikacijski kanal

Uspostava sigurnog kanala (sjednice) u TLS-u



- Faza rukovanja sastoji se od sljedećih *podfaza*:
 - Dogovora o korištenju verzije protokola i kriptografskih algoritama: ClientHello, ServerHello
 - Razmjene vjerodajnica - javnih ključeva, certifikata: Certificate
 - Dogovoru o sjedničkom ključu: ServerKeyExchange, ClientKeyExchange
 - Prelazak na štićenu komunikaciju: ChangeCipherSpec
 - Provjera identiteta i valjanosti razmijenjivanih poruka tijekom rukovanja: Finished

Dogovor o verziji protokola i algoritmima

- ClientHello sadrži:

- Najviša verzija protokola koju klijent podržava
- *Slučajan broj* – generiran na strani klijenta, za stvaranje sjedničkih ključeva
- Podržani kriptografski algoritmi (skupovi algoritama - cipher suites)
- Podržane metode sažimanja podataka
- *(identifikator sjednice – sessionId – u slučaju postupka nastavljelog rukovanja)*

- ServerHello sadrži:

- Odabrana verzija protokola (najviša zajednički podržana od klijenta i poslužitelja)
- *Slučajan broj* – generiran na strani poslužitelja, za stvaranje sjedničkih ključeva
- Odabrani algoritam i metoda sažimanja podataka
- *identifikator sjednice – sessionId (u slučaju prihvatanja nastavljelog rukovanja vraća sessionId poslan od strane klijenta)*

Razmjena vjerodajnica

- **Certificate sadrži:**

- Certifikat javnog ključa poslužitelja
- Slanje poruke Certificate ovisi o odabranom kriptografskom algoritmu
- Može biti dogovorena i neštićena veza ...

- **Dvostrana autentikacija**

- U verziji rukovanja s razmjenom certifikata klijent također šalje svoje vjerodajnice (certifikat svog javnog ključa):
 - Nakon ServerKeyExchange, poslužitelj šalje poruku CertificateRequest
 - Klijent odgovara s porukom Certificate
 - Klijent prosljeđuje CertificateVerify poruku koja sadrži potpisane sažetke prethodno poslanih poruka prema poslužitelju u fazi rukovanja (slično kao poruka Finished)

Dogovor o sjedničkom ključu

- **ServerKeyExchange** sadrži:
 - Podatke nužne za dogovor o zajedničkoj tajni (Diffie-Hellman key agreement), ako je odabran odgovarajući algoritam
- **ClientKeyExchange** sadrži:
 - Javni ključ klijenta *ili* PreMasterSecret šifriran javnim ključem servera *ili* ništa (ovisno o odabranom algoritmu)
- **Stvaranje zajedničke tajne (*MasterSecret*) temeljeno na:**
 - Razmijenjenim slučajnim brojevima
 - *PreMasterSecret*
- ***MasterSecret* se koristi za daljnje stvaranje (na klijentu i poslužitelju):**
 - MAC ključa
 - Simetričnog ključa za šifriranje aplikacijskih podataka (sjednički ključ)

Štićena komunikacija i provjera rukovanja

- Slanje TLS zapisa ChangeCipherSpec
 - Šalju obje strane u komunikaciji
 - Završava fazu rukovanja i započinje šifriranje i autoriziranje komunikacije s dogovorenim sjedničkim ključem i MAC ključem izvedenim iz *MasterSecret* dijeljene tajne (kako je već dogovoreno tijekom dogovora o algoritmu)
- Slanje poruke Finished:
 - Poruku šalju obje strane u komunikaciji
 - Poruka sadrži MAC-ove prethodno razmijenjenih poruka u fazi rukovanja
 - Sadržaj poruke šifriran sjedničkim ključem
 - Prijemna strana verificira razmijenjene poruke s pristiglim MAC-ovima
 - U slučaju greške, rukovanje se smatra neuspješnim i veza prekida

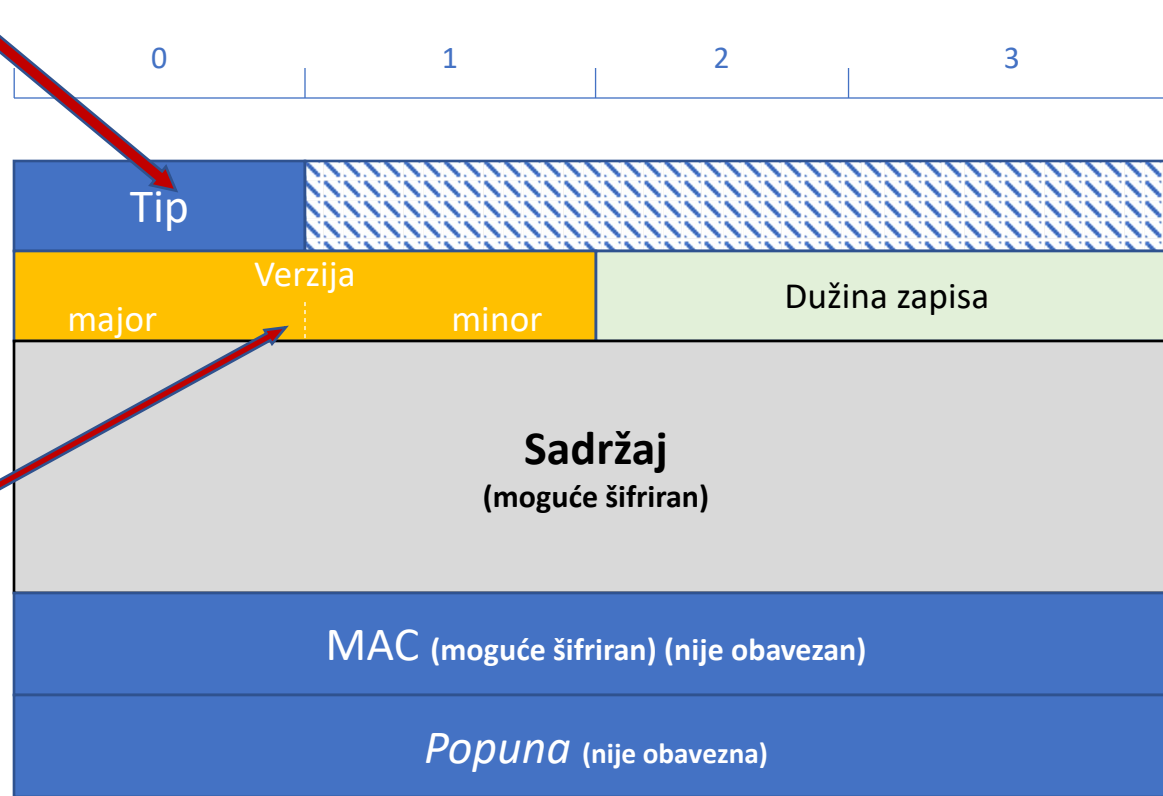
Postupak nastavljenog rukovanja (engl. *resumed handshake*)

- Izvođenje operacija asimetrične kriptografije *resursno skupo i dugotrajno*
- Prethodno postojao komunikacijski kanal između klijenta i poslužitelja
- Ključni podaci o sjednici pohranjeni na poslužitelju
 - Poslužitelj unutar ServerHello poruke šalje klijentu sessionID (identifikator sjednice)
 - Poslužitelj asocira sessionID s lokalno pohranjenim ključnim podacima o sjednici
 - MasterSecret, verzija protokola, odabrani algoritmi šifriranja i sažimanja ...
 - Kod ponovne uspostave veze klijent u sklopu ClientHello poruke prosljeđuje sessionID
 - Nema potrebe za ponovnom razmjenom i stvaranjem ključnih podataka o sjednici
 - Sjednički ključevi su novi – stvaraju se na osnovu MasterSecret (stara) i slučajnih brojeva (novostvoreni!)
 - Rješenje **nije RESTful** – pohranjuju se podaci na poslužitelju!
- *Sjedničke ulaznice (Session tickets)*
 - Poslužitelj pohranjuje ključne podatke o sjednici u ulaznicu
 - Ulaznica je šifrirana i nije čitljiva klijentu, klijent je lokalno pohranjuje
 - Kod ponovne uspostave veze, klijent prosljeđuje ulaznicu poslužitelju
 - Poslužitelj dešifrira i verificira valjanost ulaznice, obnavlja stanje sjednice
 - Rješenje je **RESTful** !

TLS zapis

Tip sadržaja TLS zapisa (engl. *content type*)

- 2 0. ChangeCipherSpec
- 2 1. Alert
- 2 2. Handshake
- 2 3. Application
- 2 4. Heartbeat



Dužina zapisa obuhvaća:

- sadržaj zapisa
- MAC (ako postoji)
- popunu (ako postoji)

Dužina zapisa ograničena na **16KB**.

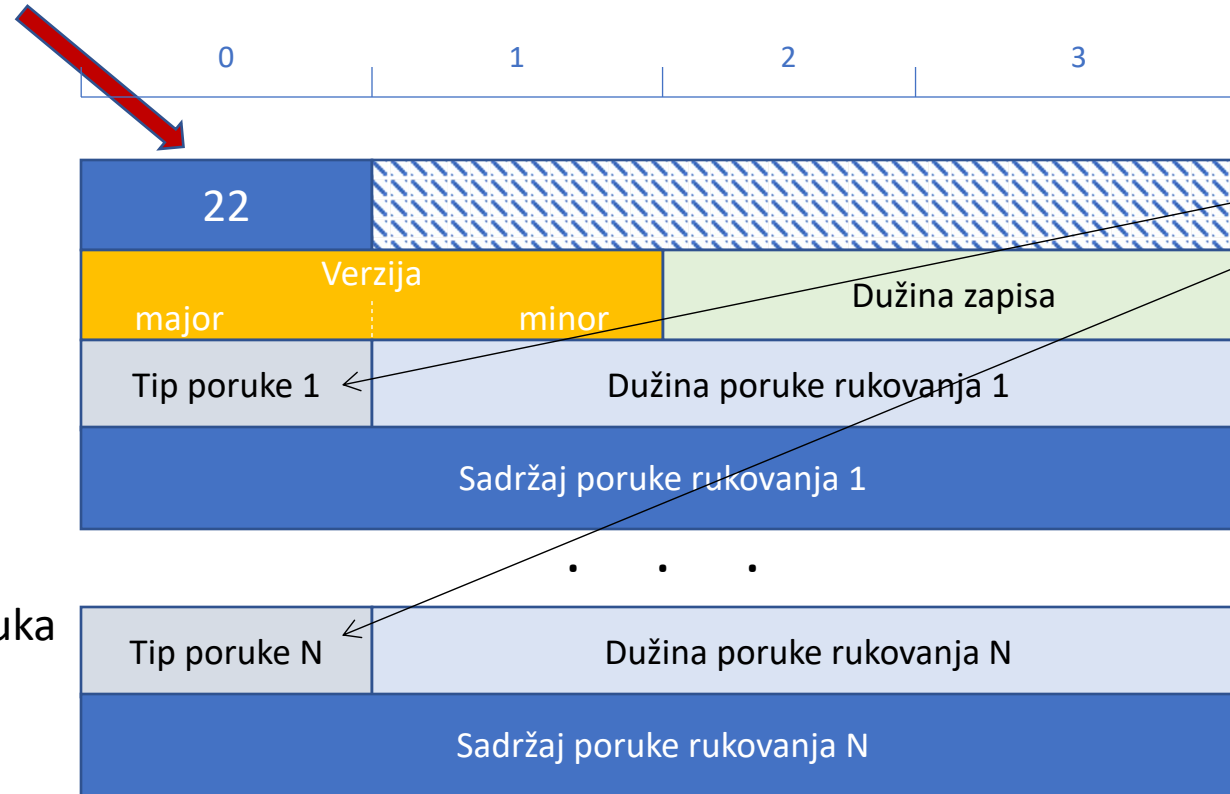
Verzija protokola

major	minor	protokol
3	0	SSL 3.0
3	1	TLS 1.0
3	2	TLS 1.1
3	3	TLS 1.2
3	4	TLS 1.3

MAC i popuna ne postoje u fazi rukovanja (potrebno prijeći u štićenu fazu komunikacije razmjenom zapisa TLS tipa ChangeCipherSpec)

TLS zapis – faza rukovanja

Tip sadržaja TLS zapisa

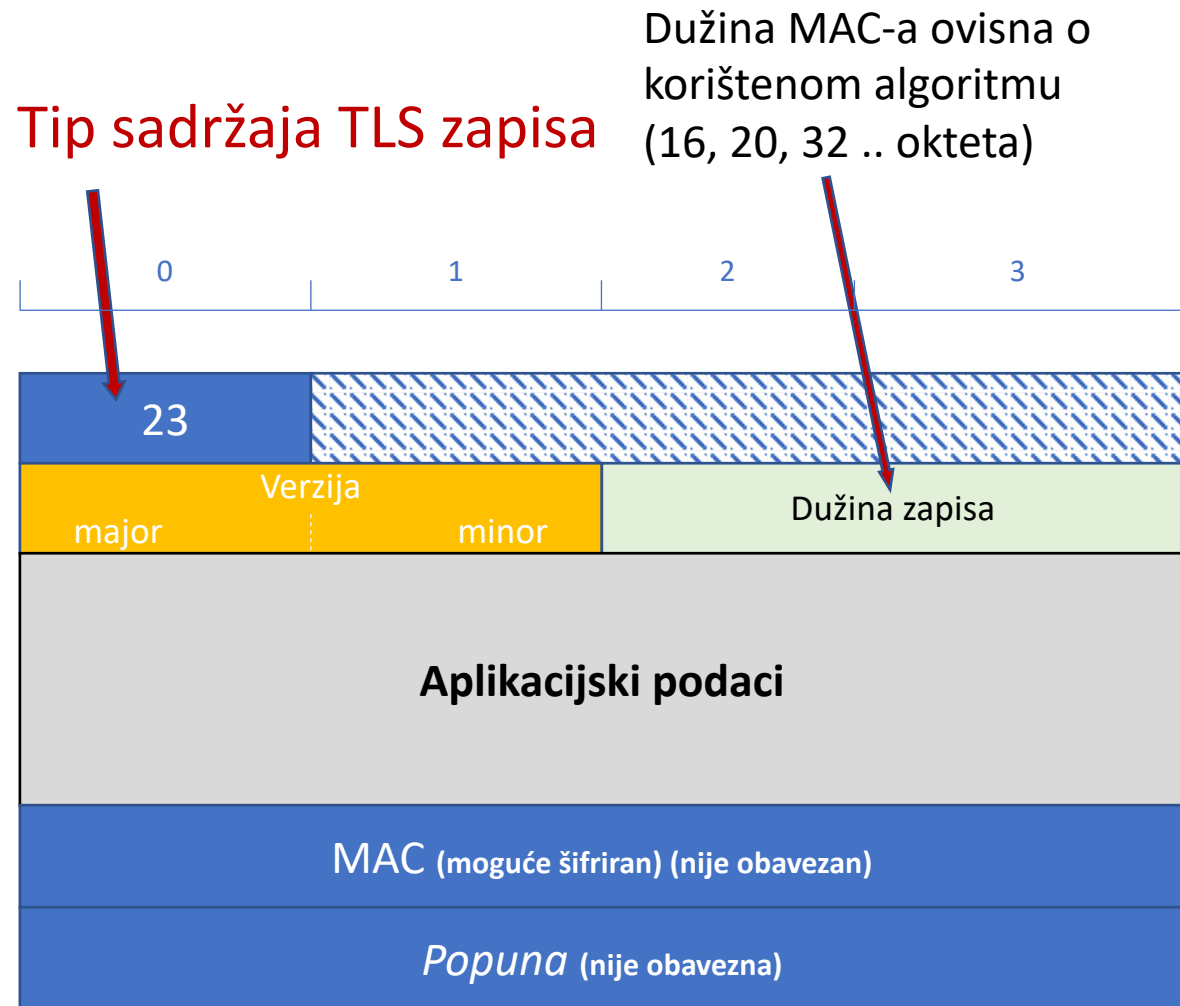


Unutar jednog zapisa tipa sadržaja „rukovanje” (22) moguće je istovremeno prenijeti jednu ili više poruka faze rukovanja

Tipovi poruka rukovanja

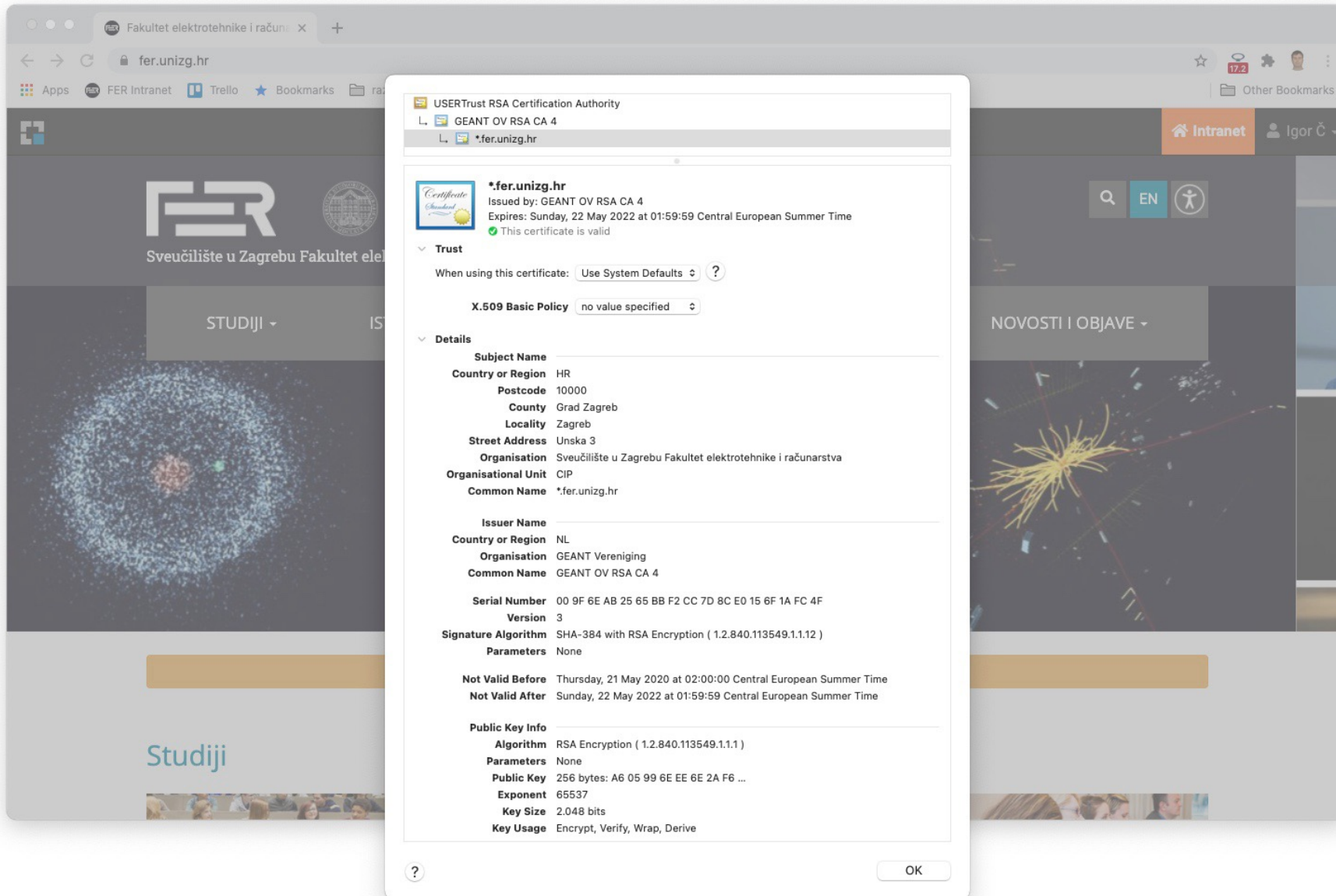
0	HelloReuquest
1	ClientHello
2	ServerHello
11	Certificate
12	ServerKeyExchange
•	•
20	Finished

TLS zapis – prijenos aplikacijskog prometa



- Prenošeni podaci između entiteta u komunikaciji su pakirani u zapise
 - Podaci mogu biti sažeti (ako je tako dogovoreno u fazi rukovanja)
 - Integritet podataka može biti zaštićen korištenjem MAC (ako je tako dogovoreno ...)
 - Tajnost podataka može biti zaštićena kriptiranjem korištenjem sjedničkog ključa (ako je tako dogovoreno ...)

Certifikati poslužitelja (HTTPS)



Nodejs express + HTTPS

openssl req -x509 -nodes -days 365
-newkey rsa:2048 -keyout server.key
-out server.crt



```
//Embarrassingly simple express+https example
//required modules
const express = require("express");
const https   = require("https");
const fs      = require("fs");

//read the keys and certificate (sync read for simplicity)
const privateKey = fs.readFileSync("./server.key", "utf-8");
const certificate = fs.readFileSync("./server.crt", "utf-8");
const credentials = {key: privateKey, cert: certificate};

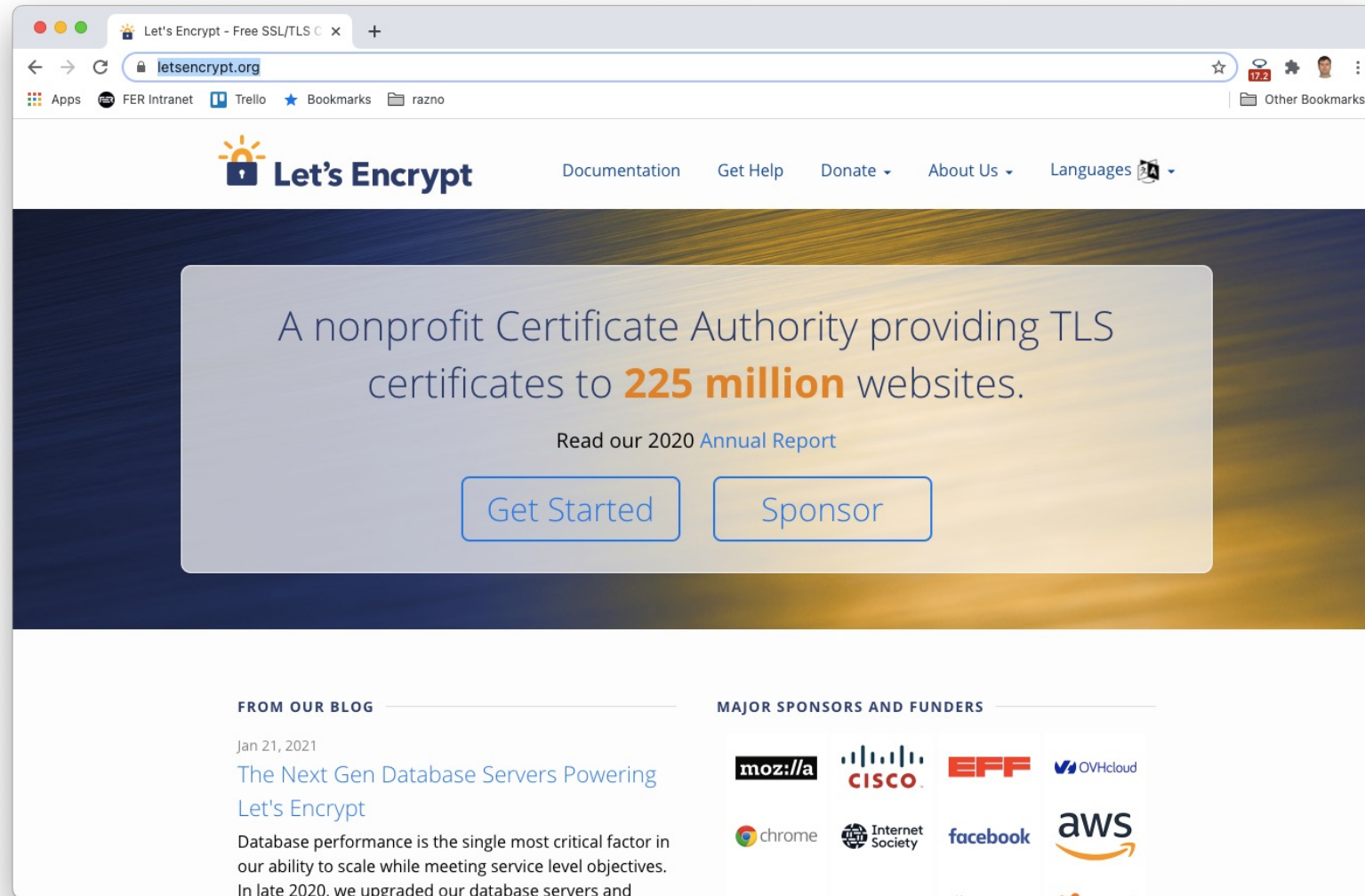
//instantiate the express middleware
const app = express();

app.get("/", (req,res) => {
  res.send("Secure Hello World!");
});

//create https server
const httpsServer = https.createServer(credentials, app);

//start listening for connections
const PORT = 8443;
httpsServer.listen(PORT, () => {
  console.log("HTTPS server started at port " + PORT)}
);
```

Let's Encrypt



<https://letsencrypt.org/>

Popis najvećih pružatelja CA

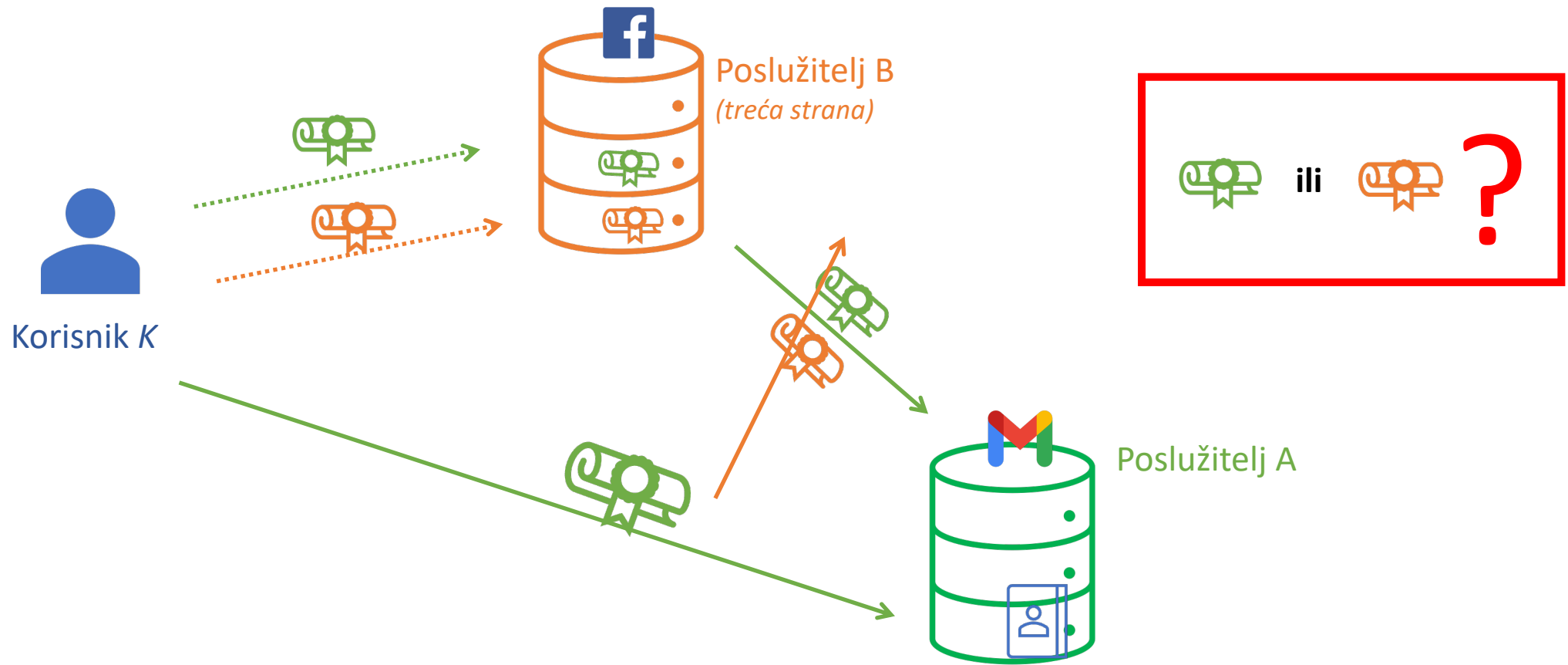
Faza razvoja ili kućni sigurni serveri:
OpenSSL samopotpisani certifikati


Otvoreno računarstvo

9. Sigurnost u otvorenim sustavima

- Autentikacija i autorizacija
- HTTP sheme autentikacije
- Tokeni (JWT)
- Sigurni komunikacijski kanal (TLS/SSL, HTTPS)
- Delegiranje prava (OAuth2)
- Autentikacija (OpenID, OpenIDConnect)

Problem delegacije dozvola pristupa



 Vjerodajnice korisnika K na poslužitelju A za pristup resursima korisnika K (korisničko ime i lozinka)

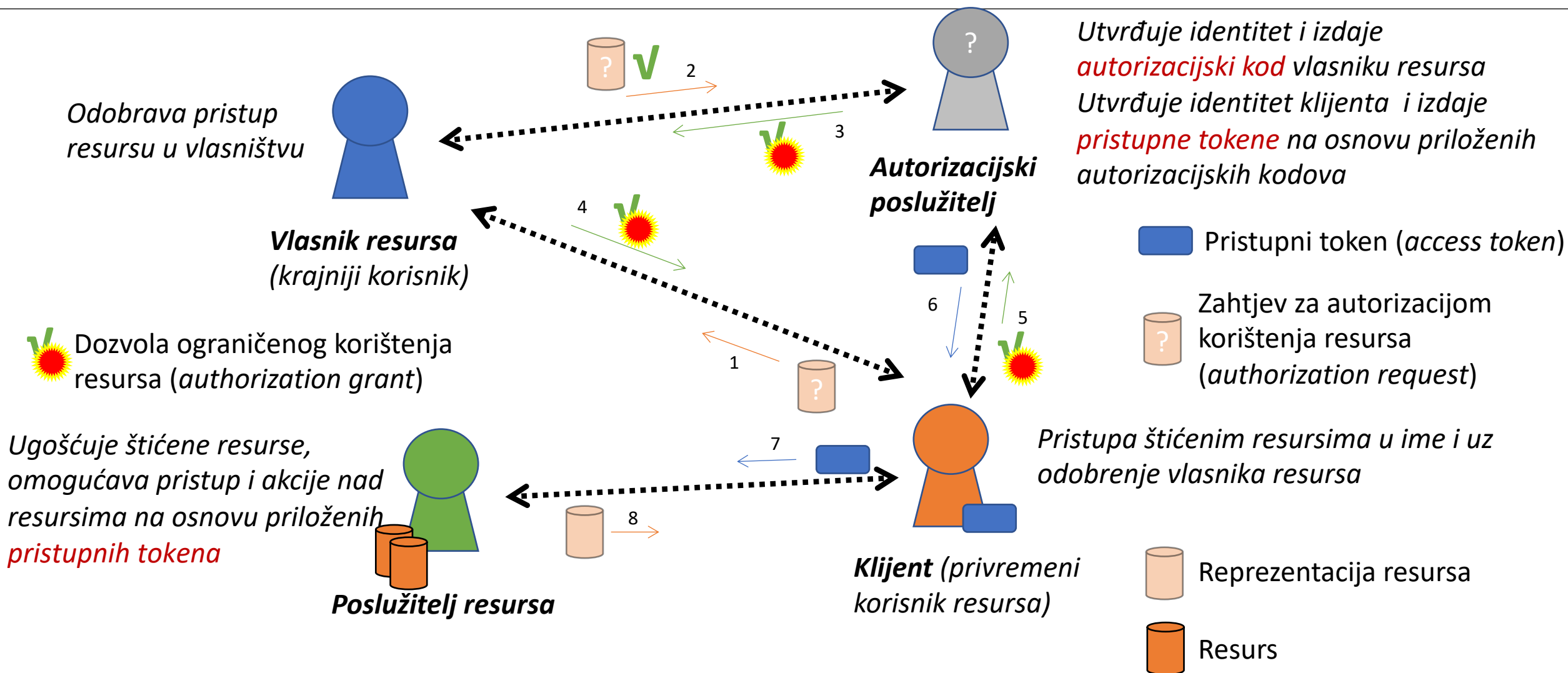
 Vjerodajnice poslužitelju B za pristup resursima korisnika K na poslužitelju A

OAuth 2.0

- Autorizacijski okvir koji omogućava *trećoj strani* ograničen pristup resursima
 - u ime vlasnika resursa, uz ishodeno odobrenje vlasnika resursa
 - u ime treće strane
 - otvorena norma: [RFC 6749](#), [RFC 6750](#), [RFC 8252](#)
- Ključne značajke OAuth 2.0
 - odvajanje uloga *vlasnika resursa* i *klijenta* (korisnika resursa)
 - odvajanje *upravljanja autorizacijom pristupa* resursu i samog *pružanja pristupa* resursu
 - svaki zahtjev prema resursu mora uključivati informaciju o dozvoli u obliku *pristupnog tokena*
- *Treća strana*
 - ne mora biti vlasnik resursa, ne upravlja pristupom resursu
 - web aplikacija, poslužitelj i slični entiteti ... ili
 - klijentska aplikacija - korisnički agent (preglednik weba), *nativni* klijent ...
- OAuth 2.0 je isključivo predviđen za korištenje s protokolom HTTP(S)
 - prenošenje sigurnosno osjetljivih podataka između komponenti sustava



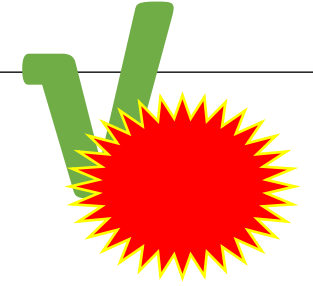
OAuth 2.0 uloge



Autorizacijski poslužitelj može ujedno biti i **poslužitelja resursa**

Autorizacijski poslužitelj može izdavati pristupne tokene valjane kod više **poslužitelja resursa**

Autorizacijski kod



- Izdan od autorizacijskog poslužitelja
 - potvrda da je autorizacija provedena
 - Utvrđen identitet vlasnika resursa
 - Utvrđena vrsta dozvole (resursi i akcije) i klijent kojem je ta dozvola izdana
 - (još) NIJE verificiran identitet klijenta (aut. poslužitelj izravno komunicirao samo s vlasnikom resursa)
 - NIJE dozvola pristupa resursima na poslužitelju resursa
 - Klijent tek treba autorizacijski kod zamijeniti za pristupni token
- Svojstva autorizacijskog koda
 - nerazumljiv ikome osim autorizacijskom poslužitelju
 - kratkog vremena valjanosti (red veličine minuta)
 - jednokratn

- Pristupni token (*access token*) predstavlja vjerodajnice za pristup štićenim resursima na **poslužitelju resursa**
 - niz base64URL kodiranih znakova
- Tipičan sadržaj tokena - atributi (parovi ime=vrijednost) ☐ **JWT !?**
 - atributi koji određuju *skup resursa* na koje se token odnosi
 - atributi koji određuju *vremenski period* valjanosti atributa
 - *atributi koji definiraju korisnika koji pristupa resursu (opcija)*
 - aplikacijski specifični atributi
- Neproziran (*opaque*) za klijenta
 - integritet tokena jamčen digitalnim potpisom
 - tajnost sadržaja tokena jamčena šifriranjem (opcija)
 - token u pravilu mora biti prenošen sigurnim komunikacijskim kanalom

- Vrijeme valjanosti pristupnog tokena je **kratko**
 - gubitak pristupnog tokena predstavlja kratkotrajan ali značajan sigurnosni rizik
 - nakon isteka vremena valjanosti, klijent:
 - zahtijeva novi pristupni token od autorizacijskog poslužitelja korištenjem *refresh* tokena ili
 - ponavlja postupak traženja autorizacije pristupa od vlasnika resursa
- Sadržaj tokena interpretiran od strane poslužitelja resursa
 - tumačenje tijekom postupka autorizacije
 - token može biti:
 - Referenca (engl. *reference token*) na autorizacijske podatke pohranjene lokalno (poslužitelj resursa) ili na autorizacijskom poslužitelju – ~~RESTful~~
 - Skup samo-dostatnih podataka (engl. *self-encoded token*) – svi podaci potrebni za autorizaciju nalaze se unutar tokena – *RESTful* rješenje!

- Reference token (za srednje složene sustave API-ja)
 - Po primitku tokena podaci o dozvolama pristupa resursu se dohvaćaju iz spremišta (cache, baza podataka ...)
 - + jednostavno upravljanje životnim vijekom dozvola (revokacija tokena)
 - + potpuna slika svih aktivnih tokena u sustavu
 - - sporiji postupak provjere valjanosti tokena
 - - problem skalabilnosti rješenja u velikim sustavima
- Self-encoded token
 - Sve informacije potrebne za provjeru dozvole pristupa resursu se nalaze unutar tokena
 - + brza provjera valjanosti tokena (potpis, vrijeme valjanosti, izdavalatelj, kome je namijenjen ...)
 - + brz proces provjere dozvole pristupa resursu
 - - tokeni se ne mogu poništiti, čeka se istek njihove valjanosti (ili uvođenje liste poništenih tokena -> problem skalabilnosti!)
 - - nema potpune slike svih aktivnih tokena u sustavu

Refresh token

Refresh
token

- *Refresh* token predstavlja vjerodajnice za reizdavanje pristupnih tokena od strane **autorizacijskog poslužitelja**
 - izdaje se kod prvog izdavanja pristupnog tokena
 - izdavanje *refresh* tokena od strane autorizacijskog poslužitelja nije obavezno
- Tipičan sadržaj tokena - atributi (parovi ime=vrijednost) ? **JWT**
- Neproziran (*opaque*) za klijenta
- Sadržaj tokena interpretiran od strane autorizacijskog poslužitelja
 - za razliku od inicijalnog izdavanja tokena, *vlasnik resursa* nije uključen u postupak reizdavanja novog pristupnog tokena
- Vrijeme valjanosti *refresh* tokena je **znatno dulje od vremena valjanosti pristupnog tokena**
 - gubitak pristupnog tokena predstavlja srednjoročan sigurnosni rizik
 - novi refresh tokeni mogu biti izdani u postupku reizdavanja pristupnih tokena (time invalidiraju stare *refresh* tokene)
- Povlačenje autorizacije pristupa resursu za nekog klijenta (korišteni self-encoded tokeni)
 - + odbija se reizdavanje pristupnih tokena (i refresh tokena)
 - - postoji vremenski odmak između čina povlačenja autorizacije i samog gubitka pristupa klijenta (najviše do vremena valjanosti pojedinog pristupnog tokena)

Primjeri politika vremena valjanosti tokena

Pristupni
token



token

admin

customer

privileged

1 min

1 min

ne izdaje se

24 h

nema isteka

ne izdaje se

1 h

24 h

4 h

Trajanje kodova i tokena

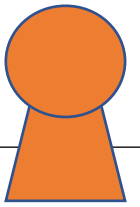


Trajanje kodova i tokena je kako slijedi:

- autorizacijski kod (eng. authorization code): 1 minuta
- pristupni token (eng. access token) i ID token: 1 sat
- token za osvježavanje (eng. refresh token): 8 sati

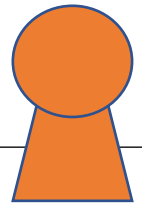
Refresh
token

Pristupni
token



- Klijent mora biti registriran pri autorizacijskom poslužitelju
 - Norma ne propisuje metodu, primjeri:
 - registracija putem obrasca,
 - dostavljanje certifikata i potpisanih podataka o klijentu na stranu autorizacijskog poslužitelja
 - postupak otkrivanja klijenta (engl. *client discovery*) korištenjem sigurnog kanala ...
- Tijekom registracije moraju se priložiti sljedeći podaci o klijentu
 - tip klijenta (povjerljiv ili javan)
 - URI krajnjih komunikacijskih točaka za preusmjeravanje (*client redirection URIs*)
 - ostale informacije korištene od autorizacijskog poslužitelja (ime, URI aplikacije, opis, logo, pravne informacije ...)

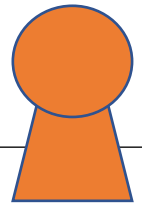
Klijent (II)



- Tipovi klijenata

- povjerljiv klijent (*confidential*) – sposoban čuvati tajnost povjerenih mu vjerodajnica
 - npr. aplikacija weba na sigurnom poslužitelju, korištenje štićenih komunikacijskih kanala
 - vlasnici resursa pristupaju aplikaciji korištenjem sigurnog web sučelja (user agent) i kom. kanala (HTTPS)
- javni klijent (*public*) – nije sposoban čuvati tajnost povjerenih mu vjerodajnica
 - npr. klijentski programi izvršavani na računalu vlasnika resursa, web aplikacije izvršavane unutar preglednika, IoT uređaji, komunikacija neštićenim komunikacijskim kanalom
 - vlasnici resursa koriste preglednik weba i dohvaćene JavaScript temeljene klijentske web aplikacije

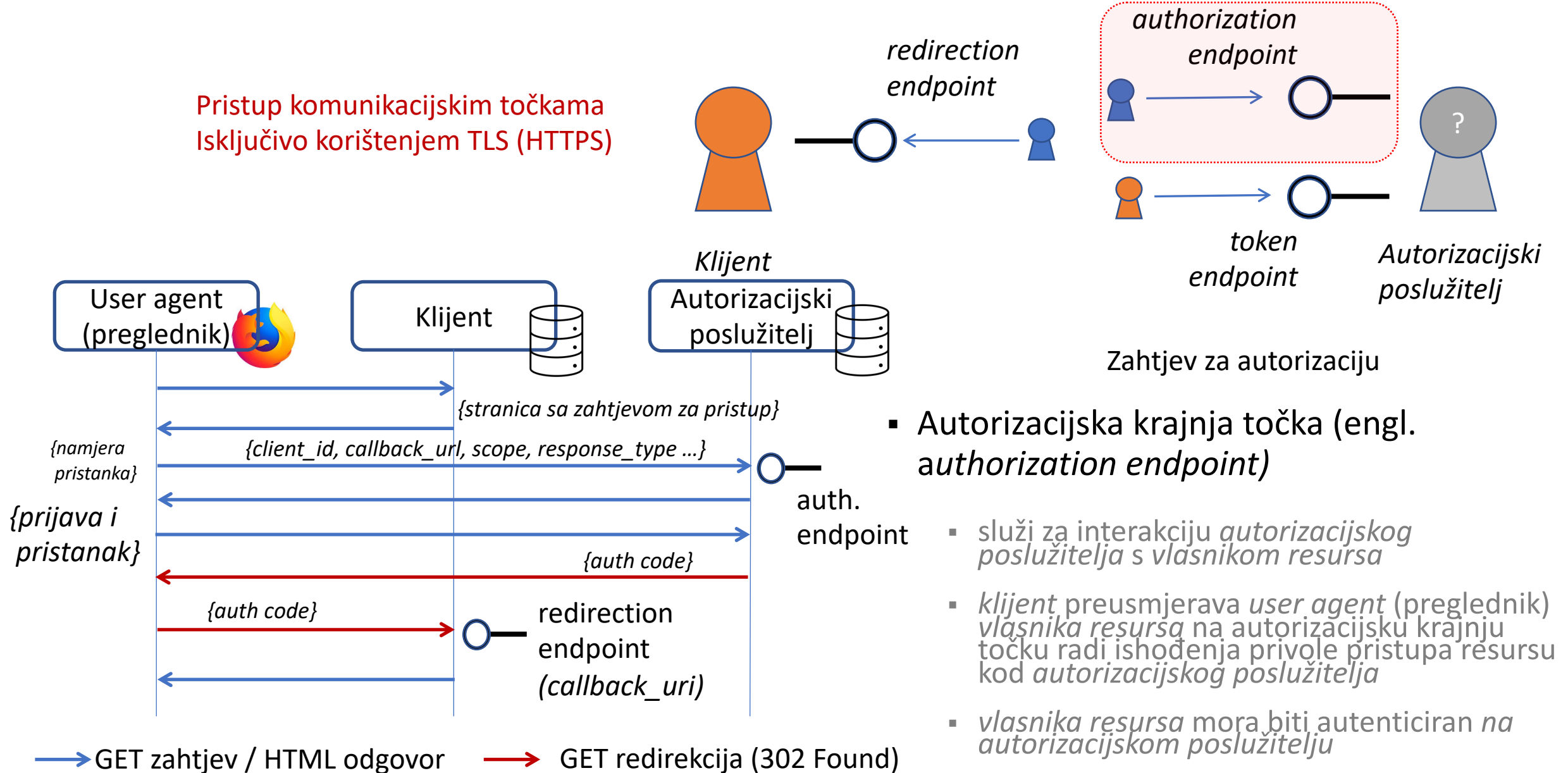
Klijent (III)



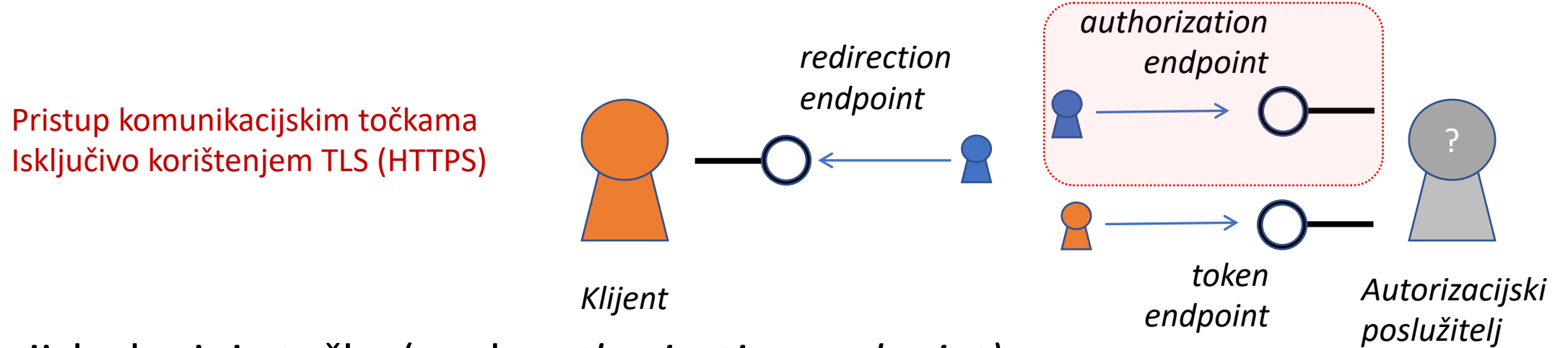
- Registriranom klijentu autorizacijski poslužitelj izdaje jedinstveni identifikator klijenta (clientID)
 - jedinstven na razini pojedinog autorizacijskog poslužitelja
 - nije tajna informacija, ne može se samostalno koristiti za autentikaciju
- Autorizacijski poslužitelj može provjeriti identitet *povjerljivih klijenata* na osnovu dijeljene tajne ili PKI
 - korištenje dijeljene tajne
 - korištenje para javnog/tajnog ključa
- Korištenje autentikacije s *javnim klijentima* za utvrđivanje njihova identiteta je moguće, ali sigurnosno rizično
 - Napadač može jednostavno preuzeti dijeljenu tajnu i predstavljati se kao klijent
 - Preglednik: view source, inspect ...
 - Pokretni klijent: reverzno inženjerstvo aplikacije

OAuth 2.0 API (I) – autorizacijska krajnja točka

Pristup komunikacijskim točkama
Isključivo korištenjem TLS (HTTPS)

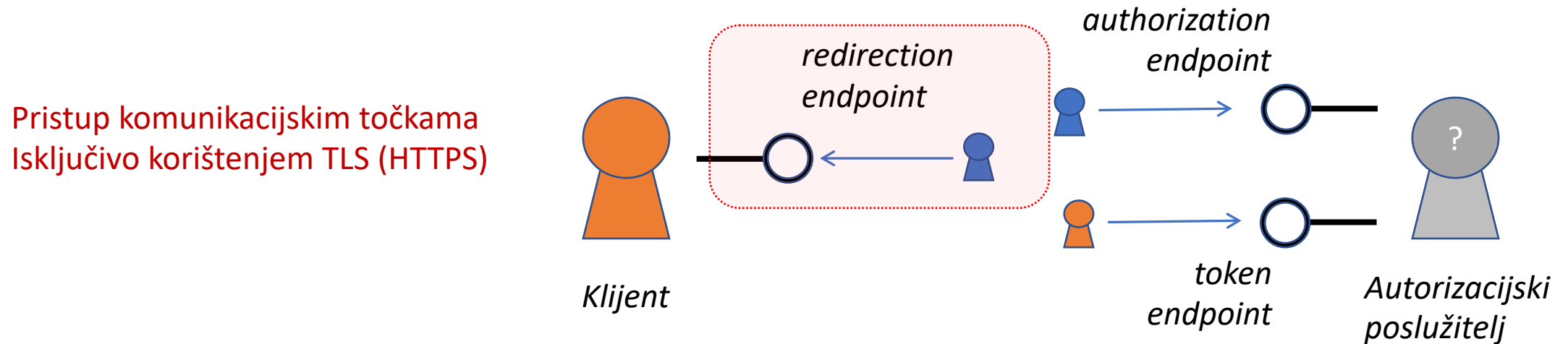


OAuth 2.0 API (II) – autorizacijska krajnja točka



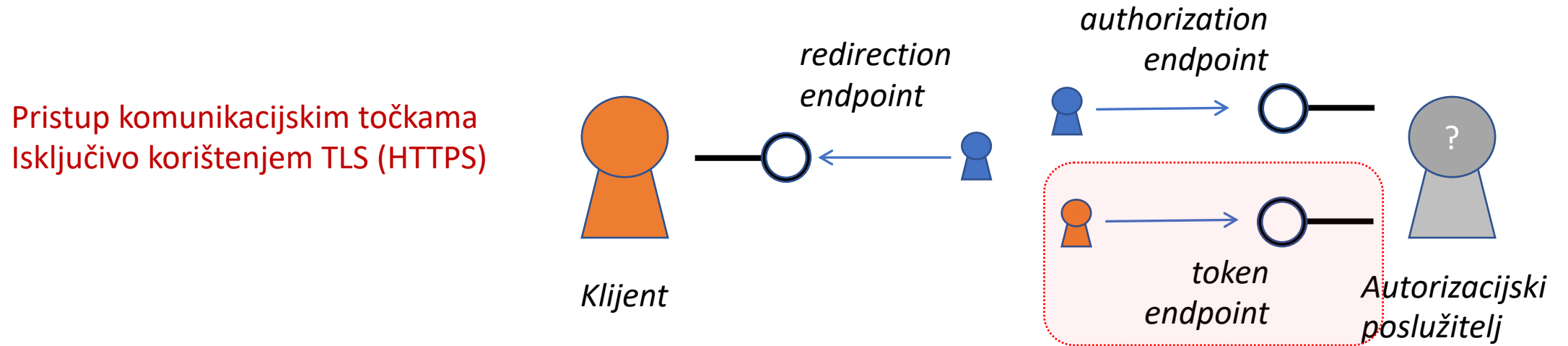
- Autorizacijska krajnja točka (engl. *authorization endpoint*)
 - unutar URI na koji se *user agent* preusmjerava, *klijent* uključuje parametre upita (*query string*):
 - ***response_type*** – definira traženi postupak autorizacije („code”, „token” ili ime proširenog tipa autorizacije)
 - ***scope*** – jedan ili više traženih profila pristupa grupi resursa (npr. profile, email, contacts:read, contacts:write ...)
 - ***client_id*** – identifikator klijenta kod autorizacijskog poslužitelja (dobiven tijekom postupka registracije *klijenta* kod *aut. poslužitelja*)
 - ***callback_uri*** – URI redirekcijske komunikacijske točke na *klijentu*, a na koju treba biti proslijeđen odgovor *aut. poslužitelja* prema *klijentu*
 - ***state*** – jedinstveni niz znakova kojim se identificira transakcija
 - ...
 - *autorizacijski poslužitelj*, posredstvom *user agenta vlasnika resursa*, *klijentu* vraća dozvolu korištenja resursa u željenom obliku (u ovisnosti o navedenom *response_type*)
 - ***autorizacijski kod*** ili
 - ***pristupni token***

OAuth 2.0 API (III) – krajnja točka preusmjeravanja



- Krajnja točka preusmjeravanja (engl. *redirection endpoint*)
 - *preduvjet: URI klijentskog redirection endpointa treba biti registriran kod autorizacijskog poslužitelja tijekom procesa registracije klijenta*
 - nakon završetka procesa davanja privole pristupa *klijenta* resursu vlasnika, vlasnikov *user agent* (preglednik weba) se preusmjerava natrag na *klijenta* gdje nastavlja interakciju s klijentom
 - tijekom preusmjeravanja, vlasnikov *user agent* (preglednik) služi za prenošenje dozvole pristupa resursu od autorizacijskog poslužitelja ka klijentu
 - Autorizacijski kod prenosi se unutar *query string* dijela URL-a

OAuth 2.0 API (IV) – token krajnja točka



- Token krajnja točka (engl. *token endpoint*)
 - korištena od klijenta za dohvat pristupnog tokena, kao dio zahtjeva nužno priložiti
 - dozvolu korištenja resursa (*authorization code*) ili valjani *refresh* token
 - identitet klijenta koji traži pristup resursima: *client_id*
 - *tajni podatak dijeljen između klijenta i autorizacijskog poslužitelja, kako bi klijent dokazao svoj identitet*
 - poslužitelj vraća *pristupni token* (sa zapisanim pravima pristupa) i, opcionalno, *refresh token*
 - *nužan siguran komunikacijski kanal između klijenta i autorizacijskog poslužitelja (HTTPS)*

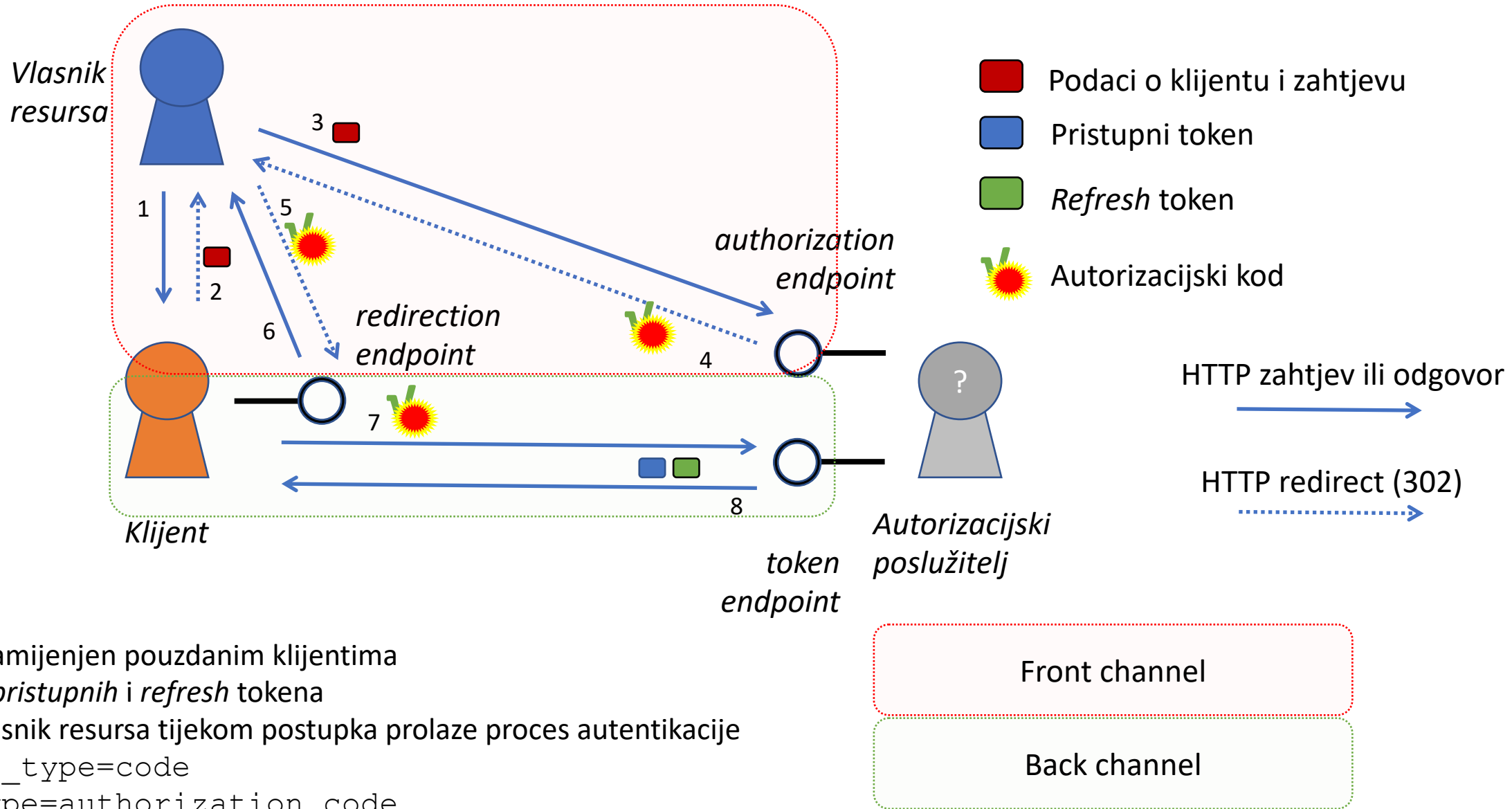
Postupci (tijekovi) autorizacije

1. Postupak temeljen na izdavanju **autorizacijskog koda**
2. **Implicitni** postupak izdavanja pristupnog tokena
3. Postupak temeljen na vjerodajnici u obliku **zaporke vlasnika resursa**
4. Postupak temeljen na **vjerodajnici klijenta**
5. *Dodatni postupci (proširenja norme)*

- Postupci se razlikuju po:

- stupnju sigurnosti vjerodajnice (tipu klijenta)
- korištenju komunikacijskih krajnjih točaka klijenta i autorizacijskog poslužitelja
- uloga uključenih u tok autorizacije
- tipu klijenta (povjerljiv ili javan) uključenog u postupak autorizacije
- ...

Izdavanje autorizacijskog koda (I)



- Postupak namijenjen pouzdanim klijentima
- Izdavanje i *pristupnih* i *refresh* tokena
- I klijent i vlasnik resursa tijekom postupka prolaze proces autentikacije
- `response_type=code`
- `grant_type=authorization_code`

Izdavanje autorizacijskog koda (II)

1. Vlasnik (korištenjem preglednika weba) pristupa klijentu i traži uslugu za koji je nužan pristup resursima koji su njegovo vlasništvo
2. Klijent pakira potrebne **parametre** za zahtjev za pristup resursu, te ih uključuje u *redirect* odgovor vlasniku, upućujući ga na *authorization endpoint* autorizacijskog poslužitelja
3. Vlasnik pristupa *authorization endpoint-u* autorizacijskog poslužitelja i prenosi **parametre** klijenta, vrši se postupak autentikacije klijenta i autorizacije zahtjeva za pristup resursima (pristanak klijenta)
4. Autorizacijski poslužitelj formira **autorizacijski kod** i uključuje ga u *redirect* odgovor vlasniku, upućujući ga na *redirection endpoint* klijenta
5. Vlasnik pristupa *redirect endpoint-u* klijenta i prenosi **autorizacijski kod** poslužitelja
6. Klijent vraća sadržaj vlasniku ili radi daljnju redirekciju na sadržaj (još nije pristupljeno resursu!)
7. Klijent šalje zahtjev za tokenom na *token endpoint* autorizacijskog poslužitelja i prilaže **autorizacijski kod**, vrši se autentikacija klijenta i provjera koda
8. Autorizacijski poslužitelj vraća klijentu *pristupni token* i, opcionalno, *refresh token*
9. ... (klijent korištenjem *pristupnog tokena* pristupa resursu na *poslužitelju resursa*, nakon isteka roka trajanja *pristupnog tokena*, traži novi od *autorizacijskog poslužitelja* novi *pristupni token* na osnovu *refresh tokena*) ...

Izdavanje autorizacijskog koda (III)

3. Zahtjev za autorizacijom (**front channel** korisnikov preglednik izvršava redirect na URL naveden od strane klijenta)

```
GET /authorize?response_type=code&client_id=s6BhdRkqt3&state=xyz&  
redirect_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb HTTP/1.1  
Host: server.example.com
```

4. Autorizacijski kod (**front channel** redirect odgovor vraćen korisnikovu pregledniku od strane autorizacijskog poslužitelja)

```
HTTP/1.1 302 Found  
Location: https://client.example.com/cb?code=Splxl0BeZQQYbYS6WxSbIA&state=xyz
```

7. Zahtjev za pristupnim tokenom (**back channel**)

```
POST /token HTTP/1.1  
Host: server.example.com  
Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW  
Content-Type: application/x-www-form-urlencoded  
  
grant_type=authorization_code&code=Splxl0BeZQQYbYS6WxSbIA  
&redirect_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb
```

8. Pristupni i **refresh** tokeni (**back channel**)

```
HTTP/1.1 200 OK  
Content-Type: application/json; charset=UTF-8  
Cache-Control: no-store  
Pragma: no-cache  
  
{  
  "access_token": "2YotnFZFEjrlzCsicMWpAA",  
  "token_type": "example",  
  "expires_in": 3600,  
  "refresh_token": "tGzv3J0kF0XG5Qx2TlKWIA",  
  "example_parameter": "example_value"  
}
```

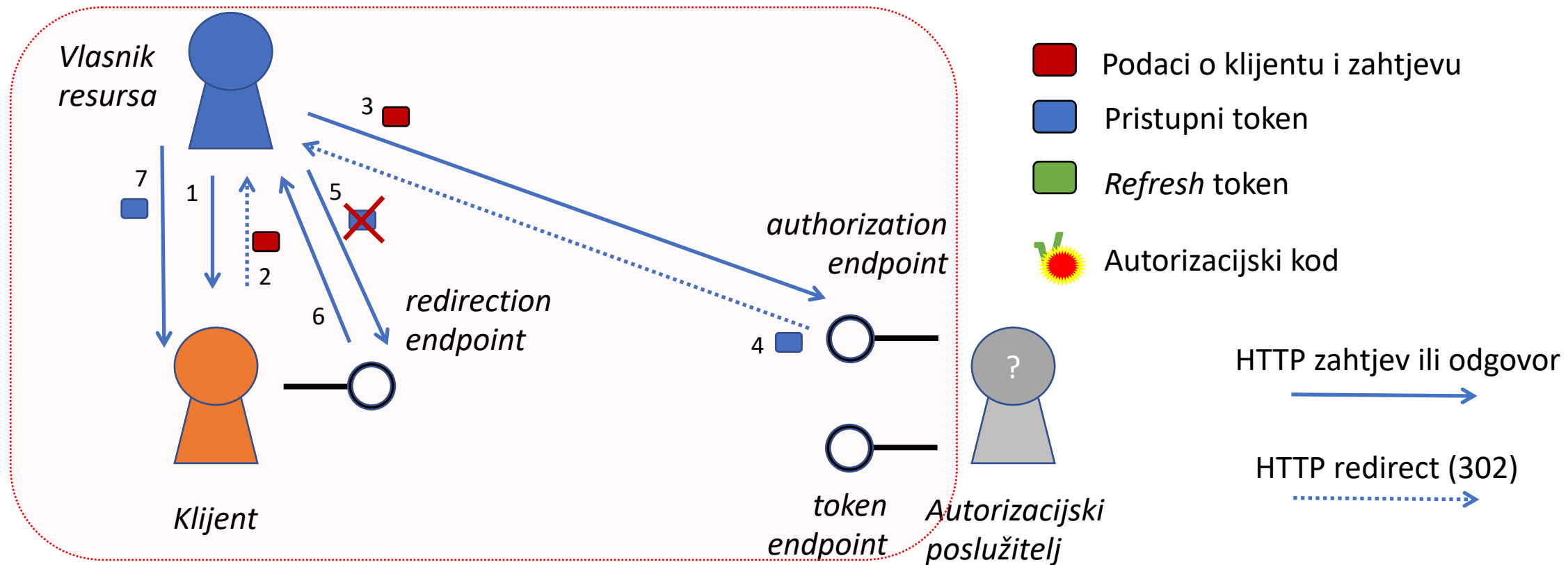
Proof Key for Code Exchange - PKCE (I)

- Proširenje postupka izdavanja autorizacijskog koda ([RFC7636](#))
 - Problem presretanja autorizacijskog koda prenošenog od *autorizacijskog poslužitelja*, putem *korisničkog agenta*, do *klijenta* (**front channel**)
 - Proširenje postupka kod zahtjeva autorizacijskog koda:
 - Klijent generira novu *tajnu* (slučajan niz znakova duljine 43-128) kod svakog pokretanja postupka traženja autorizacije
 - U inicijalnom zahtjevu uključuje i parametre:
 - `code_challenge`= Base64URL(SHA256(tajna)) ili tajna
 - `code_challenge_method`= [plain ili sha256]
- Proširenje postupka zamjene autorizacijskog koda za pristupni token
 - U zahtjev se dodaje parametar:
 - `code_verifier`=tajna
 - Autorizacijski poslužitelj provjerava
 - `code_challenge_method`=plain ? `code_challenge == code_verifier`
 - `code_challenge_method`=sha256 ? `code_challenge == Base64URL(SHA256(code_verifier))`

Proof Key for Code Exchange - PKCE (II)

- Inicijalno zamišljeno kao metoda sprječavanja napada krađom i injekcijom autorizacijskog koda
- Koristi se za poboljšanje sigurnosti kod izdavanja autorizacijskog koda za
 - Aplikacije koje se ne mogu oslanjati na klijentsku tajnu (engl. *client secret*)
 - Mobilne aplikacije
 - Single-page web aplikacije
 - ... ali i za sve ostale aplikacije koje koriste klijentsku tajnu

Implicitni postupak (I)



- Postupak namijenjen *javnim* klijentima s dostupnim (poznatim) *redirection endpoint* URI-jem
- *De facto* za klijenske aplikacije (JavaScript unutar preglednika, mobilna aplikacija)
- Izdavanje samo pristupnih tokena (nema *autorizacijskog koda* niti *refresh* tokena)
- Ne uključuje postupak autentikacije *klijenta*
- `response_type=token`

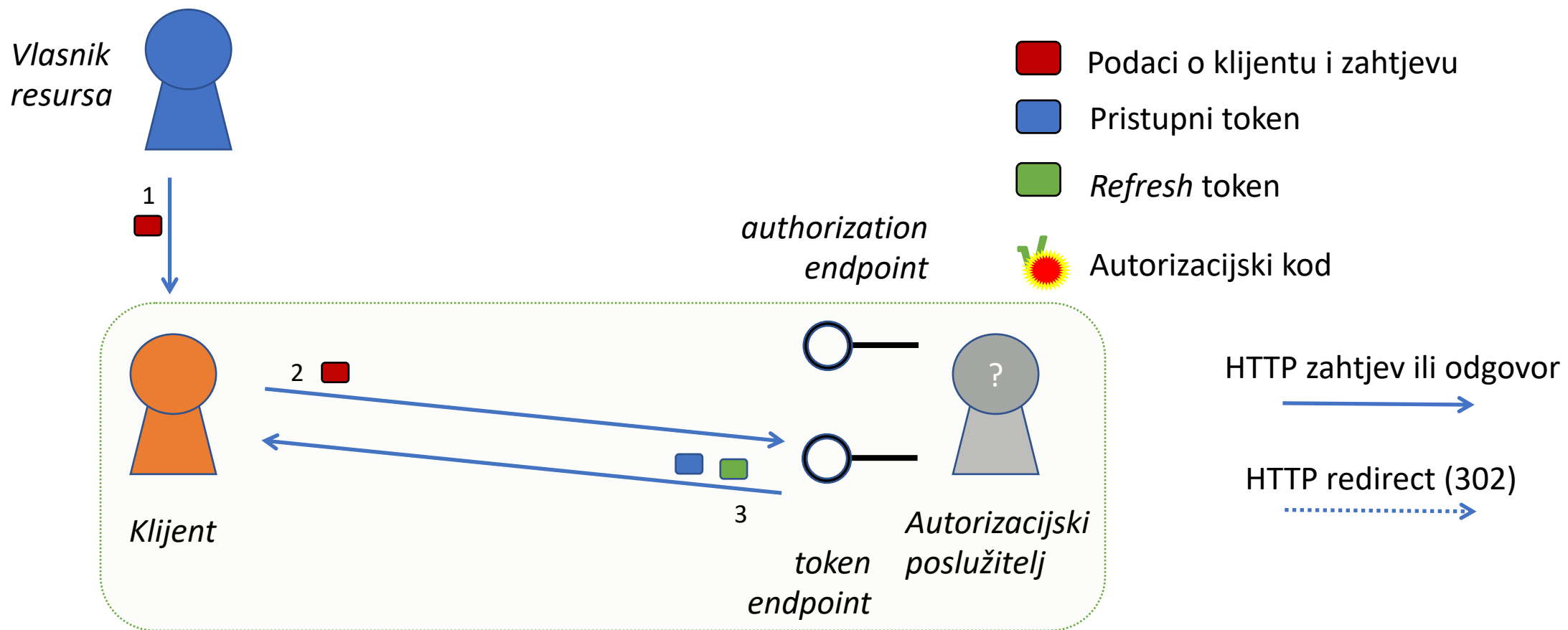
Implicitni postupak (II)

1. *Vlasnik resursa* (korištenjem preglednika weba) pristupa *klijentu* i traži uslugu za koji je nužan pristup resursima na *poslužitelju resursa*, a koji su njegovo vlasništvo
2. *Klijent* pakira potrebne **parametre** za zahtjev za pristup resursu, te ih uključuje u odgovor vlasniku, upućujući ga na *autorizacijsku krajnju točku autorizacijskog poslužitelja*
3. *Vlasnik resursa* (njegov preglednik – *user agent*) pristupa *autorizacijskoj krajnjoj točki autorizacijskog poslužitelja* i prenosi **parametre klijenta**, vrši se *autentikacija vlasnika resursa* i autorizacija zahtjeva za pristup resursima (privola vlasnika resursa za akcijama navedenim u opsegu)
4. *Autorizacijski poslužitelj* formira **pristupni token** i uključuje ga u *redirect* odgovor prema *user-agentu vlasnika resursa*, upućujući ga na *redirekcijsku krajnju točku klijenta* (token je u fragment dijelu URL-a)
5. *Vlasnik resursa* (putem preglednika – *user-agenta*) pristupa *redirekcijskoj krajnjoj točki klijenta*, ali još ne prenosi **pristupni token**
6. *Klijent* vraća *vlasniku resursa* sadržaj stranice, uključujući i skriptu (*JavaScript kod*) koja može pristupiti u vlasnikovom pregledniku pohranjenom **pristupnom tokenu**
7. *Vlasnikov* preglednik izvršava skriptu koja čita lokalno pohranjeni **pristupni token** (i proslijeđuje ga *klijentu*)
8. ... (klijent korištenjem **pristupnog tokena**, do njegova isteka, pristupa resursu na *poslužitelju resursa*) ...

🔍 *Klijent može biti:*

- zaseban program (*poslužitelj*) ili
- (*JavaScript*) skripta izvršavana unutar preglednika (*user-agent*) *vlasnika resursa*

Zaporka vlasnika resursa (I)



- Vlasnik resursa jednokratno „posuđuje” svoje vjerodajnice (zaporku) klijentu za privremeni pristup
- Podrazumijeva visoki stupanj povjerenja između vlasnika resursa i klijenta
- `grant_type=password`

Zaporka vlasnika resursa (II)

1. *Vlasnik resursa* (korištenjem preglednika weba) prosljeđuje *klijentu* vlasnikovo **korisničko ime i zaporku** za autentikaciju na *autorizacijskom poslužitelju*
2. *Klijent* zahtijeva izdavanje **pristupnog tokena** na *token endpoint-u* *autorizacijskog poslužitelja*, zahtjevu prilaže vlasnikovo **korisničko ime i lozinku**
3. Autorizacijski poslužitelj vrši postupak autentikacije *klijenta* i validacije **korisničkog ime i zaporke** *vlasnika resursa*, autorizira zahtjev za pristup resursima, *klijentu* vraća **pristupni token** (i, opcionalno **refresh token**)
4. ... (klijent korištenjem **pristupnog tokena**, do njegova isteka, pristupa resursu na *poslužitelju resursa*) ...

2. Zahtjev za pristupnim tokenom

```
POST /token HTTP/1.1
Host: server.example.com
Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW
Content-Type: application/x-www-form-urlencoded

grant_type=password&username=johndoe&password=A3ddj3w
```

Autentikacija klijenta

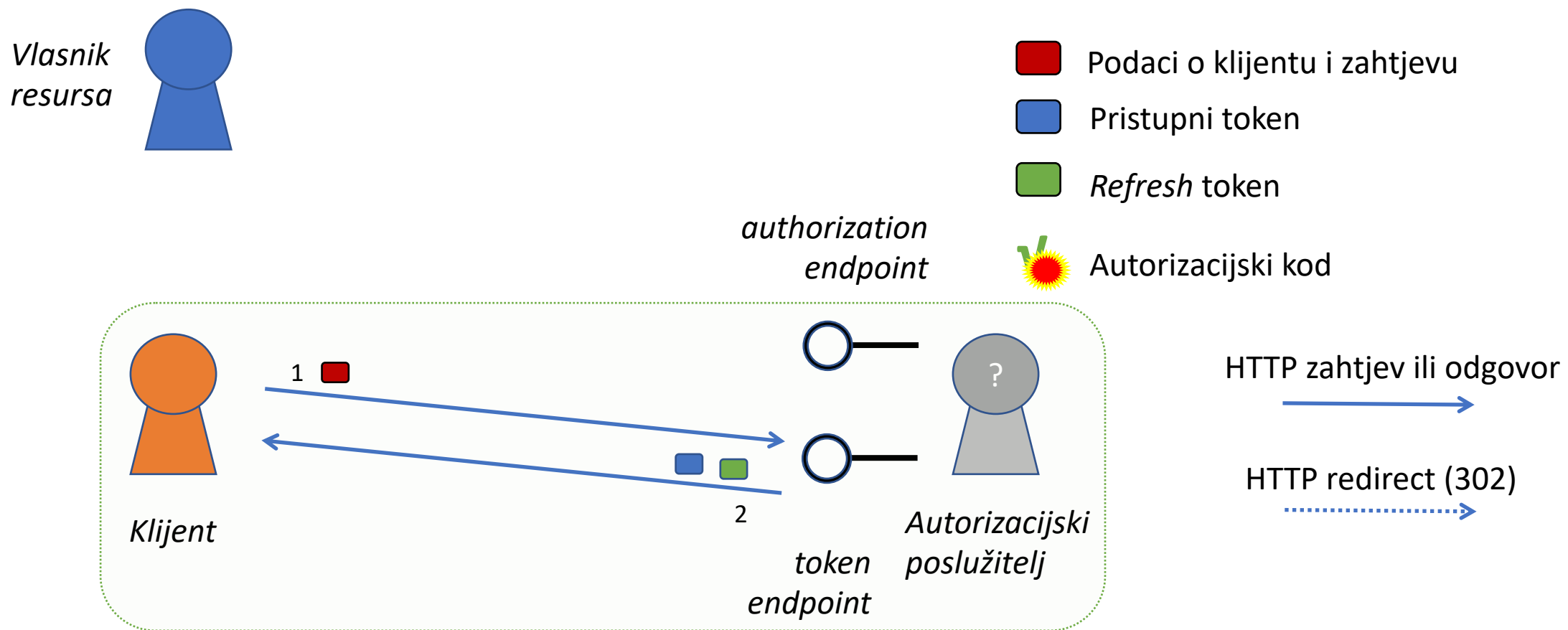
Autentikacija vlasnika resursa

3. Pristupni i refresh tokeni

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Cache-Control: no-store
Pragma: no-cache
```

```
{
  "access_token": "2YotnFZFEjrlzCsicMWpAA",
  "token_type": "example",
  "expires_in": 3600,
  "refresh_token": "tGzv3JOkF0XG5Qx2TlKWIA",
  "example_parameter": "example_value"
}
```

Klijentske vjerodajnice (I)



- Klijent pristupa resursima pod svojom kontrolom ili od prije ima prava nad pristupanim skupom resursa
- Smije se upotrebljavati samo za povjerljiv tip klijenata
- `grant_type=client_credentials`

Klijentske vjerodajnice (II)

1. Klijent zahtijeva autentikaciju i izdavanje *pristupnog tokena* na *token endpoint-u* autorizacijskog poslužitelja
2. Autorizacijski poslužitelj vrši postupak autentikacije klijenta, autorizira zahtjev za pristup resursima, klijentu vraća *pristupni token*

1. Zahtjev za pristupnim tokenom

```
POST /token HTTP/1.1
Host: server.example.com
Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW
Content-Type: application/x-www-form-urlencoded
```

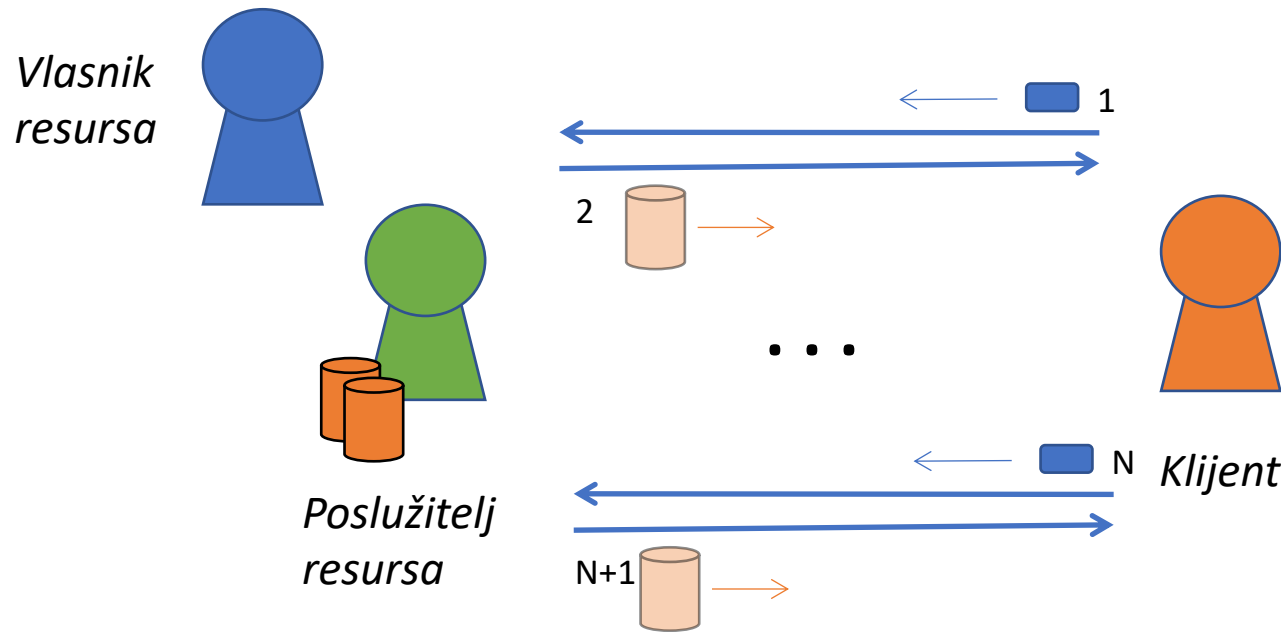
```
grant_type=client_credentials
```

2. Pristupni token

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache
```

```
{
  "access_token": "2YotnFZFEjrlzCsicMWpAA",
  "token_type": "example",
  "expires_in": 3600,
  "example_parameter": "example_value"
}
```

Pristup resursu korištenjem pristupnog tokena



Provjera valjanosti pristupnog tokena:

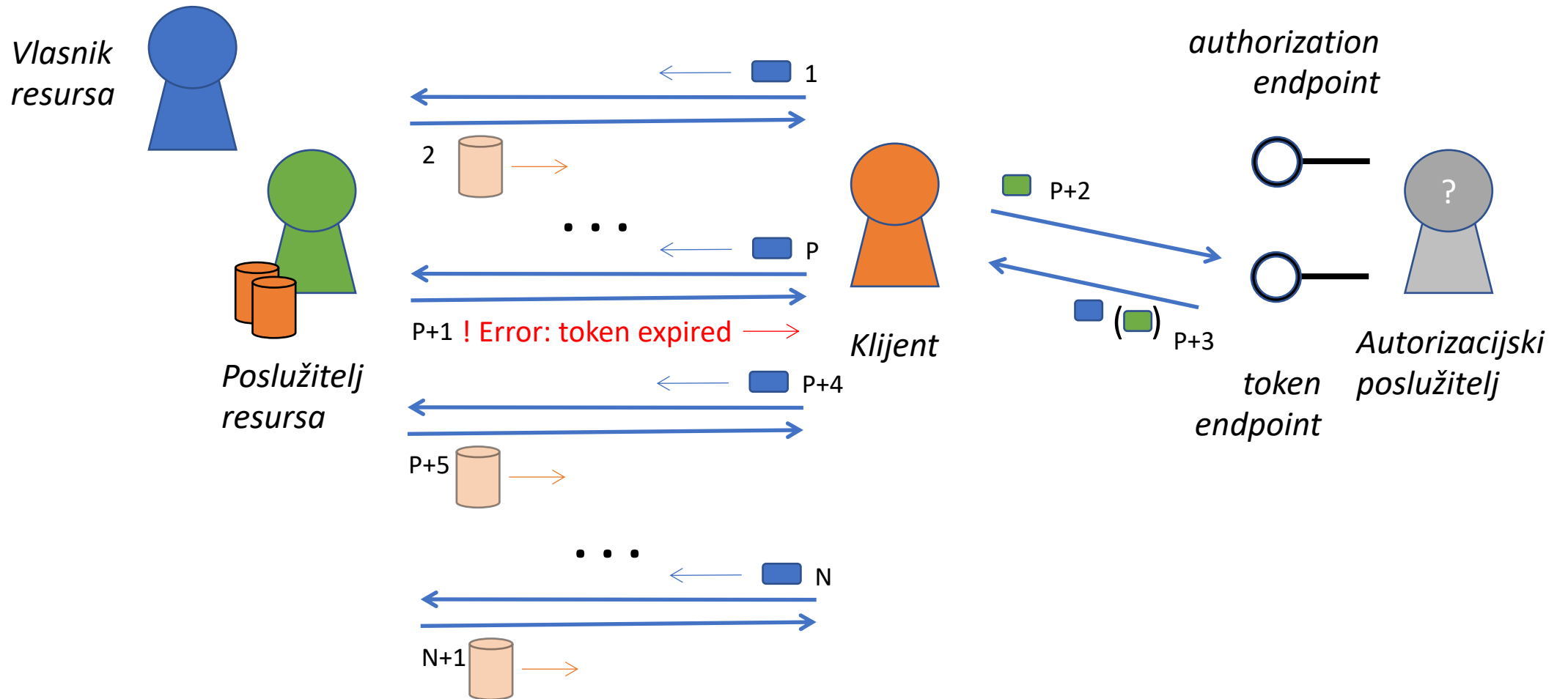
- struktura tokena (h.p.s)
- integritet tokena
- izdavalac tokena (*aut. poslužitelj*)
- ciljni korisnik tokena (*klijent*)
- razdoblje valjanosti tokena (od-do)
- opseg tokena
- (vlasnik resursa)
- ...

1. Zahtjev za pristupom resursu (primjer Bearer i OAuth-HTTP-MAC metoda autentikacije)

```
GET /resource/1 HTTP/1.1
Host: example.com
Authorization: Bearer mF_9.B5f-4.1JqM
```

```
GET /resource/1 HTTP/1.1
Host: example.com
Authorization: MAC id="h480djs93hd8",
  nonce="274312:dj83hs9s",
  mac="kDZvddkndxvhGRXZhvuDjEWhGeE="
```

Obnavljanje pristupnih tokena



- Nakon isteka roka valjanosti *pristupnog tokena*, *klijent* (ako posjeduje *refresh token*) može zatražiti drugi *pristupni token* bez kontaktiranja *vlasnika resursa* (sve dok ne istekne valjanost dozvole pristupa resursu izdane od vlasnika)
- U postupku izdavanja novog *pristupnog tokena*, *autorizacijski poslužitelj* može povremeno izdati i novi *refresh token* - novi *refresh token* zamjenjuje prethodno korišteni *refresh token* (sprječava višestruko korištenje *refresh tokena*)

Otvoreno računarstvo

9. Sigurnost u otvorenim sustavima

- Autentikacija i autorizacija
- HTTP sheme autentikacije
- Tokeni (JWT)
- Sigurni komunikacijski kanal (TLS/SSL, HTTPS)
- Delegiranje prava (OAuth2)
- Autentikacija (OpenID, OpenIDConnect)

OpenID, OpenIDConnect, SAML

- Pružanje usluge provjere identiteta u sustavu (SSO):
 - [OpenID Connect](#)
 - poseban tip OAuth2 autorizacijskog poslužitelja
 - Izdaje tokene identiteta
 - [SAML](#) (Security Assertion Markup Language)
 - Skup profila za razmjenu autentikacijskih i autorizacijskih podataka
 - *SAML identity provider* – pruža usluge upravljanja identitetima i autentikacije
 - *SAML service provider* – koristi podatke SAML identity providera za upravljanje pristupa uslugama

OAuth2 i identitet korisnika

- *Korisnik (vlasnik resursa) daje ovlasti za pristup njegovim resursima trećoj strani - klijentu*
 - *Klijent koristi autorizacijski kod, pristupni token i refresh token*
 - Niti jedna od navedenih informacija ne sadrži informacije o korisniku
 - *Jedino autorizacijski poslužitelj zna tko je korisnik – korisnik mora biti autenticiran na autorizacijskom poslužitelju kako bi poslužitelj bio siguran u identitet onoga tko daje ovlasti klijentu*
- OAuth2 nije zamišljen kao mehanizam autentikacije, samo autorizacije !
- Što ako korisnički identitet tretiramo kao resurs?
 - Koristimo autorizacijski poslužitelj za:
 - Autentikaciju korisnika
 - Autorizaciju pristupa resursu „identitetu korisnika”
 - Poslužujemo resurs „identitet korisnika” (autorizacijski poslužitelj i poslužitelj resursa su jedno)



- Autorizacijski poslužitelj koji podržava OIDC – *pružatelj identiteta* (engl. identity provider)
- Pružatelj identiteta može se koristiti za prijavu na više usluga, tzv. SSO (engl. Single Sign-On)
- Primjeri takvih javnih usluga:
 - Google Accounts
 - Facebook
 - Twitter
 - Auth0
 - Apple iCloud
 - Microsoft
 - GitHub
 - ...
 - [AAI EduHr OIDC](#)


Sign Up


Email


By clicking any of the Sign Up buttons,
I agree to the [terms of service](#)

SIGN UP

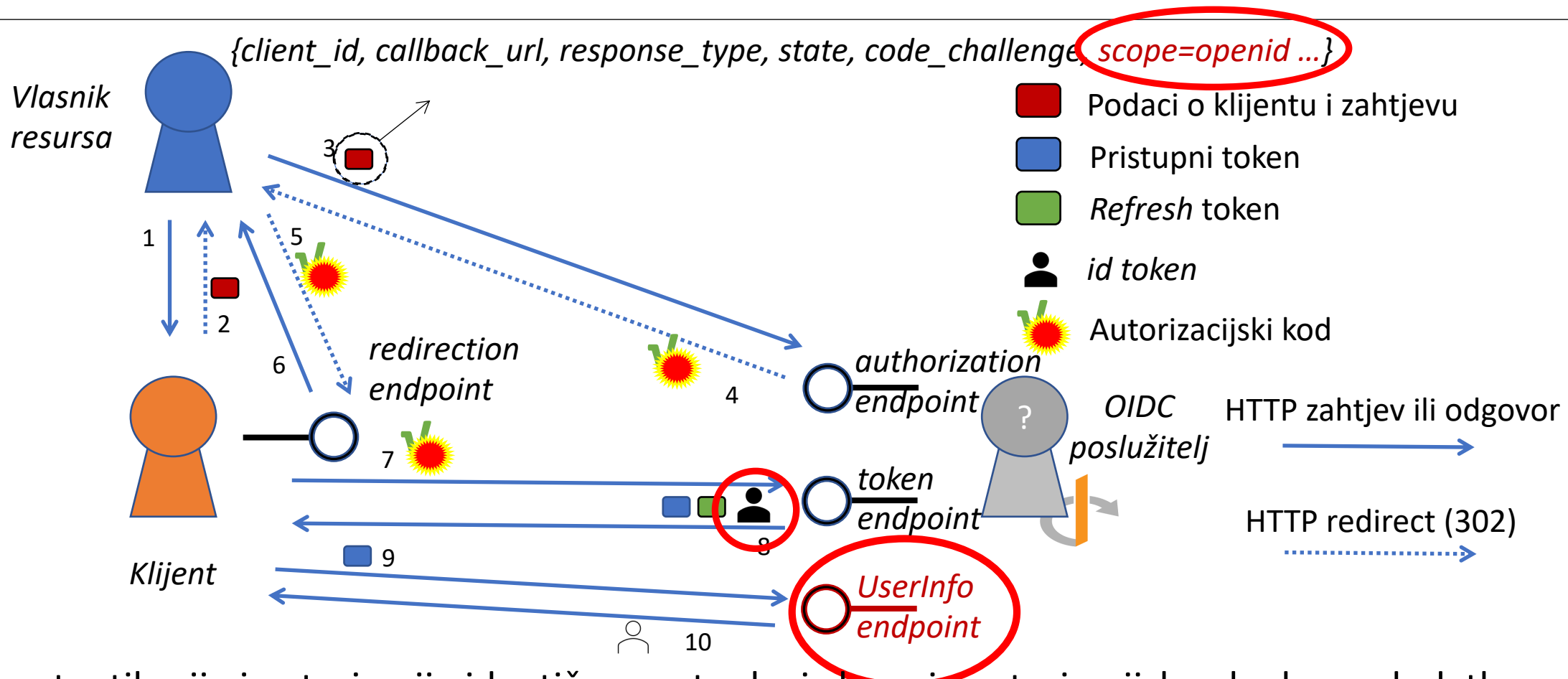
or

 SIGN UP WITH GITHUB

 SIGN UP WITH GOOGLE

 SIGN UP WITH MICROSOFT

Autentikacija korisnika (I)



- Postupak autentikacije i autorizacije identičan postupku izdavanja autorizacijskog koda, uz dodatke:
 - profil *openid*
 - id token*
 - UserInfo* krajnja točka na *OIDC poslužitelju*

- *Korisnikov agent (preglednik)* je usmjeren na *IODC poslužitelj (koraci 2 i 3)*
 - Unutar parametara (query string) uobičajenih za OAuth2 tijek, *klijent* mora postaviti parametar *scope* koji sadrži opseg *openid*
 - Unutar parametra *scope* potrebno navesti i dodatne OIDC opsege ako klijent traži pristup informacijama o korisniku (email adresa, telefon ...) – ti podaci će biti isporučeni u obliku JWT tvrdnji unutar id tokena i bit će im omogućen pristup na pristupnoj točki /UserInfo
- Standardni OIDC opsezi
 - openid -> obavezan (OAuth poslužitelj takav zahtjev tretira kao autentikacijski, ne autorizacijski)
 - profile - osnovni podaci o korisniku
 - email - email adresa korisnika
 - address - adresa korisnika
 - phone - telefonski broj korisnika
- Nestandardni OIDC opsezi
 - ovisno o implementaciji OIDC poslužitelja
 - primjer: AAI@Edu.hr IODC [opsezi](#)

`https://login.aaiedu.hr/sso/module.php/oidc/authorize.php?response_type=code&client_id=neki-id-klijenta&redirect_uri=https://neki-uri-za-preusmjerenje.primjer.hr&scope=openid profile
&state=neki-state-132&nonce=neki-nonce-123`



■ Id token

- Vraćen klijentu u sklopu zamjene autorizacijskog koda za *pristupni token* (koraci 7 i 8)
- Format JWT
- Podaci o korisniku u obliku JWT tvrdnji
 - sadržane tvrdnje ovise o opsezima navedenim prilikom traženja autorizacijskog koda (profile, email ...)

Zaglavlje ID tokena

```
{
  "typ": "JWT",
  "alg": "RS256",
  "kid": "69d8c46574"
}
```

ID ključa koji je korišten za potpisivanje tokena
Na osnovu ove informacije moguće je dohvatiti i
odabrati javni ključ za provjeru potpisa JWT-a

Korisni podaci u ID tokenu

```
{
  "iss": "https://login.aaiedu.hr",
  "aud": "6e55295209782b7b2",
  "jti": "c44f4cffcc84f7990f7a1d5b2c",
  "nbf": 1602674470,
  "exp": 1602675070,
  "sub": "bfal605be44a50a7c",
  "iat": 1602674470,
  "nonce": "dtnmeBL5HVnhQkIR",
  "name": "Ivan Horvat",
  "family_name": "Horvat",
  "given_name": "Ivan",
  "preferred_username": "ihorvat@primjer.hr",
  "email": "ivan.horvat@primjer.hr",
  "hrEduPersonUniqueNumber": [
    "LOCAL_NO: 1234",
    "OIB: 12345678912",
    "JMBAG: 1234567891"
  ]
}
```

JWT tvrdnje

OIDC tvrdnje

Dodatne informacije o korisniku

- komunikacijska krajnja točka /UserInfo
 - konkretan URL ovisi o postavkama poslužitelja
 - upitu je obavezno priložiti *pristupni token*
 - vraća JSON zapis s podacima o korisniku, sukladno pravima koja su tražena unutar parametra scope zahtjeva (profile email telephone ... + profili specifični za konkretnu implementaciju OIDC poslužitelja)
- Uglavnom se koristi ako:
 - je korišten implicitan tijek autorizacije u kojem se izdaje samo *pristupni token*
 - prilikom traženja autorizacijskog koda nije naveden *openid* opseg