UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Blaž Hafnar

KOMUNIKACIJSKI PROTOKOL ZA SPLETNO DISTRIBUCIJO HOTELSKIH NAMESTITEV

DIPLOMSKO DELO UNIVERZITETNEGA ŠTUDIJA

Mentor: prof. dr. Nikolaj Zimic

Ljubljana, 2008

Zahvala

Na tem mestu bi se rad zahvalil mentorju prof. dr. Nikolaju Zimicu za strokovno pomoč in usmerjanje pri izdelavi diplomske naloge. Zahvalil bi se tudi zaposlenim iz podjetja MAiS Informacijski sistemi, ki so me seznanili s področjem informacijskih sistemov za vodenje hotelov. Ob tej priložnosti bi se zahvalil tudi svojim staršem, ki so mi bili v podporo skozi vsa leta študija ter vsem ostalim, ki so mi v tem času kakorkoli stali ob strani.

Kazalo vsebine

Povzetek ... 1
Abstract ... 3
1. Uvod ... 5
2. Predstavitev spletnega rezervacijskega sistema ... 7
2.1. Spletni rezervacijski portal ... 8
2.2. Vmesnik hotelskega rezervacijskega sistema ... 8
2.3. Komunikacijski strežnik ... 9
2.4. Plačilni servisi ... 10
3. Varnost v elektronskih komunikacijah ... 11
3.1. Kriptografske tehnike ... 11
3.1.1. Simetrična kriptografija ... 11
3.1.2. Asimetrična kriptografija ... 13
3.1.3. Kombiniranje simetričnega in asimetričnega šifriranja ... 14
3.1.4. Ostale kriptografske tehnike ... 15
3.2. Uporaba kriptografskih tehnik ... 16
3.2.1. Šifriranje sporočil ... 16
3.2.2. Integriteta sporočil ... 16
3.2.3. Digitalni podpis ... 17
3.2.4. Avtentikacija uporabnikov ... 17
3.3. Infrastruktura javnih ključev ... 18
3.3.1. Certifikatne agencije ... 18
3.3.2. Prijavne službe ... 19
3.3.3. Digitalni certifikati ... 19
3.3.4. Delovanje klientov ... 21
4. Oblika sporočil ... 23
4.1. Osnove XML jezika ... 23
4.1.1. Struktura dokumenta ... 23
4.1.2. Definiranje sintakse in pravilno oblikovan dokument ... 27
4.1.3. Razširitve XML jezika ... 30
4.2. Šifriranje in podpisovanje XML dokumentov ... 31
4.2.1. Kanonična oblika dokumenta ... 31
4.2.2. Podpisovanje ... 32
4.2.3. Šifriranje ... 34
4.2.4. Kombiniranje podpisovanja in šifriranja ... 36
5. Standard sintakse sporočil ... 37
6. Prenos sporočil ... 39
6.1. Struktura SOAP sporočila ... 39
6.2. Prenos SOAP sporočila ... 41
7. Postavitev komunikacijskega protokola ... 43
7.1. Osnovna zgradba sporočil ... 43
7.1.1. Glava sporočila ... 44
7.1.2. Telo sporočila ... 46
7.1.3. Rokovanje s sporočili ... 48
7.2. Postopki izmenjave sporočil ... 50
7.2.1. Avtentikacija ... 51
7.2.2. Rezervacija ... 52
7.2.3. Posodabljanje statične vsebine ... 60
7.2.4. Preverjanje prisotnosti odjemalcev ... 60

7.3. Vpliv komunikacije na komponente sistema ... 60
7.3.1. Priprava sporočila ... 61
7.3.2. Sprejemanje in pošiljanje sporočila ... 61
7.3.3. Varnost podatkov ... 62
8. Sklepne ugotovitve ... 65
9. Priloge ... 67
9.1. Primer sporočil postopka rezervacije ... 67
9.1.1. Povpraševanje po razpoložljivosti hotela ... 67
9.1.2. Povpraševanje po ponudbi hotela ... 68
9.1.3. Izračun cene izbrane ponudbe ... 69
9.1.4. Določitev garancije ... 70
9.1.5. Potrditev rezervacije ... 71
9.2. Primer sporočil postopka posodabljanja statične vsebine ... 73
9.3. Primer sporočil postopka preverjanja prisotnosti odjemalcev ... 74
Uporabljeni viri ... 75
Izjava o avtorstvu ... 77

Kazalo slik

Slika 1: Zgradba rezervacijskega sistema 7
Slika 2: Problem blokovnega šifriranja podatkov 12
Slika 3: Certifikacijska pot v hierarhičnem modelu zaupanja 20
Slika 4: Struktura SOAP sporočila 39
Slika 5: XML shema elementa *WebResHeader* 45
Slika 6: XML shema elementa *FaultCode* 47
Slika 7: Sprejemanje sporočila 49
Slika 8: Posredovanje sporočila 50
Slika 9: XML shema elementa *Authentication* 51
Slika 10: Diagram zaporedja avtentikacije odjemalca 52
Slika 11: Diagram zaporedja postopka rezervacije 54
Slika 12: Prikaz ponudbe sob na spletnem portalu 56

Seznam kratic

3DES	(*Triple DES*) Trojni DES.
A5/1	Algoritem za tekoče šifriranje, ki se uporablja v GSM telefoniji.
AES	(*Advanced Encryption Standard*) Standard za šifriranje podatkov. Naslednik DES algoritma za blokovno šifriranje.
Base64	Znakovna predstavitev binarnih podatkov s 64 znaki, ki predstavljajo podmnožico znakov, ki jih vsebuje večina kodni tabel.
CA	(*Certification Authority*) Certifikatna agencija.
CAST-128	Algoritem za blokovno šifriranje, zasnovan z uporabo CAST metode.
CBC	(*Chiper Block Chaining*) Način blokovnega šifriranja podatkov.
CRL	(*Certificate Revocation List*) Seznam preklicanih digitalnih certifikatov.
CRS	(*Computer Reservation System*) Računalniški rezervacijski sistem.
DES	(*Data Encryption Standard*) Standard za šifriranje podatkov. Algoritem za blokovno šifriranje.
DH	(*Diffie Hellman*) Postopek za izmenjavo ključa.
DTD	(*Document Type Definitions*) Sintaksa za opis strukture XML dokumenta.
ECB	(*Electronic Code Book*) Elektronska kodirna knjiga. Način blokovnega šifriranja.
GDS	(*Global Distribution System*) Globalni distribucijski sistem.
GSM	(*Global System for Mobile communications*) Svetovni standard mobilnih komunikacij.
HMAC	(*Hash Message Authentication Code*) Postopek za zagotavljanje integritete sporočila.
HTML	(*HyperText Markup Language*) Označevalni jezik za oblikovanje spletnih strani.
HTTP	(*Hypertext Transfer Protocol*) Protokol za prenos hiperteksta.
HTTPS	(*Secure Hypertext Transfer Protocol*) Varni HTTP, razširja HTTP z zaščito prenosa podatkov.
IDEA	(*International Data Encryption Algorithm*) Algoritem za blokovno šifriranje.
IP	(*Internet Protocol*) Internetni protokol.
IPsec	(*Internet Protocol Security*) Skupek protokolov za zaščito internetnega protokola.
LDAP	(*Lightweight Directory Access Protocol*) Protokol za dostop do X.500 omrežnega imenika.
MAC	(*Message Authentication Code*) Postopek za zagotavljanje integritete sporočila.
MD5	(*Message Digest algorithm 5*) Algoritem za izračun zgoščevalne funkcije.
MIT	*Massachusetts Institute of Technology*.
NBS	*National Bureau of Standards*.
NIST	*National Institute of Standards and Technology*.
OCSP	(*Online Certificate Status Protocol*). Protokol za poizvedovanje o statusu digitalnih certifikatov v realnem času.
OTA	*OpenTravel Alliance* je skupnost podjetij, ki razvijajo OpenTravel standard.
PFS	(*Perfect Forward Secrecy*) Lastnost postopka izmenjave ključa, ki preprečuje odkritje sejnega ključa.

PGP	(*Pretty Good Privacy*) Model zaupanja v infrastrukturi javnih ključev.
PKI	(*Public Key Infrastructure*) Infrastruktura javnih ključev.
RC4	(*Rivest Cipher 4*) Algoritem za tekoče šifriranje.
RIPEMD-160	(*RACE Integrity Primitives Evaluation Message Digest*) Algoritem za izračun zgoščevalne funkcije.
RSA	(*Rivest Shamir Adleman*) Algoritem za asimetrično šifriranje.
S/MIME	(*Secure Multipurpose Internet Mail Extensions*) Standard za šifriranje in podpisovanje elektronske pošte.
SGML	(*Standard Generalized Markup Language*) Standardiziran meta jezik, za definiranje označevalnih jezikov.
SHA-1	(*Secure Hash Algorithm*) Algoritem za izračun zgoščevalne funkcije.
SMS	(*Short Message Service*) Protokol za izmenjavo kratkih tekstovnih sporočil med mobilnimi napravami.
SMTP	(*Simple Mail Transport Protocol*) Protokol za prenos elektronske pošte.
SOAP	(*Simple Object Access Protocol*) Protokol za izmenjavo strukturiranih informacij.
SQL	(*Structured Query Language*) Standardni programski jezik za delo s podatkovnimi bazami.
SSL	(*Secure Sockets Layer*) Kriptografski protokol za varno komunikacijo preko računalniških omrežij.
TCP	(*Transmission Control Protocol*) Protokol za nadzor prenosa podatkov.
TLS	(*Transport Layer Security*) Kriptografski protokol za varno komunikacijo preko računalniških omrežij.
TTP	(*Trusted Third Party*) Zaupanja vredna tretja oseba.
URI	(*Uniform Resource Identifier*) Niz znakov, ki identificira ali poimenuje vir.
W3C	(*World Wide Web Consortium*) Mednarodni konzorcij, ki skrbi za standardizacijo in razvoj spleta in z njim povezanih tehnologij.
X.500	Standard, ki definira storitev omrežnega imenika.
X.509	Standard, ki definira strukturo digitalnih certifikatov in z njimi povezane postopke.
XHTML	(*Extensible HyperText Markup Language*) Označevalni jezik za oblikovanje spletnih strani, ki ustreza XML pravilom.
XML	(*Extensible Markup Language*) Razširljiv označevalni jezik.
XOR	(*Exclusive OR*) Logični operator za logično operacijo ekskluzivni ali.

Povzetek

Elektronsko poslovanje preko interneta odpira podjetjem novo možnost neposredne prodaje svojih produktov. Številni rezervacijski portali hotelom omogočajo prodajo svojih storitev preko interneta. Portali pri tem v večini primerov prevzamejo vlogo posrednika pri sklepanju posla, s čimer pa hoteli izgubijo neposreden nadzor nad rezervacijami lastnih namestitev.

Diplomsko delo predstavi rešitev, ki omogoča hotelom in njihovim bodočim gostom neposredno sklepanje rezervacij preko spletnega rezervacijskega portala. Za ta namen definira poseben komunikacijski protokol, ki omogoča sklepanje rezervacij v realnem času. Definirani protokol hotelskemu rezervacijskemu sistemu omogoča popoln nadzor nad vodenjem procesa rezervacije, ki ga preko spletnega portala koristi uporabnik.

Za definiranje protokola vsebina obdela principe varovanja podatkov in način predstavitve le-teh v sporočilih. Vsebina tako predstavi osnove kriptografije, ki so potrebne za zagotavljanje varnosti podatkov v sporočilih. Ob tem so izpostavljeni tudi digitalni certifikati, ki omogočijo proces avtentikacije preko računalniških omrežij. Poleg tega vsebina predstavi tudi XML (*Extensible Markup Language*) standard in iz njega izpeljane standarde, s pomočjo katerih je definirana oblika sporočil protokola in postopki za obdelavo le-teh. Pri opisu komunikacije z danim protokolom so definirani tudi podatki, katerih izmenjava je potrebna pri procesu rezervacije hotelskih namestitev. Za predstavitev le-teh v sporočilih pa je izpostavljen OpenTravel standard. Ta predstavlja standard sintakse sporočil za izmenjavo informacij med informacijskimi sistemi v potovalni industriji. Mednje sodijo tudi sistemi za vodenje hotelov.

Ključne besede:

komunikacijski protokol, hotelska rezervacija, zaščita podatkov, XML, OpenTravel

Abstract

By e-commerce via Internet, a new way of direct selling of a company's products has been offered. For the purpose of online distribution of hotel capacities, numerous booking portals have been established. In most cases, a booking portal takes a complete control over the sales, which causes hotels to lose their direct control over the booking of their capacities.

The bachelor thesis introduces a solution which enables future hotel guests to make a direct booking at a selected hotel via a booking portal, thereby closing a booking arrangement between a hotel and its guest directly. For this purpose, a special communication protocol enabling such a booking in real-time is defined. In such a way, hotels have a complete control over the booking process, by which such a real-time booking service is offered to the Internet users.

In order to define the communication protocol, the thesis focuses on data security principles and protocol message data representation. It discusses cryptography basics, which is essential for ensuring of data security. Digital certificates for user authentication over a computer network are introduced. Furthermore, the thesis explains the XML (*Extensible Markup Language*) standard and its extensions, by which a message structure and a message processing model of the protocol are defined. Also, when describing a communication based on the defined protocol, data which have to be exchanged during such a booking process are determined. For this data representation, the OpenTravel standard is used. This standard defines message syntax for information exchange among information systems in travel industry, where also hotel management systems are to be found.

Key words:

Communication Protocol, Hotel Booking, Data Security, XML, OpenTravel

1. Uvod

Internet danes predstavlja globalno komunikacijsko infrastrukturo, ki neprestano raste in ima vsak dan večjo vlogo tako v poslovnem svetu kot tudi pri posameznikovih vsakdanjih opravilih. Sam pojav interneta je omogočil različne oblike elektronskega poslovanja, kot so spletne trgovine, bančništvo, licitacije, rezervacije idr. Izraz elektronsko poslovanje se uporablja za oblike poslovanja, ki za avtomatizacijo sklepanja poslovnih transakcij uporabljajo informacijske tehnologije, kamor sodi tudi internet. Ta oblika poslovanja ima vse večji porast, saj omogoča premagovanje časovnih in geografskih ovir. Sklepanje poslov v takem smislu je možno 24 ur na dan 7 dni v tednu ne glede na fizično lokacijo ponudnika in povpraševalca. Postopek je enostaven, hiter, potrošniku pa hkrati omogoča pregled in primerjavo ponudbe več ponudnikov na enem mestu.

Spletne rezervacije kot obliko elektronskega poslovanja srečamo v okviru računalniških rezervacijskih sistemov oziroma CRS-jev (*Computer Reservation System*). CRS-ji so bili sprva v uporabi letalskih družb, ki so jih uporabljale za pregled in vodenje prodaje letalskih vozovnic [2, 7]. Prvi tak avtomatiziran rezervacijski sistem je leta 1946 uvedla letalska družba American Airlines. Tehnološki napredek na področju računalništva in komunikacij je omogočil širitev dostopnosti CRS-jev sprva do potovalnih agencij, nato pa z razcvetom interneta do vsakega posameznika. To je omogočilo tudi uporabo CRS-jev na drugih področjih, kot je najem in rezervacija hotelskih sob, rent-a-car, nakup vozovnic za železniški promet idr. Večjim CRS-jem, ki združujejo prodajo in rezervacijo več različnih družb, pravimo tudi globalni distribucijski sistemi oziroma GDS-ji (*Global Distribution System*). Primeri največjih GDS-jev danes so sistemi Amadeus, SABRE, Galileo, WorldSpan in Patheo. Ti poleg nakupa letalske karte običajno ponujajo tudi rent-a-car in rezervacijo hotelske namestitve.

V okviru računalniškega rezervacijskega sistema deluje tudi spletni portal hotel.si, ki je namenjen distribuciji kapacitet slovenskih hotelov. Rezervacije preko sistema se izvedejo v realnem času in s takojšnjo potrditvijo. To pomeni, da se rezervacija zabeleži v hotelskem rezervacijskem sistemu v trenutku, ko uporabnik potrdi rezervacijo na spletnem vmesniku. Takoj zatem je uporabnik obveščen o uspešnosti zabeleženja rezervacije. Spletni portal ima pri tem le vlogo vizualne navigacije, medtem ko je vsa logika razpolaganja namestitev, določanja cene in načina plačila vodena s strani avtomatiziranega procesa, realiziranega v okviru posameznega hotelskega rezervacijskega sistema. Portal sam tako ne hrani nobene informacije o razpoložljivostih posameznega hotela, kot to počno drugi podobni portali. Hranjenje podatkov o razpoložljivosti posameznih hotelov zahteva (v nekaterih primerih ročne) sinhronizacije podatkov, zaradi česar prihaja do velikih zakasnitev med izvedbo in potrditvijo posameznih rezervacij. Hoteli morajo pri tem vedno vnaprej zadržati delež kapacitet, ki ga namenijo spletni prodaji. V primeru rezervacij v realnem času to ni potrebno, ker se postopek avtomatizirane rezervacije izenači z rezervacijo, izvedeno v recepciji hotela. Cena rezervacij v realnem času so le zanemarljive časovne zakasnitve, ki so posledica izmenjave informacij med portalom in hotelskim rezervacijskim sistemom.

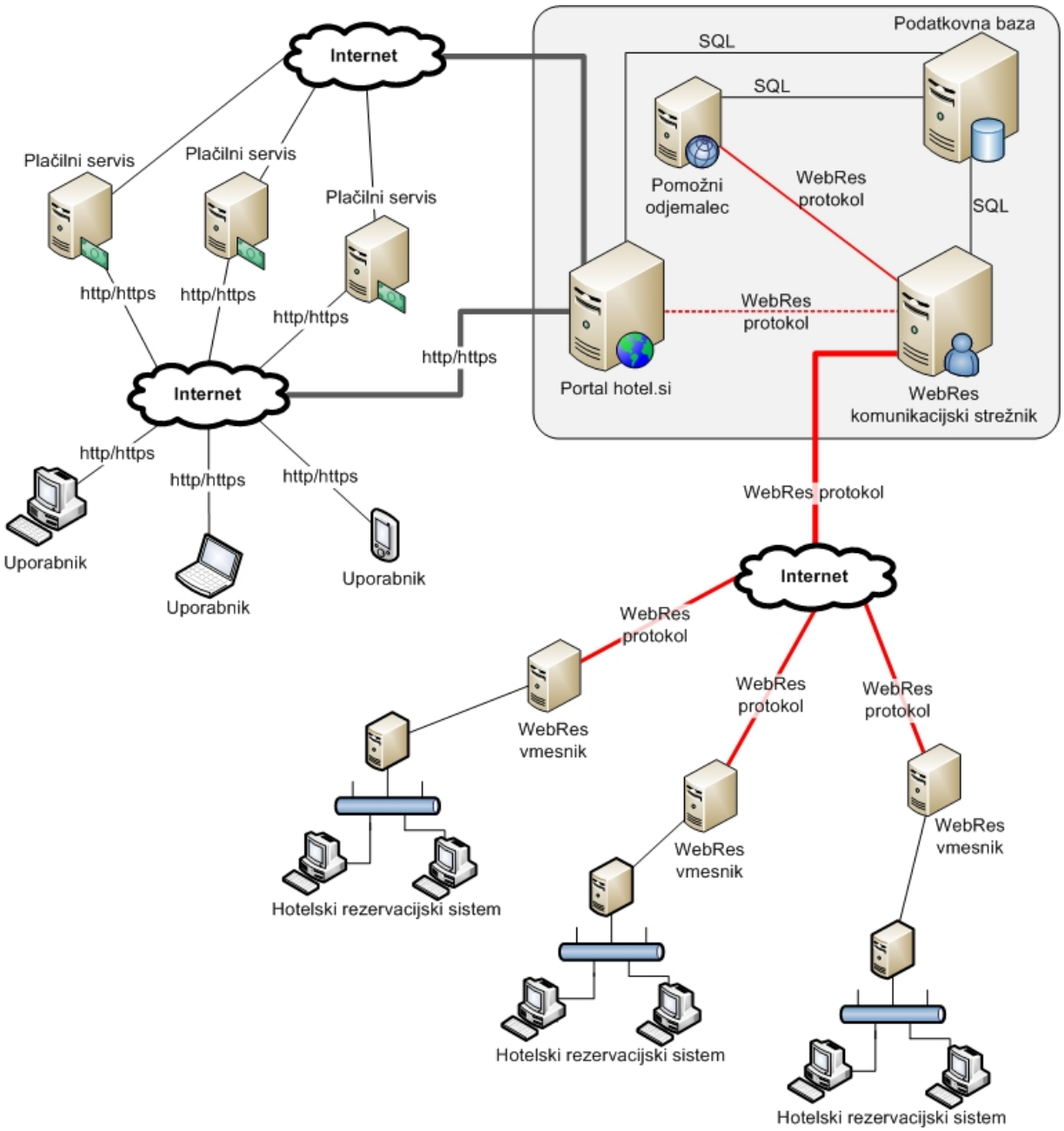
Za nudenje storitev spletnih rezervacij hotelom na takšen način je potrebno vzpostaviti način komuniciranja med portalom in raznolikimi sistemi za vodenje hotelov. Zato potrebujemo komunikacijski protokol, ki bo definiral pravila in postopke komuniciranja. Ta morajo zagotoviti varno in učinkovito komunikacijo med sistemi, ki se nahajajo na fizično ločenih lokacijah.

Diplomsko delo definira komunikacijski protokol, ki v danem sistemu rešuje problem rezervacij hotelskih namestitev v realnem času. Pri zagotavljanju varnosti podatkov med prenosi protokol poseže po kriptografskih tehnikah. Pri tem z uporabo digitalnih certifikatov izkorišča infrastrukturo javnih ključev za avtentikacijo pošiljateljev sporočil. Protokol prav tako naslovi problema oblikovanja sporočil in predstavitve podatkov v njih. Z uporabo sporočil v XML (*Extensible Markup Language*) obliki protokol prikaže prednost združitev standardov, izpeljanih iz iste osnove. Pri tem z vključitvijo SOAP (*Simple Object Access Protocol*) standarda protokol definira zunanjo obliko sporočil. Za predstavitev vsebine sporočil protokol uporabi OpenTravel standard, ki definira sporočila za izmenjavo potrebnih informacij. Kriptografske tehnike protokol vključuje v samo zgradbo z uporabo XML šifriranja in podpisovanja. Na podlagi vsega tega protokol tudi definira pravila in postopke, s katerimi omogoči sam proces rezervacij v realnem času.

Vsebina diplomskega dela se razdeli na tri dele. V prvem delu, ki ga predstavlja 2. poglavje, je prikazana zgradba celotnega rezervacijskega sistema in razložena interakcija med posameznimi komponentami sistema. Nato sledi drugi del vsebine, ki je namenjen podrobnejšemu pregledu postopkov in standardov, s katerimi je protokol zgrajen. V 3. poglavju je narejen pregled kriptografije in predstavitev infrastrukture javnih ključev. Pri tem so izpostavljene tudi grožnje komuniciranja in tehnike za preprečevanje le-teh. V 4. poglavju je predstavljen XML standard s svojimi razširitvami. Tu sta podrobneje predstavljeni tudi razširitvi, ki sta namenjeni šifriranju in podpisovanju XML dokumentov. V 5. poglavju je predstavljen OpenTravel standard, ki z uporabo XML shem definira sintakso sporočil, namenjenih sporazumevanju med informacijskimi sistemi v potovalni industriji. Ta del vsebine se zaključi s 6. poglavjem, ki predstavi SOAP protokol, katerega strukturo definirani protokol tudi razširi. Zadnji del vsebine predstavlja 7. poglavje, ki definira sam protokol in opiše komunikacijo med komponentami sistema z uporabo le-tega.

2. Predstavitev spletnega rezervacijskega sistema

Spletni portal hotel.si je del porazdeljenega računalniškega rezervacijskega sistema. Vsak del sistema opravlja svojo nalogo v rezervacijskem procesu. Tako spletni portal poskrbi za vodenje uporabnika skozi proces rezervacije, vmesniki hotelskih rezervacijskih sistemov priskrbijo ustrezne podatke za izvršitev rezervacije namestitve v hotelu, komunikacijski strežnik pa za kontrolo komuniciranja med vmesniki in portalom. Vključitev storitev plačilnih servisov potrošniku omogoča plačilo rezerviranih hotelskih kapacitet.



Slika 1: Zgradba rezervacijskega sistema

Poleg teh treh komponent sistema je tu še podatkovna baza, ki je namenjena hranjenju podatkov, potrebnih za delovanje sistema, kot so osnovni podatki o hotelih in podatki, potrebni za avtentikacijo pripadajočih vmesnikov. Taka zasnova sistema omogoča podvajanje komponent sistema v primeru potrebe po širitvi zaradi zasičenosti ali redundance sistema. Sistem ima prav tako lastnosti odprtega sistema, ker omogoča priklop različnih rezervacijskih sistemov. Za to pa je potreben dobro definiran protokol, ki omogoča različnim proizvajalcem hotelskih rezervacijskih sistemov enostavno implementacijo vmesnikov.

Namen direktne interakcije med spletnim distribucijskim sistemom in rezervacijskimi sistemi posameznih hotelov je v realizaciji postopka rezervacije v realnem času. To pomeni, da niti potrošnik niti ponudnik ne čaka več kot je potrebno. Ko uporabnik potrdi izbrano namestitev in vnesene podatke, se rezervacija s pomočjo vmesnika zabeleži v rezervacijskem sistemu hotela. Zatem pa vmesnik takoj ustvari potrdilno pismo hotelirja, ki se preko portala posreduje uporabniku. Posel sklepata hotelir in gost neposredno, ker je celoten postopek v rokah vmesnika hotelskega rezervacijskega sistema. Pri tem se portal s storitvijo pojavlja na ravni ponudnika kot visoko vredna infrastruktura za spletno distribucijo in ne kot pravna oseba v pogodbenem razmerju med strankama.

2.1. Spletni rezervacijski portal

Spletni portal je realiziran kot spletna aplikacija in kot komponenta sistema predstavlja neposredni vmesnik za interakcijo z uporabnikom. Uporabnik potrebuje za interakcijo s sistemom le računalnik oz. kakšno drugo napravo, povezano z internetom in naloženim brskalnikom, kar omogoča komunikacijo s spletnim strežnikom z uporabo HTTP (*Hypertext Transfer Protocol*) in HTTPS (*Secure Hypertext Transfer Protocol*) protokolov. HTTPS protokol je razširjena oblika HTTP protokola, ki poleg osnovnih lastnosti nudi še zaščito prenosa podatkov zaupne narave z uporabo šifriranja in avtentikacije.

Uporabnik lahko s pomočjo portala išče hotele in njihove namestitve po različnih kriterijih, kot so: lokacija, rang, datum prihoda in odhoda, tip namestitve, dodane storitve itd. Pri zahtevi uporabnika za pregled kapacitet določenega hotela portal s poizvedbo od vmesnika pridobi potrebne podatke in jih prikaže. Na enak način portal poskrbi za navigacijo v rezervacijskem procesu. Tako spletni vmesnik ponuja uporabniku možnosti in od njega zahteva podatke, kot to zahteva logika, ki je del vmesnika hotelskega rezervacijskega sistema. V primeru, ko uporabnik izbere namestitveno enoto in vmesnik zahteva plačilo akontacije, portal uporabniku najprej ponudi možne oblike plačila. Naloga portala je v tem primeru tudi interakcija z izbranim plačilnim servisom. Portal mora opraviti predajo zahtevka za plačilo plačilnemu servisu in sprejeti potrdilo o uspešnosti uporabnikovega plačila od plačilnega servisa. Ob uspešno zaključeni rezervaciji uporabnik preko portala prejme tudi potrdilno pismo.

2.2. Vmesnik hotelskega rezervacijskega sistema

Dobro definiran protokol za komunikacijo med portalom in vmesnikom omogoča priklop in izgradnjo vmesnika za kateri koli hotelski rezervacijski sistem. Vmesnik igra vlogo avtomatiziranega receptorja, ki je sposoben odgovarjati na zahteve portala, ki so rezultat interakcij uporabnikov. Vmesnik mora znati vračati informacije o:

- razpoložljivih namestitvah;
- ceni rezervacije določene namestitve;
- načinu plačila;
- uspešnosti izvedene rezervacije.

Rezervacijo, izvršeno preko takega vmesnika, je možno enačiti z rezervacijo na recepciji hotela. Rezultat je ravno tako zabeležena rezervacija v rezervacijskem sistemu, s čimer dosežemo direktno odražanje sprememb v razpoložljivostih hotelskih kapacitet. Prav tako ni potrebe po usklajevanju podatkov med spletnim distribucijskim sistemom in hotelskim rezervacijskim sistemom, ker se vsi podatki pridobijo direktno iz hotelskega rezervacijskega sistema.

Dodatna možnost takšne zasnove je uporaba principov umetne inteligence pri gradnji vmesnika. Tak »pametni vmesnik« lahko sam na podlagi podatkov o povpraševanju in ponudbi določa ceno namestitev in tako poveča prihodke hotela za razliko od distribucijskih sistemov, kjer je neposredna komunikacija z gostom onemogočena in ceno določa sistem v svojo korist.

2.3. Komunikacijski strežnik

Komunikacijski strežnik predstavlja jedro komunikacije med portalom in vmesniki. Prav tako predstavlja priklopno točko vmesnikov v sistem. Komunikacijski strežnik, vmesniki in portal sodelujejo po principu odjemalec-strežnik, pri čemer igrajo vmesniki in portal vlogo odjemalca.

Osnovna naloga komunikacijskega strežnika je posredovanje sporočil med portalom in vmesnikom. Vsaka zahteva uporabnika na spletnem portalu predstavlja svojo nit, ki teče v okviru spletnega strežnika. Tako imamo v primeru več hkratnih zahtev različnih uporabnikov v okviru portala več servirnih niti. Vsaka taka nit povprašuje vmesnike. V primeru, ko nit želi pridobiti podatke o namestitvenih enotah nekega hotela, vzpostavi povezavo s komunikacijskim strežnikom in mu pošlje sporočilo z informacijo, kateremu vmesniku je to sporočilo namenjeno. Strežnik to sporočilo nato pošlje vmesniku, ki je stalno povezan s strežnikom. Odgovor poteka po enakem postopku v obratni smeri. Ena zahteva uporabnika lahko pomeni več poizvedb z enim vmesnikom ali z več različnimi vmesniki. Posamezna nit spletnega portala je aktivna le za čas, ko obdeluje zahtevo uporabnikovega brskalnika. Zato je le-ta povezana s komunikacijskim strežnikom le za čas poizvedovanja vmesnika. Vmesniki pa morajo biti stalno prisotni, saj je le tako možno povpraševanje s strani portala.

V določenih primerih pošiljajo tudi vmesniki zahteve za portal. Za take primere je na komunikacijski strežnik povezan tudi pomožni odjemalec, ki posredno predstavlja portal. Vmesniki lahko posodabljajo podatke hotela na portalu tako, da zahteve za posodobitev predajo pomožnemu odjemalcu.

Takšen način komuniciranja prinaša določene prednosti. Avtentikacija poteka le med komunikacijskim strežnikom in vmesnikom. To se zgodi le, ko vmesnik vzpostavi povezavo. Vmesnik nato ostane povezan in v čakanju na morebitne zahteve po informacijah s strani portala. Druga prednost je vzpostavljanje samo ene povezave v primeru povpraševanja več različnih vmesnikov. Prav tako portalu ni potrebno preverjati identitete vmesnika, ker je za avtentikacijo že poskrbel komunikacijski strežnik. Vmesniki imajo zato enostavnejšo zgradbo, ker imajo zgradbo odjemalca. Brez komunikacijskega strežnika pa bi vmesniki morali igrati vlogo strežnika, katerega bi ob zahtevah uporabnikov neposredno naslavljale niti portala. Zgradba sistema ponuja tudi širitev rezervacijskega sistema na druge uporabniške vmesnike (npr. vmesnik za rezervacije preko SMS - *Short Message Service* sporočil) in vmesnike za interakcije z drugimi rezervacijskimi sistemi. Vklopitev take funkcionalnosti bi zahtevala le izgradnjo komponente, ki bi na enak način kot spletni portal pridobivala informacije od

vmesnikov hotelskih rezervacijskih sistemov in jih preoblikovala v ustrezno obliko (npr. SMS sporočilo).

Pomembno je, da se zavedamo, da poteka komunikacija med komunikacijskim strežnikom, portalom in vmesniki preko interneta, ki je dostopen vsakomur in ni območje varne komunikacije kot velja za lokalno omrežje. Zato je potrebno pri zasnovi protokola imeti v mislih ne le učinkovitost, ampak tudi varovanje podatkov, kot so osebni podatki uporabnikov in podatki o njihovih plačilnih sredstvih.

2.4. Plačilni servisi

Plačilni servisi ponujajo trgovcem storitve elektronskega sprejemanja plačil [12, 13]. Pri tem omogočajo kupcem različne načine plačil, od uporabe kreditnih kartic do nakazil med bančnimi računi. Tako sistemu trgovca ni potrebno skrbeti za postopke v zvezi s plačilom, ker za vse poskrbi plačilni servis. Poleg tega plačilni servisi nudijo kupcem dodatne storitve, med katerimi so tudi elektronske denarnice. Elektronske denarnice omogočajo kupcem, da pri plačilnem servisu hranijo določeno vsoto denarja, namenjeno elektronskemu poslovanju. Tako jim ni potrebno vedno vnašati vseh podatkov o kreditni kartici, ker zadostujeta že uporabniško ime in geslo elektronske denarnice. Najbolj priljubljen globalni plačilni servis danes je PayPal. V Sloveniji pa imamo tudi lastne ponudnike, kot so spletni plačilni servis Abanke, NLB Klik, Moneta (plačevanje z mobilnim telefonom) idr.

Integracija plačilnega servisa se od ponudnika do ponudnika razlikuje. Postopek načeloma poteka v treh korakih. Prvi je predaja kupca plačilnemu servisu skupaj s parametri, potrebnimi za izvršitev plačila (npr.: znesek, račun trgovca, url naslov za vrnitev kupca in url naslov, kamor plačilni servis sporoči status plačila). V primeru spletnega vmesnika plačilnega servisa se to stori s preusmeritvijo brskalnika na določen naslov vključno s potrebnimi prametri (z uporabo POST metode ali na kakšen drugačen način). V drugem koraku kupec opravi plačilo preko plačilnega servisa, ki ga nato preusmeri na povratni naslov, ki je bil servisu posredovan s parametri. V tretjem koraku kupec čaka na spletni strani trgovca, dokler plačilni servis ne sporoči uspešno opravljenega plačila. Ko plačilni servis opravi denarno transakcijo, to sporoči sistemu trgovca, tako da sproži zahtevo z določenimi prametri na naslov za potrditev plačila, ki je bil servisu posredovan s parametri ob predaji kupca.

V rezervacijski postopek se vključi plačilni servis takrat, ko vmesnik hotelskega rezervacijskega sistema zahteva plačilo polne ali delne cene namestitve. Plačilo vnaprej služi hotelu kot zagotovilo za prihod gosta oziroma kot nadomestilo za nastalo izgubo ob morebitnem ne prihodu, gosta. Ker gre pri tem za prenose denarja, je dobro, če poskrbimo za ponudbo večjega števila plačilnih servisov in s tem omogočimo uporabniku, da izvede plačilo preko servisa, ki mu sam najbolj zaupa. Seveda pa moramo pri izbiri ponudnikov plačilnih servisov poskrbeti, da integriramo tiste, ki so zanesljivi in imajo ustrezno poskrbljeno za varnost. Le tako bodo lahko uporabniki hitro, enostavno in brezskrbno opravili plačilo.

3. Varnost v elektronskih komunikacijah

Elektronske komunikacije igrajo ključno vlogo pri gradnji distribuiranih sistemov, kjer omogočajo interakcijo med posameznimi komponentami, ki se nahajajo na fizično ločenih lokacijah. Uporabniki ne uporabljajo sistemov, ki jim ne zaupajo. Zato je ključnega pomena za pridobitev zaupanja v sistem preprečevanje varnostnih groženj, ki se pojavijo pri komuniciranju med komponentami sistema preko odprtih sistemov, kot je internet. Kriptografija je znanstvena veda, ki ponuja tehnike in metode za varno hranjenje in prenašanje podatkov v informacijskih in komunikacijskih sistemih. Kriptografske tehnike in metode so danes nujno potrebne v elektronskem poslovanju, saj omogočajo varnost in zanesljivost poslovnih transakcij.

Kot varnostne grožnje v komuniciranju med legalnimi uporabniki razumemo poseganje napadalcev v komunikacijo na naslednje načine [1]:

- *prekinitev* – napadalec povzroči neuporabnost enega ali več sistemov;
- *prisluškovanje* – napadalec zbira zaupne podatke s prisluškovanjem;
- *ponarejanje* – napadalec prestreza sporočila in jih nato preoblikuje v svoj prid ter posreduje naprej;
- *kraja identitete* – napadalec se izdaja za nekoga od legalnih uporabnikov.

Kriptografija nudi tehnike in metode za preoblikovanje sporočil v obliko, ki omogoča [1, 6]:

- *tajnost sporočil* – vsebina sporočil je skrita pred nezaželenimi osebami;
- *integriteto sporočil* – prejemnik sporočil ima dokaz, da sporočilo med prenosom ni bilo preoblikovano;
- *avtentikacijo uporabnikov* – pošiljatelj in prejemnik se lahko nedvoumno identificirata;
- *ne-zanikanje* – prejemnik in pošiljatelj ne moreta zanikati izmenjave sporočil.

Kriptografske algoritme za varovanje sporočil s šifriranjem v osnovi razdelimo na simetrične in asimetrične glede na to, ali uporabljajo za šifriranje in dešifriranje isti ključ ali dva različna ključa. Poleg algoritmov za šifriranje se uporablja še cela množica pomožnih postopkov, ki so nujno potrebni za učinkovito izvedbo kriptografskih sistemov. V kriptografiji velja Kerchoff-ov princip, ki pravi, da mora biti kriptografski sistem varen kljub temu, da ima napadalec vse informacije o sistemu [6]. To pomeni, da je potrebno skrivati le ključ, ne pa tudi algoritma. Razkritje podrobnosti zagotavlja implementacijo varnih algoritmov, saj so tako lahko podrobno proučeni s strani strokovnjakov.

3.1. Kriptografske tehnike

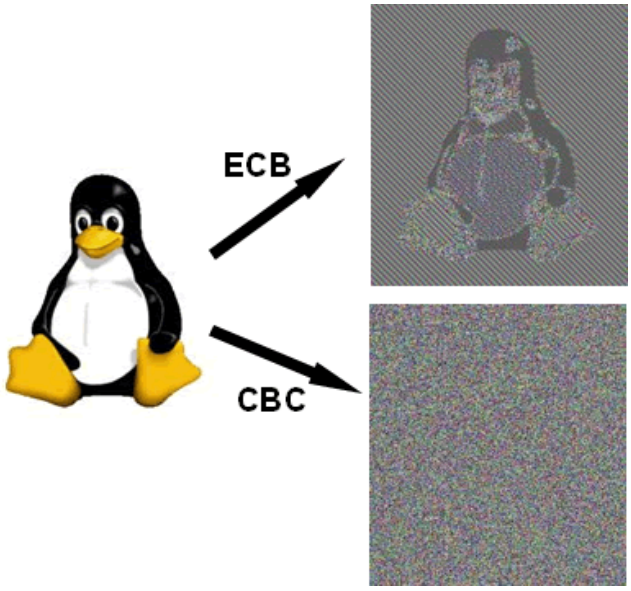
3.1.1. *Simetrična kriptografija*

Simetrična kriptografija obsega algoritme, pri katerih se za šifriranje in dešifriranje podatkov uporablja en sam ključ. Ker algoritmi uporabljajo isti ključ za obe operaciji, pravimo, da so simetrični. Prvi problem simetrične kriptografije je, da rabimo različen ključ za vsak par uporabnikov, za katerega želimo zagotoviti tajno izmenjavo informacij. Tako za varno komunikacijo med n uporabniki potrebujemo n^2 ključev. Drugi problem je izmenjava ključa med uporabnikoma. Ključ mora ostati tajen, saj v nasprotnem primeru lahko dešifrira sporočilo vsak, ki pridobi ključ. Izmenjavo skrivnosti oziroma ključa uspešno rešujejo algoritmi za asimetrično šifriranje. Ne glede na omenjeni slabosti so simetrični algoritmi

ključnega pomena v komunikacijah, ker omogočajo učinkovito in hitro implementacijo postopkov šifriranja in dešifriranja.

Simetrične algoritme delimo na tiste, ki vršijo tekoče šifriranje (*stream cipher*) in tiste, ki vršijo blokovno šifriranje (*block cipher*). Algoritmi za tekoče šifriranje na podlagi tajnega ključa in podatkov sproti generirajo neskončen niz bitov, ki predstavlja ključ, s katerim se nato z binarno operacijo XOR (ekskluzivni ali) šifrira sporočilo. Neskončno dolg ključ se generira sproti in se tako uporabi samo za eno šifriranje in dešifriranje (seveda se ključ po enakem postopku generira tudi na prejemnikovi strani, drugače bi bila operacija enosmerna). Ideja izhaja iz metode enkratnega ključa (*One-Time Pad*) [6]. Algoritmi za tekoče šifriranje so bili v osnovi namenjeni strojni realizaciji in so zaradi svoje hitrosti prevladovali v preteklosti. Danes jih počasi izpodrivajo programske rešitve z blokovni algoritmi in se pojavljajo le še na redkih področjih, kjer ne zadoščajo programske rešitve. Primer sta algoritma A5/1 in RC4. A5/1 se uporablja v GSM (*Global System for Mobile communications*) telefoniji in je namenjen strojni realizaciji in generira bitni ključ. RC4 pa je že namenjen programski realizaciji in je zato bajt enota za generianje ključa. Algoritem A5/1 je primer kršitve Kerchoff-ovega principa, saj je bil zamolčan kot skrivnost GSM komunikacij. Vendar je kmalu prišel v javnost s pomočjo povratnega inženiringa, pri čemer je bilo odkritih veliko slabosti šifriranega algoritma.

Danes prevladujejo algoritmi za blokovno šifriranje. V večini so realizirani programsko in so zasnovani tako, da zagotovijo čim večjo varnost in učinkovitost. Šifriranje poteka tako, da se sporočilo najprej razdeli na bloke fiksnih dolžin. Potem se izvede postopek šifriranja nad vsakim blokom posebej. Tak direktni način blokovnega šifriranja se imenuje ECB (*Electronic Code Book*) ali elektronska kodirna knjiga, ker se blok z določeno sekvenco bitov vedno preslika na enak način. Problem metode ECB je, da omogočimo napadalcu spreminjanje zaporedja blokov ali celo razkritje podatkov. Za preprečevanje slednjega problema, ki je prikazan na sliki 2, se uporabljajo metode blokovnega šifriranja kot je CBC (*Chiper Block Chaining*), ki pri šifriranju posameznih blokov vključujejo informacijo iz sosednjih blokov [3, 6]. Propagiranje informacij iz sosednjih blokov pomeni tudi propagiranje napak v blokih. Tako napaka v enem bitu lahko pomeni ovržbo vsaj dveh blokov. Zato je tekoče šifriranje bolj primerno kot blokovno za uporabo v okoljih, kjer pogosto prihaja do napak pri prenosih.



Slika 2: Problem blokovnega šifriranja podatkov

Danes obstaja več algoritmov za blokovno šifriranje in se med seboj razlikujejo v lastnostih, kot so dolžina ključa, bloka, stopnji varnosti in hitrosti. Primeri najbolj pogosto uporabljenih algoritmov za blokovno šifriranje danes so DES, 3DES, AES, IDEA in CAST-128.

DES (*Data Encryption Standard*) je najbolj pogosto uporabljen šifrirni algoritem v praksi. Začetki algoritma segajo v 70. leta 20. stol., ko v Združenih državah Amerike NBS (*National Bureau of Standards*) sproži zahtevo po razvoju standarda za šifriranje zaupnih podatkov za uporabo v industriji in vladne namene. Kot standard je bil sprejet IBM-ov algoritem DES z bloki dolžin 64 bitov in ključem dolžine 56 bitov. Algoritem je bil pred standardizacijo nekoliko spremenjen s strani ameriške vlade, zato se sumi, da obstaja bližnjica za razbijanje. Vendar ne glede na to, ali bližnjica obstaja ali ne se danes algoritem ne smatra več za varnega zaradi prekratke dolžine ključa, ki pri današnjih zmogljivostih računalnikov omogoča smiselno izčrpno iskanje ključa. Izčrpno iskanje ključa je osnoven način za napad na vsak kriptografski algoritem pri čemer čas iskanja narekuje dolžina ključa. Danes se za varno dolžino ključa smatra 128 bitov, čas, potreben za razbitje DES algoritma s 56-bitnim ključem, se ocenjuje na od nekaj dni do nekaj ur, odvisno od opreme, ki je napadalcu na voljo. Poleg izčrpnega iskanja se za napad na DES uporabljajo tudi postopki diferencialne in linearne kriptoanalize, ki v primerjavi z izčrpnim iskanjem ključa bistveno zmanjšata čas iskanja [6].

3DES (*Triple DES*) ali trojni DES se uporablja kot postopek za preprečevanje šibkosti DES algoritma. Pri 3DES pošljemo sporočilo trikrat skozi DES algoritem po naslednjem postopku:

1. šifriraj sporočilo z DES s ključem K1;
2. izhod iz 1. točke dešifriraj s ključem K2;
3. izhod iz 2. točke šifriraj s ključem K1.

Postopek je trikrat počasnejši od prvotnega DES algoritma, vendar je zaradi 112-bitnega ključa (ključa K1 + K2) dvakrat varnejši. Pri čemer sekvenca šifriraj-dešifriraj-šifriraj omogoča združljivost z DES algoritmom ob uporabi istega ključa za K1 in K2.

AES (*Advanced Encryption Standard*) je algoritem, ki je bil razvit kot nadomestilo za DES. Ker je postala šibkost DES vse bolj jasna, je v Združenih državah Amerike NIST (*National Institute of Standards and Technology*, poprej znan kot NBS) izrazil zahtevo po novem, kvalitetnejšem šifrirnem standardu. Med tekmovalnimi algoritmi je zmagal algoritem Rijndeal, katerega avtor je bil Adi Shamir. AES odlikuje dobro definirano matematično ozadje ter hitrost in je eden najbolje proučenih kriptografskih algoritmov, ki je trenutno brez znanih varnostnih pomanjkljivosti. AES je postal standard leta 2002 in je danes med najbolj priljubljenimi algoritmi za simetrično šifriranje. AES podpira bloke in ključe dolžin 128, 192 in 256 bitov. Za šifriranje zaupnih podatkov so ustrezne vse tri dolžine ključa. Za strogo zaupne pa zadostujeta le 192 in 256-bitni dolžini.

3.1.2. *Asimetrična kriptografija*

V asimetrični kriptografiji ali kriptografiji z javnim ključem se uporabljata dva ključa, eden za šifriranje in drugi za dešifriranje. Tak par ključev delimo na javni in privatni ključ, pri čemer za par ključev velja, da kar z enim od ključev zašifriramo, lahko le z drugim dešifriramo. Javni ključ je dostopen vsem. Vsakdo, ki želi poslati sporočilo določenemu uporabniku in želi, da ga bo lahko prebral le ta uporabnik, vzame njegov javni ključ in z njim šifrira sporočilo. Tako šifrirano sporočilo lahko dešifriramo le s pripadajočim privatnim ključem, ki ga pozna le lastnik. Privatni ključ je potrebno skrbno varovati, kajti le tako lahko preprečimo dostop napadalcem do zaupnih informacij. Ker je javni ključ dostopen vsem, ni

potrebe po izmenjavi ključa. Prav tako zahteva tak sistem varne izmenjave sporočil manjše število ključev, saj za n uporabnikov potrebujemo le $2 \cdot n$ ključev, kar je bistveno manj kot pri simetrični kriptografiji. Poleg šifriranja kriptografija z javnim ključem omogoča tudi postopek digitalnega podpisa. Če pošiljatelj zašifrira sporočilo s svojim privatnim ključem, preden ga pošlje, bo prejemnik lahko preveril, da je sporočilo res poslal pošiljatelj, ker bo lahko dešifriral sporočilo le z javnim ključem, ki pripada pošiljatelju. Bistvo postopkov šifriranja in podpisovanja v okviru asimetrične kriptografije je, da lahko vsakdo šifrira in preverja podpis, medtem ko dešifrira in podpisuje lahko le lastnik privatnega ključa.

RSA (*Rivest Shamir Adleman*) je najbolj razširjen algoritem, ki implementira postopek kriptografije z javnim ključem in je postal *de facto* standard za šifriranje podatkov v industriji, še posebej pri prenosu sporočil preko interneta. RSA je ključna komponenta v elektronskem poslovanju, ki zagotavlja visoko stopnjo varnosti poslovnih transakcij. Algoritem so avtorji Ron Rivest, Adi Shamir in Leonard Adleman prvič javno objavili leta 1977 na MIT (*Massachusetts Institute of Technology*). Algoritem zagotavlja princip kriptografije z javnim ključem z uporabo modularne aritmetike in lastnosti velikih praštevil [6]. Javni ključ uporabnika predstavljata šifrirni eksponent e in modul n. Privatni ključ pa eksponent za dešifriranje d. Ključa pridobimo tako, da najprej izberemo dve veliki praštevili p in q. Nato izračunamo števili n in f, kot prikazujeta enačbi (1) in (2). Eksponenta e in d morata ustrezati pogojema (3) in (4). Postopek šifriranja in dešifriranja prikazujeta enačbi (5) in (6) pri čemer je M sporočilo, C pa šifrirano sporočilo.

$$n = p \times q \qquad (1)$$

$$f = (p-1) \times (q-1) \qquad (2)$$

$$1 < e < f \ \wedge \ (f \text{ in } e \text{ tuji števili}) \qquad (3)$$

$$d \times e \equiv 1 \bmod f \qquad (4)$$

$$C = M^e \bmod n \qquad (5)$$

$$M = C^d \bmod n \qquad (6)$$

Napad z izčrpnim iskanjem ključa je pri uporabi ključev dolžin 1024 – 4096 nesmiseln in je zato varen način šifriranja podatkov. Največji problem, ki se pojavi pri RSA algoritmu, je, ko uporabnik zanika lastništvo ključev in s tem zanika hkrati svoj digitalni podpis. Ta problem se rešuje z vpeljavo digitalnih certifikatov in certifikatnih agencij, ki vežejo uporabnika in njemu pripadajoč par ključev. Slabost RSA algoritma je v računski zahtevnosti operacij, ki so potrebne pri realizaciji šifriranja in dešifriranja, zaradi česar algoritem ni primeren za uporabo v okoljih, kjer so potrebe po hitrem prenosu dolgih sporočil.

3.1.3. *Kombiniranje simetričnega in asimetričnega šifriranja*

Slabost RSA algoritma je počasnost, saj se uporabljajo dolgi ključi in množenje velikih števil. Na drugi strani pa imamo šifriranje s simetričnim ključem, kjer imamo problem izmenjave ključa, vendar hitre postopke šifriranja s kratkimi ključi. Če združimo dobre lastnosti obeh kriptografskih postopkov, lahko premostimo njune slabosti. To storimo tako, da za izmenjavo ključa za simetrično šifriranje uporabimo počasno asimetrično šifriranje, kjer ni problema z izmenjavo ključev. Nato uporabimo izmenjani ključ za hitro šifriranje podatkov s simetričnimi algoritmi. Počasni asimetrični algoritmi se tako uporabijo le pri vzpostavitvi povezave in poskrbijo, da se vzpostavijo pogoji, ki omogočijo varno uporabo hitrih simetričnih algoritmov.

Tak način šifriranja se v praksi pogosto uporablja. Primer je SSL (*Secure Sockets Layer*) protokol, ki se najbolj pogosto uporablja za zaščito komunikacije med spletnim strežnikom in uporabnikovim brskalnikom. SSL v začetni fazi (*SSL handshake*) vzpostavi med odjemalcem in strežnikom tako imenovani sejni ključ z uporabo kriptografije z javnim ključem. Sejni ključ nato služi kot ključ za simetrično šifriranje sporočil, ki se izmenjujejo med strežnikom in brskalnikom.

3.1.4. *Ostale kriptografske tehnike*

Poleg algoritmov za šifriranje podatkov igrajo pomembno vlogo pri zagotavljanju varnega prenašanja sporočil tudi zgoščevalne funkcije, DH (*Diffie-Hellman*) izmenjava ključa in generatorji naključnih števil.

RSA algoritem ni edini način za izmenjavo tajnega ključa za simetrično šifriranje podatkov. Algoritem DH je postopek, ki omogoča, da uporabnika, ki želita komunicirati, ustvarita ključ ne da bi si ga pošiljala med seboj. Varnost postopka temelji na težavnosti rešitve diskretnega logaritma, sam dogovor pa poteka na naslednji način:

1. imamo *praštevilo p* in število *g*, ki mu pravimo *generator* (števili nista skrivnost in sta lahko javno znani);
2. pošiljatelj najprej zgenerira vrednost *a* in pošlje prejemniku vrednost *x* (7);
3. po prejemu pošiljatelj ravno tako zgenerira lastno vrednost *b* in odgovori pošiljatelju z vrednostjo *y* (8);
4. nato lahko oba izračunata ključ *k* (9) brez skrbi, da bi do njega lahko prišel tudi prisluškovalec, ker mora le-ta priti do enega od števil *a* in *b* (10), za kar pa je potrebno rešiti diskretni logaritem [6].

$$x = g^a \bmod p \qquad (7)$$

$$y = g^b \bmod p \qquad (8)$$

$$k = x^b = y^a = (g^a)^b \bmod p = (g^b)^a \bmod p = g^{a \cdot b} \bmod p \qquad (9)$$

$$g^a \cdot g^b = g^{a+b} \neq g^{a \cdot b} \bmod p \qquad (10)$$

Tako vzpostavljen ključ se lahko nato uporabi za šifriranje podatkov s simetričnimi algoritmi. Edini problem postopka je podvrženost "napadu moža v sredini" (*man in the middle attack*). Napadalec lahko prestreže izmenjavo števil *x* in *y* in s posredovanjem lastnih števil vsakemu od uporabnikov spelje promet preko sebe. Za preprečevanje napada imamo tri rešitve, in sicer lahko DH postopek varno izvedemo [6]:

- s šifriranjem z dogovorjenim skrivnim ključem;
- s šifriranjem z javnim ključem;
- s podpisovanjem s privatnim ključem.

Na prvi pogled je DH izmenjava nesmiselna v primeru, ko imamo katerega od naštetih načinov že vzpostavljenega. Vendar omogoča pri sejnih ključih vzpostaviti lastnost PFS (*Perfect Forward Secrecy*). PFS zagotavlja, da sejnega ključa ni mogoče odkriti, čeprav je eden od parov ključev za simetrično kriptografijo oz. ključ za simetrično kriptografijo odkrit [6].

Kriptografske zgoščevalne (*hash*) funkcije iz poljubno dolgega niza ustvarijo niz fiksne dolžine. Zgoščevalne funkcije uporabljamo pri ustvarjanju digitalnih podpisov in zagotavljanju integritete sporočil. Tako pri podpisovanju ni potrebe po opravljanju zamudnega asimetričnega šifriranja nad dolgim sporočilom, ampak lahko to opravimo le nad

kratkim nizom fiksne dolžine, ki je rezultat zgoščevalne funkcije sporočila. Kriptografske zgoščevalne funkcije za razliko od standardnih, ki so namenjene zgolj detekciji napak pri prenosu podatkov, zagotavljajo učinkovitost in zelo majhno verjetnost trka pri iskanju nizov z enako vrednostjo funkcije. Zaželena lastnost zgoščevalnih funkcij (in tudi šifrirnih algoritmov) je tako imenovani "*avalanche effect*", kjer sprememba v enem ali več bitov na vhodu povzroči spremembo v vsaj polovici bitov na izhodu. Najbolj pogosto uporabljene zgoščevalne funkcije danes so SHA-1, MD5 in RIPEMD-160. MD5 se ne smatra več za varno, ker so bili že odkriti trki.

Kriptografska naključna števila se pogosto uporabljajo pri generiranju ključev in ostalih naključnih vrednosti, kot so eksponenti pri DH izmenjavi ključa. Od navadnih naključnih števil se razlikujejo v tem, da poleg zahteve po statistični neodvisnosti izpolnjujejo tudi pogoj o nepredvidljivosti. Nepredvidljivost je velik problem, ker je računalnik deterministični stroj. Kljub temu pa lahko najdemo določene izvire naključnosti kot so premiki miške, pritiski tipk tipkovnice, promet na omrežju idr. Naključnost je pogosto spregledana varnostna luknja. Če sistem, ki na primer za uporabnike ustvarja pare javnih in privatnih ključev, pri tem uporablja predvidljiv generator naključnih števil, lahko trije uporabniki na podlagi svojih ključev ugotovijo privatni ključ četrtega uporabnika, kar lahko predstavlja veliko varnostno grožnjo.

3.2. Uporaba kriptografskih tehnik

3.2.1. *Šifriranje sporočil*

S šifriranjem sporočil zagotavljamo zaupnost sporočil. Sporočila lahko šifriramo z uporabo simetrične in asimetrične kriptografije. Postopka sta naslednja:

Simetrična kriptografija	Asimetrična kriptografija
Pošiljatelj pred pošiljanjem sporočilo zašifrira z dogovorjenim skrivnim ključem.	Pošiljatelj zašifrira sporočilo s prejemnikovim javnim ključem.
Prejemnik ob sprejetju sporočilo dešifrira z istim dogovorjenim skrivnim ključem.	Prejemnik dešifrira sporočilo z lastnim privatnim ključem.

3.2.2. *Integriteta sporočil*

Integriteto sporočila zagotavljamo z uporabo metod za izračun "prstnega odtisa" (*fingerprint*) sporočila:

- *zgoščevalne funkcije*. Da omogočimo prejemniku verifikacijo integritete sporočila, najprej izračunamo rezultat njegove zgoščevalne funkcije. Rezultat, ki predstavlja prstni odtis, nato zašifriramo in ga skupaj s sporočilom pošljemo prejemniku. Prejemnik nato ob prejetju sporočila in zašifrirane vrednosti zgoščevalne funkcije sam izračuna vrednost zgoščevalne funkcije nad sporočilom in jo primerja s sprejeto vrednostjo, ki jo mora seveda najprej dešifrirati.
- *MAC*. Pri uporabi MAC (*Message Authentication Code*) je postopek podoben kot pri uporabi zgoščevalnih funkcij, le da tu uporabimo za izračun »prstnega odtisa« enega od blokovnih algoritmov za simetrično šifriranje. MAC izračunamo tako, da pri šifriranju sporočila v CBC načinu zavržemo vse bloke razen zadnjega, ki ga uporabimo za prstni odtis sporočila. To je možno, ker pri CBC načinu zašifriramo vsak blok tako, da nad blokom najprej izvedemo XOR operacijo z rezultatom šifriranja prejšnjega bloka, kar povzroči, da se informacije vseh blokov vključijo v zadnji blok.

MAC-a ni potrebno zašifrirati, ker je že sam po sebi rezultat šifriranja. Postopek preverjanja integritete sporočila je podoben uporabi zgoščevalnih funkcij.

- *HMAC*. HMAC (*Hash Message Authentication Code*) je podoben postopku uporabi MAC-a, le da tu izračunamo prstni odtis sporočila tako, da najprej z uporabo binarnih operacij zamešamo ključ za simetrično šifriranje s sporočilom in nato nad rezultatom izračunamo vrednost zgoščevalne funkcije [6].

3.2.3. *Digitalni podpis*

Digitalni podpis je možno realizirati le z uporabo asimetrične kriptografije. Pošiljatelj digitalno podpiše sporočilo, tako da izračuna vrednost zgoščevalne funkcije sporočila in jo nato zašifrira s svojim privatnim ključem. Digitalni podpis nato pošlje skupaj s sporočilom. Prejemnik nato preveri digitalni podpis in se s tem prepriča, da je sporočilo res poslal pošiljatelj. To storit tako, da sam izračuna vrednost zgoščevalne funkcije prejetega sporočila in jo primerja z vrednostjo, ki jo dobi z dešifriranjem digitalnega podpisa s pošiljateljevim javnim ključem.

3.2.4. *Avtentikacija uporabnikov*

Uporabnika, ki si izmenjujeta sporočila varovana z uporabo simetrične kriptografije, se v primeru, da je ključ, ki ga pri tem uporabljata, poznan le njima, med seboj z uporabo skupne skrivnosti – ključa avtomatsko avtenticirata. Pri asimetrični kriptografiji to ni mogoče, ker nimamo vzpostavljene skupne skrivnosti med uporabnikoma in morata zato drug drugemu dokazati svojo identiteto z uporabo digitalnega podpisa in digitalnega certifikata, ki zagotovita vez med ključem in identiteto lastnika.

Glede na to, ali dokažeta svojo identiteto oba uporabnika ali samo eden, ločimo vzajemno in enosmerno avtentikacijo. Pri enosmerni avtentikaciji se predstavi samo eden od uporabnikov. Primer enosmerne avtentikacije so spletne trgovine, kjer se predstavi z digitalnim certifikatom le spletni strežnik. Za ponudnika take storitve je striktna identiteta kupca nepomembna, važno je samo, da kupec izvede veljavno plačilo kupljenega blaga. Kupcu pa je avtentikacija strežnika pomembna, ker mora biti prepričan, da res plačuje trgovcu in ne komu drugemu. Kadar se prepričata o identiteti ena druge obe strani, pravimo, da sta opravili vzajemno avtentikacijo. Primer vzajemne avtentikacije srečamo pri spletnem bančništvu, kjer se z digitalnim certifikatom predstavita tako spletni strežnik banke kot tudi uporabnikov brskalnik. S tem banka dobi zagotovilo, da z bančnim računom opravlja njegov lastnik. Uporabnik pa je lahko prepričan, da je res na pravi spletni strani banke.

Najbolj pogosta napada na slabo zasnovane avtentikacijske protokole sta "napad s ponovitvijo" (*replay attack*) in "napad moža v sredini". Za preprečevanje napada moža v sredini se uporabljajo kriptografske tehnike, ki preprečujejo ponarejanje in prisluškovanje. Problem napada s ponovitvijo se pojavi, ko uporabnik, ki hoče dokazati svojo identiteto, pošlje šifrirane podatke o svoji identiteti drugemu uporabniku. Če napadalec posname tako sporočilo, se lahko s kasnejšim posredovanjem tega sporočila prejemniku predstavi kot pošiljatelj, kljub temu da sporočila ne more dešifrirati. Za preprečevanje napada s ponovitvijo se skupaj s sporočilom vedno šifrirajo še neke dodatne informacije, ki so vedno drugačne, odraz česar je vedno drugačen rezultat šifriranja sporočila. Za take dodatne informacije se uporabljajo sejni žetoni, enkrat uporabljena števila (*nonce – number used once*) in časovni žigi (*time stamp*).

3.3. Infrastruktura javnih ključev

Kriptografija z javnim ključem sama po sebi ne zagotavlja zadostne varnosti v elektronskem poslovanju. Zato potrebujemo dodatno ogrodje, ki bo poskrbelo za varno generiranje in izmenjavo ključev ter vzpostavilo zaupanja vredno vez med posameznim uporabnikom in pripadajočim parom ključev. Tako ogrodje je infrastruktura javnih ključev oz. PKI (*Public Key Infrastructure*). PKI je sestavljena iz varnostnih politik, kriptografskih tehnik, aplikacij in procedur za generiranje, hranjenje in upravljanje ključev ter digitalnih certifikatov. Zaupanje med uporabniki je lahko vzpostavljeno le v primeru, ko so lahko prepričani o medsebojni identiteti in varnosti informacij, ki si jih izmenjujejo. PKI rešuje probleme zaupanja, avtentikacije in varnosti ter s tem vzpostavlja zaupanje in varnost fizičnega sveta v elektronskem svetu.

Poleg osnovnih funkcij tajnosti sporočil, integritete sporočil, avtentikacije uporabnikov in ne-zanikanja, ki jih ponuja kriptografija, za varno in zaupanja vredno elektronsko interakcijo potrebujemo še [1]:

- varnostne politike, ki specificirajo pravila za uporabo kriptografskih tehnik;
- mehanizme za upravljanje, hranjenje in ustvarjanje ključev;
- napotke za upravljanje, hranjenje, distribuiranje in ustvarjanje ključev in digitalnih certifikatov.

Za integracijo vsega potrebnega PKI gradijo različne komponente, izmed katerih vsaka z opravljanjem svojih nalog prispeva k varni uporabi kriptografije z javnim ključem.

3.3.1. *Certifikatne agencije*

Osnovne naloge certifikatnih agencij (CA – *Certification Authority*) so izdajanje in overjanje digitalnih certifikatov. CA je poznana kot zaupanja vredna tretja oseba (TTP – *Trusted Third Party*), ki z lastnim digitalnim podpisom overi digitalni certifikat in vzpostavi zaupanja vredno vez med identiteto uporabnika in njegovim parom ključev za asimetrično šifriranje. Uporabnika digitalnih certifikatov se med seboj lahko varno avtenticirata in izmenjata informacije, če imata digitalni certifikat, ki ga je podpisala CA, ki ji oba zaupata. Glede na to, kako s CA vzpostavimo zaupanje med uporabniki, poznamo različne modele zaupanja. V monopolnem modelu imamo eno samo CA, ki ji vsi zaupajo. Problem v takem modelu je, da bi ob ogrozitvi sistema CA prišlo do izpada zaupanja med uporabniki. Druga možnost je oligarhija, kjer imamo več CA, ki jim zaupamo. Tako v primeru izpada ene CA uporabniki še vedno lahko vzpostavijo zaupanje preko preostalih CA. Na tak model zaupanja naletimo danes pri uporabi spletnih brskalnikov, ki imajo na seznamih zaupanja vrednih lahko preko petdeset CA. Tretja možnost vzpostavitve je model zaupanja, kjer lahko vsak uporabnik igra vlogo CA. Primer slednjega je sistem PGP (*Pretty Good Privacy*), ki je namenjen varni izmenjavi elektronske pošte [1]. Digitalni certifikat, s katerim se CA predstavlja, ponavadi CA overi in podpiše sama sebi. Pri vzpostavljanju oligarhičnih modelov zaupanja pa naletimo tudi na pojav, ko si več CA overi in podpiše digitalne certifikate med seboj in si s tem vzpostavijo zaupanje med klienti.

CA izda nov digitalni certifikat le v primeru, ko od prijavne službe prejme zahtevo za izdajo novega certifikata. Pri tem CA v obliki certifikata združi podatke o uporabniku in javni ključ. Ali par ključev ustvari CA ali uporabnik je odvisno od certifikatne politike, ki jo prakticira posamezna CA. S certifikatno politiko CA navaja pravila in obveze, s katerimi zagotavlja varnost pri upravljanju s podatki uporabnikov in njihovimi certifikati. Certifikatna politika CA je tudi dokument, na podlag katerega se lahko odločimo, ali zaupamo določeni

CA ali ne. Poleg izdajanja digitalnih certifikatov mora CA izvajati tudi funkcijo preklica obstoječih certifikatov. CA prekliče certifikat v naslednjih primerih [1]:

- podatki, objavljeni na certifikatu niso več veljavni;
- pripadajoč privatni ključ je bil izgubljen in je zato potrebna izdaja novega para ključev;
- pripadajoč privatni ključ je bil izpostavljen varnostni grožnji in je zato potrebna menjava para ključev;
- v primeru menjave certifikata CA zaradi katerega od prej navedenih razlogov.

3.3.2. *Prijavne službe*

Prijavna služba (*Registration Authority*) je odgovorna za interakcijo med klienti in CA. Prijavne službe opravljajo sprejemanje zahtev za izdajo digitalnih certifikatov. Pri tem preverijo zahtevo in identiteto klienta, ki je podal zahtevo. Ko je preverjanje zaključeno, zahtevo za izdelavo certifikata posredujejo CA. Ko je certifikat izdelan, ga CA posreduje nazaj prijavni službi, ki nato poskrbi zato, da je varno predan pravi osebi. Prijavne službe imajo posebno vlogo pri širitvi PKI, saj je lahko vsaki geografski regiji dodeljena svoja prijavna služba.

3.3.3. *Digitalni certifikati*

Digitalni certifikat je elektronski dokument, ki zagotavlja integriteto javnega ključa in veže javni ključ in z njim povezane informacije z lastnikom para ključev za asimetrično kriptografijo na zaupanja vreden način. S tem preprečuje varnostne grožnje, povezane z modifikacijo ključa, krajo identitete in zanikanjem lastništva uporabljenih ključev. Tako je poskrbljeno, da samo javni ključ, ki ga je CA overila s certifikatom, deluje s privatnim ključem pravega lastnika certifikata. Na digitalnem certifikatu najdemo poleg podatkov o certifikatu tudi podatke o uporabniku. Podatki o uporabniku so lahko poljubno podrobni, vendar bolj ko so podrobni, večja je verjetnost, da bo zaradi spremembe določenega podatka postal certifikat neveljaven.

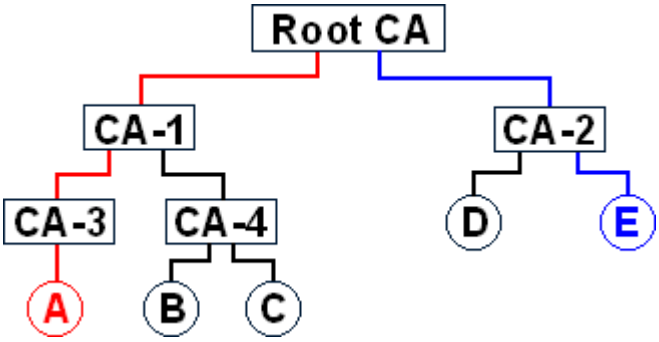
Leta 1988 je International Telecommunications Union, agencija za informacijske in komunikacijske tehnologije, predstavila X.509 standard za strukturo digitalnih certifikatov. Od takrat je X.509 postal *de facto* standard v industriji za avtentikacijo uporabnikov na odprtih sistemih, kot je internet. Trenutna je tretja verzija standarda, ki predvideva naslednjo strukturo digitalnega certifikata [1]:

- *verzija certifikata*: opisuje verzijo X.509 standarda po kateri je certifikat strukturiran.
- *serijska številka certifikata*: to je enolični celoštevilski identifikator, ki ga uporablja CA za identifikacijo certifikata.
- *algoritem CA-jinega podpisa*: identificira algoritem, s katerim je CA podpisala certifikat.
- *ime CA*: identificira CA, ki je podpisala in izdala certifikat.
- *obdobje veljavnosti*: vsebuje datuma pričetka in konca veljavnosti certifikata.
- *ime subjekta*: identificira lastnika privatnega ključa, ki pripada javnemu ključu, ki je del certifikata.
- *informacije o javnem ključu subjekta*: javni ključ in podatki o algoritmu, s katerim se sme uporabljati.
- *enolični identifikator subjekta*: ta atribut vsebuje enolični identifikator lastnika certifikata pri CA, ki je izdala certifikat.

- *enolični identifikator CA*: ta atribut vsebuje enolični identifikator CA in poleg atributa, ki vsebuje ime CA, nudi mehanizem dodatne verifikacije, da je certifikat resnično izdala določena CA.
- *razširitve*: neobvezen sklop atributov, ki jih vpelje tretja verzija standarda. Razširitve predstavljajo dodatna polja, ki jih lahko vsak definira in vključi v certifikat. Danes najbolj pogosti razširitvi sta *uporaba ključa*, ki definira omejitve uporab ključa, in *alternativno ime subjekta*, ki predstavlja dodatno ime lastnika ključa, kot je elektronski naslov ali naslov domače strani. Posebno pozornost si zaslužijo razširitve, ki so označene kot *kritične*. Če imamo na primer razširitev uporaba ključa, ki ima navedeno omejitev zgolj za podpisovanje certifikatov, in je označena kot kritična, moramo certifikat v primeru šifriranja in podpisovanja podatkov zavrniti.

Poleg strukture digitalnega certifikata X.509 predpisuje tudi algoritem za iskanje certifikacijske poti, ki je potrebna pri preverjanju veljavnosti certifikata. Certifikacijska pot je seznam vseh digitalnih certifikatov, kjer je prvi v seznamu certifikat CA, ki ji uporabnik zaupa, in zadnji certifikat, ki ga preverjamo. Med njima je lahko več certifikatov drugih CA, ki so z medsebojnim podpisovanjem vzpostavile zaupanje. Bolj podrobno so različni modeli zaupanj in iskanje certifikacijkih poti opisani v [1], primer hierarhičnega modela zaupanja pa prikazuje slika 3, kjer sta certifikacijski poti za certifikata uporabnikov A in E, ki želita eden drugega preveriti, naslednji:

- za uporabnika A: [Root → CA-1] : [CA-1 → CA-3] : [CA-3 → A]
- za uporabnika E: [Root → CA-2] : [CA-2 → E]



Slika 3: Certifikacijska pot v hierarhičnem modelu zaupanja

Za distribucijo certifikatov lahko poskrbijo uporabniki sami, tako da vsakič eden drugemu pošljejo svoj certifikat. Druga možnost je pridobitev certifikata posameznika iz globalne zbirke. X.509 je bil razvit v povezavi s standardom X.500, ki standardizira storitev omrežnega imenika. Če so certifikati objavljeni v X.500 imeniku, lahko do njih dostopamo preko LDAP (*Lightweight Directory Access Protocol*) protokola, ki omogoča dostop do imenika aplikacijam različnih platform. V X.500 imeniku lahko CA objavlja tudi CRL-je (*Certificate Revocation List*). CRL-ji so seznami na katerih CA objavi serijske številke preklicanih certifikatov, njihovo strukturo pa standardizira X.509. Poleg serijskih številk certifikatov se hranijo tudi informacije o razlogu in datumu preklica ter informacije o samem CRL-ju, kot so serijska številka, datum posodobitve, datum naslednje posodobitve, ime in digitalni podpis CA, ki je CRL objavila. CRL-ji so primerni tudi za aplikacije, ki nimajo direktnega dostopa do distribucijskih točk, ker jih lahko hranimo lokalno. Kot slabost CRL-jev se navaja velika zakasnitev med dejanskim preklicem certifikata in objavo preklica. Alternativa CRL-jem je protokol OCSP (*Online Certificate Status Protocol*), ki omogoča poizvedovanje o statusu certifikatov v realnem času in skoraj izniči zakasnitev med izvedbo in objavo preklica. Poleg

tega tudi zmanjšuje količino prenesenih podatkov, saj se običajno prenese informacija le za določen certifikat in ne celoten seznam preklicanih certifikatov. Ko uporabnik naslovi OCSP odzivnik z informacijami o certifikatu, mu odzivnik po obdelavi odgovori z enim od naslednjih statusov certifikata [1]:

- *veljaven*: certifikat je veljaven;
- *preklican*: certifikat je preklican;
- *nepoznan*: ni informacij o tem certifikatu.

Preverjanje preklica certifikatov je pogosto zanemarjen mehanizem delovanja PKI. Primer so spletni brskalniki, ki imajo pogosto izključeno preverjanje preklica certifikatov.

3.3.4. *Delovanje klientov*

Celotna PKI je v osnovi namenjena njenim klientom in jim ponuja mehanizme za varno izmenjavo informacij. Klient postane del PKI, ko preko prijavne službe od CA pridobi digitalni certifikat. Z digitalnim certifikatom se lahko klienti med seboj avtenticirajo in z uporabo pripadajočih ključev za asimetrične kriptografijo podpisujejo in šifrirajo sporočila. Posamezen klient mora pri tem poskrbeti za varnost lastnega privatnega ključa in za preverjanje digitalnih certifikatov, s katerimi se mu predstavijo ostali klienti. Klient mora pri preverjanju certifikata opraviti naslednje korake [1]:

1. preveriti datum veljavnosti certifikata in ga primerjati s trenutnim datumom;
2. preveriti identiteto klienta, tako da se prepriča, da je certifikat izdala in podpisala CA, ki ji zaupa;
3. prepričati se ali je certifikat izdan za namen za katerega se bo uporabil (razširitev *uporaba ključa*);
4. preveriti ali je certifikat preklican.

V primeru daljše certifikacijske poti, ko je med certifikatom, ki ga preverjamo, in certifikatom CA, ki ji zaupamo, več podpisnikov, je potrebno preveriti vse vmesne certifikate in preveriti, da imajo v razširitvi *uporaba ključa* navedeno podpisovanje certifikatov. Ko tako uspešno preverimo certifikat, lahko zaupamo identiteti in lastništvu privatnega ključa klienta na drugem koncu povezave in z njim vzpostavimo varen način komuniciranja.

4. Oblika sporočil

Sistemi, ki sodelujejo pri realizaciji določenih storitev, med seboj izmenjujejo podatke v obliki sporočil. Zato da lahko prejemnik na dokaj enostaven način pridobi podatke iz poslanega sporočila, je potrebno določiti obliko le-tega. Pri definiranju oblike sporočil lahko zasnujemo lasten način zapisa posameznih podatkov ali pa uporabimo že uveljavljen standard kot je XML, ki ne ponuja le sintakse za predstavitev oblike, ampak tudi številna orodja za obdelovanje tako predstavljenih podatkov. XML je odprt, neodvisen od platforme, razširljiv (*extensible*), označevalni (*markup*) jezik, ki je poenostavljena podmnožica SGML (*Standard Generalized Markup Language*) jezika. SGML je standardiziran meta jezik, s katerim lahko definiramo lasten označevalni jezik. Eden iz nabora tako definiranih označevalnih jezikov je HTML (*HyperText Markup Language*), ki je namenjen opisu prikaza podatkov. Primer jezika, ki je rezultat uporabe XML-ja in ne SGML-ja, pa je XHTML (*Extensible HyperText Markup Language*).

XML omogoča gradnjo označevalnih jezikov za različna problemska področja. Z XML-jem lahko definiramo strukturo dokumenta, ki je primerna za hranjenje in pošiljanje podatkov. Pri tem XML ne postavlja nobenih omejitev glede semantike, ki jo struktura predstavlja, ampak le definira pravila za gradnjo označb, s katerimi opišemo posamezne podatke in pravila za opisovanje relacij med podatki. Skupku XML označb in podatkov pravimo XML dokument, ki lahko predstavlja logično strukturo programa ali datoteke. Ne glede na to, kaj XML dokument predstavlja, ga lahko ena aplikacija kot tok znakov pošlje drugi. Pri tem mora aplikacija, ki sprejme XML dokument, za uspešno ekstrakcijo podatkov poznati le sintakso XML označb dokumenta. Postopke, potrebne za ekstrakcijo podatkov ob poznavanju sintakse določenega XML dokumenta, pa nudi že sam XML standard.

V okviru W3C (*World Wide Web Consortium*) konzorcija je bila leta 1996 ustanovljena skupina za razvoj XML standarda. Standard je bil razvit z namenom, da omogoči informacijskim sistemom enostavno, široko uporabno in učinkovito izmenjavo strukturiranih podatkov preko interneta. Trenutno je najnovejša četrta izdaja prve verzije W3C priporočila za XML [8], ki specificira tako leksikalno gramatiko kot tudi zahteve za procesiranje XML dokumentov. Procesiranje XML dokumenta se nanaša na postopke razčlenjevanja (*parsing*) dokumenta. Razčlenjevalniki XML dokumentov omogočajo dostop do vsebine in strukture tako, da preslikajo dokument iz znakovne zunanje predstavitve v notranjo programsko dostopno obliko. Poleg razčlenjevalnikov, programskih knjižnic in orodij za delo z XML-jem obstajajo še razne razširitve, ki še dodatno povečujejo uporabnost standarda. XML ni samo jezik, ki olajša branje dokumentov aplikacijam, ampak predstavlja tudi človeku čitljivo in samo opisujočo tekstovno obliko. Vse to pa je vzrok, da se XML danes vse bolj pogosto uporablja pri gradnji omrežnih komunikacijskih protokolov na aplikacijski plasti.

4.1. Osnove XML jezika

4.1.1. *Struktura dokumenta*

Strukturo XML dokumentov iz logičnega vidika sestavljajo deklaracije, elementi, komentarji in ukazi za procesiranje.

Ločimo dva tipa **deklaracij**, katerih navedba ni nujna in mora biti na začetku dokumenta. Prvi tip deklaracije je XML deklaracija in je vedno prva deklaracija. Kljub temu da navedba ni obvezna, je priporočena, saj z njo razčlenjevalnikom omogočimo pravilno obravnavo dokumenta. Z XML deklaracijo lahko napovemo verzijo XML-ja (atribut *version*),

neodvisnost dokumenta od zunanjih deklaracij (atribut *standalone*) in kodno tabelo za predstavitev znakov (atribut *encoding*). Pri čemer je verzija XML-ja obvezna, ostala dva atributa pa sta opcijska. Razčlenjevalniki lahko podpirajo poljuben nabor kodnih tabel za predstavitev znakov. Priporočilo zahteva le kodni tabeli UTF-8 in UTF-16.

Primer XML deklaracije:

```
<?xml version = "1.0" standalone="yes" encoding="UTF-8"?>
```

Drugi tip deklaracije, ki je v primeru navedbe obeh vedno druga, je DTD (*Document Type Definitions*). DTD je eden od načinov za definiranje strukture XML dokumentov in je podrobneje predstavljen v razdelku 4.1.2.

Z **elementi** predstavimo podatke in relacije med njimi. Vsak element predstavlja začetna in končna značka (*tag*) in vsebina elementa med njima. Za poimenovanje značk oz. elementov obstajajo preprosta pravila:

- ime je lahko katerakoli beseda iz števk, črk in drugih znakov (XML priporočilo skoraj nima rezerviranih besed);
- imena naj bodo preprosta in opisujoča, tako da je človeku vsebina dokumenta jasna brez dodatnih pojasnil;
- imena so občutljiva na velikost črk;
- uporaba znaka ":" je prepovedana, ker je rezerviran za navajanje imenskih prostorov;
- imena se morajo začeti s črko, začetek ne sme vsebovati katere koli kombinacije velikih in malih črk zaporedja "xml".

Vsebina elementa je lahko tudi eden ali več podelementov, mešana vsebina (tekst in podelementi) in samo tekst. Element je lahko tudi brez vsebine, kar lahko predstavimo kot zaporedje začetne in končne značke brez vmesne vsebine ali le kot značko, zaključeno z "/", kar razčlenjevalniku signalizira prazen element. Prazni elementi lahko služijo kot indikatorji prisotnosti določene lastnosti objekta ali zgolj kot nosilci atributov. Elemente lahko poljubno gnezdimo, pri čemer se začetne in končne značke ne smejo prekrivati. Vsak XML dokument mora imeti natanko en korenski element. Tudi za njegovo vsebino veljajo enaka pravila kot za ostale elemente.

Primer gnezdenja elementov:

```
<diploma> ←  korenski element
   <naslov>XML oblika</naslov> ←  element s tekstom
   <avtor>Janez Novak</avtor>
   <tipDiplome tip="UNI"/> ←  element brez vsebine in atributom
   <poglavja> ←  element, ki vsebuje več pod elementov
      <poglavje>Uvod</poglavje>
      <poglavje> ←  element z mešano vsebino
           Jedro diplome
           <podpoglavje>Deklaracije</podpoglavje>
           <podpoglavje>Elementi</podpoglavje>
           <podpoglavje>Komentarji</podpoglavje>
           <podpoglavje>Ukazi za procesiranje</podpoglavje>
      </poglavje>
      <poglavje>Zaključek</poglavje>
   </poglavja>
</diploma>
```

Imenski prostori so namenjeni preprečevanju konfliktov v poimenovanju elementov v XML dokumentu. Uporabni so v dokumentih, ki združujejo komponente številnih področij, ki uporabljajo ista imena, uporabljena v različnih kontekstih. Kot zgled lahko vzamemo XML

dokument, ki predstavlja ime gojitelja ptic in ime vrste ptic, ki jo goji. Če oba podatka predstavimo z elementom *ime*, postanejo podatki dvoumni. Lahko pa definiramo imenska prostora *oseba* in *ptica*, ki bosta podala področje uporabe poimenovanja. Privzeto imensko področje deklariramo tako, da v elementu navedemo atribut *xmlns* z vrednostjo, ki je enaka URI (*Uniform Resource Identifier*) naslovu, ki definira imensko področje. URI naslov lahko v realnosti ne kaže nikamor, ker se naslov uporabi le kot niz znakov za identifikacijo imenskega področja. Uporaba več imenskih področij hkrati zahteva uporabo predpon, s katerimi povemo, kateremu imenskemu prostoru element pripada. Predpono imenskega prostora deklariramo tako, da jo navedemo kot pripono atributa *xmlns* pri deklaraciji imenskega prostora. Deklaracija se podeduje v vse podelemente elementa, ki vsebuje deklaracijo, razen v tiste, ki deklarirajo nov imenski prostor z isto predpono. Imenski prostori se nanašajo tudi na atribute.

Primer uporabe imenskih prostorov:

```
<gojitelj
   xmlns:oseba="http://mojastran.com/ns/oseba/1.0"
   xmlns:ptica="http://mojastran.com/ns/ptica/1.0">
   <oseba:ime>Janez Novak</oseba:ime>
   <ptica:ime>domači golob</ptica:ime>
</gojitelj>
```

Atribute elementi lahko vsebujejo v začetni znački. Atributi so namenjeni dodatnemu opisu elementov in niso del njihove vsebine. Atribut navedemo kot ime in vrednost, ločeno z enačajem, pri čemer mora biti vrednost vedno v dvojnih ali enojnih narekovajih. XML priporočilo definira atributa *xml:lang* in *xml:space*. S prvim napovemo aplikaciji v katerem človeškem jeziku je vsebina elementa, medtem ko z drugim razčlenjevalniku napovemo ali naj ohrani in aplikaciji posreduje tipografske znake za vertikalni in horizontalni razmik (presledek, tab, prazna vrstica …).

Z **znakovnimi sekcijami (CDATA *sections*)** lahko v vsebinske dele XML dokumenta vključimo znake in izraze rezervirane za oblikovanje XML dokumenta. Razčlenjevalniku tako napovemo, da naj celoten sklop posreduje aplikaciji kot preprost tekst. Gnezdenje CDATA sekcij in uporaba zaporedij "]]>" znotraj CDATA sekcij ni dovoljeno.

Primer znakovne sekcije (značke *greeting* ne bodo obravnavane pri razčlenjevanju):

```
<![CDATA[<greeting>Hello, world!</greeting>]]>
```

Če ne uporabimo znakovne sekcije, moramo rezervirane znake predstaviti z ubežnimi sekvencami. Z ubežnimi sekvencami lahko predstavimo tudi znake, ki jih ni mogoče vnesti s tipkovnico ali na kakšen drug način. Ubežne sekvence ponujajo predstavitev z veljavno desetiško ("&#") ali šestnajstiško ("&#x") *Unicode* kodo. Grško črko π lahko tako predstavimo na desetiški način z "π" in šestnajstiški način z "π". XML priporočilo za znake, ki definirajo XML označbe, predpisuje tudi posebne reference:

- < → "<"
- > → ">"
- & → "&"
- ' → "'"
- " → """

Ukazi za obdelavo (*Processing Instructions*) so namenjeni aplikacijam in ne razčlenjevalniku, ki jih mora le posredovati aplikacijam. Z njimi lahko nudimo aplikacijam dodatne informacije za obdelavo vsebine dokumenta in predstavljajo ime aplikacije, kateri je ukaz namenjen, in prametre, ki morajo biti v aplikaciji razumljivi obliki. Ukazi za obdelavo niso del vsebine dokumenta.

Oblika ukaza za procesiranje:

```
<?IME-APLIKACIJE PRAMETRI?>
```

Komentarji so namenjeni dodajanju opomb XML dokumentu in kot ukazi za obdelavo niso del vsebine dokumenta. Nahajati se morajo zunaj posameznih deklaracij, značk in nikoli pred XML deklaracijo. Prav tako ni dovoljeno gnezdenje komentarjev in uporaba zaporedja znakov "--" znotraj komentarja.

Primer komentarja:

```
<!-- To je Komentar -->
```

Do sedaj smo opisali logično strukturo XML dokumenta, ki pa je lahko fizično razporejen čez več datotek ali drugih virov. Fizična struktura dokumenta je hierarhija entitet. Vsaka entiteta, razen entitete dokumenta, ima enolično ime. Entiteto dokumenta predstavljajo XML deklaracija, DTD in korenski element. Ostale entitete pa razdelimo na naslednji način:

- parametrične:
  - notranje,
  - zunanje;
- splošne:
  - notranje,
  - zunanje:
    - razčlenjene,
    - nerazčlenjene.

Splošne entitete so namenjene predstavitvi večjih delov podatkov, ki jih nato lahko z referenco entitete nadomestimo znotraj dokumenta. Zunanje splošne entitete se uporabljajo za vključevanje drugih dokumentov ali datotek, kot so slike, filmi, zvoki idr.

Primer deklaracije in reference splošne notranje entitete:

```
<!ENTITY FRI "Fakulteta za računalništvo in informatiko">
```
← deklaracija

```
&FRI;
```
← referenca

Notranje entitete so vedno razčlenjene, ker so bile obdelane ob razčlenjevanju XML datoteke. Za zunanje entitete, kot so slike, ki niso v XML obliki, velja, da so nerazčlenjene. Pri zunanjih entitetah moramo tako navesti njihovo lokacijo in v primeru nerazčlenjenosti navesti parameter NDATA.

Primer splošne zunanje nerazčlenjene entitete:

```
<!ENTITY logotip SYSTEM http://mojastran.com/logo.gif NDATA GIF>
```

Parametrične entitete so namenjene uporabi znotraj DTD in je razlika le pri referencah, kjer se namesto znaka "&" uporablja znak "%". Znak "%" se doda tudi pred ime parametrične entitete pri deklaraciji.

Deklaracije notacij služijo označevanju vsebine v dokumentu. Notacije se uporabljajo za prepoznavo oblike nerazčlenjenih entitet, oblike elementov, ki nosijo atribut podatkovnega tipa NOTATION, in aplikacij, katerim so namenjeni ukazi za obdelavo.

Primer uporabe notacije z ukazom za procesiranje:

```
<!NOTATION vsota SYSTEM file://math/vsota.exe>
<?vsota 4 5?>
```

XML razčlenjevalniki lahko XML dokument uspešno razčlenijo le v primeru, če je le-ta dobro oblikovan (*well-formed*). To pa je takrat, ko ustreza pravilom XML priporočila. V

nasprotnem primeru ni XML dokument. Pri tem morajo biti dobro oblikovani tudi vsi deli XML dokumenta, ki so vanj vključeni z uporabo deklaracij zunanjih entitet.

4.1.2. *Definiranje sintakse in pravilno oblikovan dokument*

V primeru, da imamo razčlenjevalnik, ki preverja tudi pravilnost XML dokumenta, potrebujemo mehanizme, s katerimi opišemo sintakso XML dokumenta. Dva izmed takih mehanizmov sta DTD in XML shema (*XML schema*). Oba sta v nadaljevanju na kratko predstavljena, podrobnosti pa so na voljo v [4]. Če razčlenjevalniku poleg XML dokumenta podamo tudi DTD ali XML shemo, bo v primeru, da je dokument dobro oblikovan, vrnil tudi informacijo, ali je pravilno oblikovan ali ne. Tako se lahko aplikacije vnaprej izognejo obdelavi nepravilno oblikovanih XML dokumentov.

DTD ponuja sintakso, s katero lahko opišemo strukturo dokumentov tako, da definiramo dovoljene entitete, elemente, podatkovne tipe, atribute in dovoljene načine gnezdenja elementov. DTD sintaksa je del XML in izvira iz izpeljave SGML jezika. DTD deklaracija je lahko del dokumenta ali pa se nahaja v posebni *dtd* datoteki, za katero je navedena referenca.

Primer, ko je DTD del dokumenta:

```
<?xml version="1.0"?>
<!DOCTYPE oseba [
  <!ELEMENT oseba    (ime,priimek,ulica,kraj)>
  <!ELEMENT ime     (#PCDATA)>
  <!ELEMENT priimek (#PCDATA)>
  <!ELEMENT ulica   (#PCDATA)>
  <!ELEMENT kraj    (#PCDATA)>
   <!ATTLIST oseba emso CDATA #REQUIRED>
]>
<oseba emso="0101991500123">
  <ime>Janez</ime>
  <priimek>Novak</priimek>
  <ulica>Tržaška 23</ulica>
  <kraj>1000 Ljubljana</kraj>
</oseba>
```

Primer, ko je DTD ločen od dokumenta:

```
<?xml version="1.0"?>
<!DOCTYPE oseba SYSTEM "oseba.dtd">  ← DTD referenca
<oseba emso="0101991500123">
  <ime>Janez</ime>
  <priimek>Novak</priimek>
  <ulica>Tržaška 23</ulica>
  <kraj>1000 Ljubljana</kraj>
</oseba>
```

Deklaracija DTD vedno vsebuje ime korenskega elementa, kateremu nato sledi referenca na zunanji opis ali pa celoten notranji opis strukture dokumenta.

Deklaracija strukture elementa je naslednje oblike:

```
<!ELEMENT   ime-elementa   vsebina-elementa>
```

Parameter *vsebina-elementa* lahko zavzame katero od naslednjih oblik:

- ANY – vsebina je lahko katerikoli deklariran element ali tekst.
- EMPTY – element je prazen.
- #PCDATA – vsebuje obdelan tekst.

- Skupina elementov navedena v oklepajih. Če so ločeni s presledki pomeni, da vrstni red ni pomemben, z vejicami pa, da je vrstni red pomemben. Z ločilom "|", napovemo, da je lahko prisoten le eden od navedenih elementov. Vsakemu od navedenih elementov ali celotni skupini lahko dodamo enega od znakov, ki definirajo število pojavitev elementa ali skupine:
    - + – ena ali več pojavitev;
    - * – nič ali več pojavitev;
    - ? – nič ali ena pojavitev;
    - brez – natanko ena pojavitev.

Podobno definiramo tudi atribute elementov:

```
<!ATTLIST  ime-elementa  ime-atributa  tip  privzeta-vrednost>
```

Parameter *tip* lahko zavzame eno izmed naslednjih vrednosti:

- CDATA – vrednost je niz znakov;
- v oklepajih navedeno zaporedje dovoljenih vrednosti ločenih z znakom "|";
- ID – vrednost je enolični identifikator elementa, noben drug atribut tipa ID v dokumentu ne sme zavzeti iste vrednosti;
- IDREF – vrednost je ID drugega elementa;
- IDREFS – zaporedje IDREF vrednosti ločenih s presledki;
- ENTITY – vrednost je ime entitete;
- ENTITIES – seznam ENTITY vrednosti;
- NMTOKEN – vrednost mora biti veljavno XML ime;
- NMTOKENS – zaporedje NMTOKEN vrednosti;
- NOTATION – vrednost je ime notacije.

Za privzeto vrednost pa lahko navedemo naslednje:

- *vrednost* – *vrednost* je privzeta vrednost atributa;
- #REQUIRED – atribut mora biti obvezno naveden;
- #IMPLIED – navedba atributa ni obvezna;
- #FIXED *vrednost* – vrednost atributa je nespremenljiva in je vedno enaka *vrednosti*.

XML shema je XML jezik, s katerim definiramo razred XML dokumentov in opišemo strukturo pravilno oblikovanih dokumentov. XML dokument je instanca sheme, katere omejitvam ustreza. Obsežno W3C priporočilo za jezik XML shem, specificira sintakso za uporabo imenskih prostorov in rekonstruira ter razširi zmožnosti DTD-jev iz XML 1.0 standarda. Kljub svoji obsežnosti, sheme povsem ne nadomestijo uporabe DTD-jev. Vse sheme uporabljajo imenski prostor *xmlns:xs="htp://www.w3.org/2001/XMLSchema"* in imajo korenski element poimenovan *schema*. Prednost XML shem je v tem, da uporabljajo XML sintakso in zato lahko za delo z njimi uporabljamo enaka orodja kot za delo z XML dokumenti in ne potrebujemo poznavanja posebne sintakse kot pri uporabi DTD-ja. XML sheme hranimo v obliki *xsd* datotek. Pripadnost dokumenta določeni shemi označimo tako, da v korenskem elementu navedemo atribut *xsi:schemaLocation*, ki podaja lokacijo *xsd* datoteke, in imenski prostor *xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"*. Relacija razred-instanca med shemo in dokumentom omogoča tudi pretvorbo XML dokumentov v objekte objektnih programskih jezikov in obratno (*object serialization, deserialization*). To omogoča shranjevanje objektov na pomnilne medije, klice oddaljenih procedur, distribucijo objektov in detekcijo sprememb v časovno spremenljivih podatkih.

Osnovni koncept XML shem so podatkovni tipi, s katerimi definiramo dovoljeno vsebino elementov in atributov. Ločimo preproste (*simpleType*) in kompleksne (complexType) podatkovne tipe.

Preprosti tip ne more vsebovati elementov in nima atributov. Preprostega tipa so vsi atributi, prazni elementi in elementi, ki vsebujejo tekst. Priporočilo shem definira kar nekaj osnovnih preprostih tipov, na katere pogosto naletimo v programskih jezikih: *string*, *integer*, *dateTime* idr. Prav tako so na voljo atributni tipi, kot so ID, IDREF in NMTOKEN, ki jih definira DTD. Pri tem lahko definiramo tudi lastne preproste tipe z dodajanje omejitev osnovnim tipom. Z omejitvami lahko definiramo razpone vrednosti, obliko številk, znakovne vzorce idr.

Primer lastnega preprostega podatkovnega tipa za predstavitev številke kreditne kartice (zaporedje štirih štiri mestnih številk dddd-dddd-dddd-dddd, kjer d predstavlja števko):

```
<xs:simpleType name="CreditCardNumber" >
  <xs:restriction base="xs:string">
    <xs:pattern value="\d{4}-\d{4}-\d{4}-\d{4}"/>
  </xs:restriction>
</xs:simpleType>
```

Kompleksni tip lahko vsebuje elemente in ima atribute. Če postavimo vrednost atributa *mixed* elementa *xs:complexType* na vrednost *true*, lahko element tega tipa vsebuje mešano vsebino. Veljavno vsebino in atribute elementov definiramo s pomočjo indikatorjev za:

- urejenost pod elementov
  - *all* – vrstni red ni važen, vsak element se mora pojaviti natanko enkrat,
  - *choice* – samo eden od elementov se lahko pojavi,
  - *sequence* – elementi se morajo pojaviti v navedenem vrstnem redu;
- število pojavitev pod elementov (obe vrednosti sta privzeto enaki 1)
  - *maxOccurs* – maksimalno število pojavitev posameznega elementa,
  - *minOccurs* – minimalno število pojavitev posameznega elementa;
- grupiranje
  - *group* – definira množico pogosto uporabljenih elementov,
  - *attributeGroup* – definira množico pogosto uporabljenih atributov.

Prisotnost elementa v dokumentu poda element *xs:element*, ki mora vsebovati enega izmed atributov *name* ali *ref*. V primeru, da je naveden atribut *name*, ki podaja ime elementa, mora biti prisoten tudi atribut *type*, ki podaja podatkovni tip elementa. Atribut *ref* podaja ime in s tem referenco globalnega elementa. Element je globalen, če se element *xs:element*, ki ga definira, pojavi kot podelement korenskega elementa *xs:schema*. Globalni elementi morajo imeti definiran podatkovni tip. Lokalni elementi so navedeni z elementom *xs:element*, ki je podelement bodisi elementa *xs:element* globalnega elementa bodisi elementa definicije kompleksnega tipa *xs:complexType*. Lokalni elementi imajo lahko kot atribut navedena indikatorja pojavitev, s čimer definiramo dovoljeno število pojavitev elementa. Pojavitve globalnih elementov na najvišjem nivoju ni mogoče omejiti, omejimo pa jih lahko pri uporabi reference na nižjih nivojih.

Atribute definiramo z elementom *xs:attribute* in je prav tako lahko globalen ali lokalen. Element *xs:attribute* lahko vsebuje atributa *type* (podatkovni tip) in *use*. Atribut *use* lahko zavzame vrednosti *required* ali *optional*, ki podajata nujnost prisotnosti atributa. Globalni atributi ne smejo imeti nastavljenega atributa *use*, ker se lahko uporabijo v kateremkoli elementu definiranem v shemi.

Privzete vrednosti atributov in elementov lahko definiramo z enim od atributov *default* (privzeta vrednost) ali *fixed* (fiksna vrednost) elementa *xs:attribute* ali *xs:element*.

Z elementoma za grupiranje (*xs:group* in *xs:attributeGroup*) lahko definiramo množico pogosto uporabljenih elementov ali atributov. Oba sta strukturirana podobno kot element *xs:complexType* in lahko s parametrom *ref* podajata referenco do globalne grupe elementov.

Element *xs:schema* lahko vsebuje atribut *targetNamespace* s katerim definira imenski prostor v katerem se nahajajo atributi in elementi, ki jih shema definira.

Primer XML sheme (nanaša se na primer, ki je podan pri opisu DTD):

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:element name="oseba">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="ime" type="xs:string"/>
            <xs:element name="priimek" type="xs:string"/>
            <xs:element name="ulica" type="xs:string"/>
            <xs:element name="kraj" type="xs:string"/>
         </xs:sequence>
         <xs:attribute name="emso" type="xs:integer" use="required"/>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

Fleksibilnost strukture dokumenta v XML shemi opišemo z uporabo podatkovnega tipa *anyType* in elementov *xs:Any* ter *xs:anyAttribute*. Podatkovni tip *anyType* dovoljuje uporabo katerekoli vsebine v elementu ali atributu tega tipa. Element *xs:Any* omogoča uporabo elementov, ki jih shema ne specificira. Element *xs:anyAttribute* ima enako vlogo kot *xs:Any*, le da omogoča uporabo nedefiniranih atributov.

4.1.3. *Razširitve XML jezika*

Funkcionalnost uporabe XML jezika dodatno povečujejo njegove razširitve:

- **XSLT** je jezik, ki omogoča transformacijo XML dokumentov v druge XML dokumente;
- **XQuery** je ekvivalent SQL (*Structured Query Language*) jezika za pridobivanje in manipuliranje podatkov iz XML dokumentov;
- **XInclude** nudi sposobnost XML datotekam, da vključijo zunanje datoteke ali le njihov del. Rezultat obdelave XInclude elementov, je vključitev kopij delov zunanjih datotek;
- **XML Podpis (XML Signature)** je mehanizem, ki definira sintakso in pravila za podpisovanje vsebine XML dokumentov (podrobneje predstavljen v nadaljevanju);
- **XML Šifriranje (XML Encryption)** je mehanizem, ki definira sintakso in pravila za šifriranje vsebine XML dokumentov (podrobneje predstavljen v nadaljevanju);
- **XPath** je jezik za naslavljanje posameznih delov XML dokumentov in hkrati model njihove predstavitve. Razširitvam kot so XSLT, XQuery, XML Signature in XML Encryption omogoča naključen dostop do vsebine, elementov, atributov, komentarjev, imenskih prostorov in ukazov za obdelavo;
- **XPointer** je sintaksa, ki razširi XPath tako, da omogoča ekstrakcijo posameznih delov zunanjih XML dokumentov. Omogoča, da URI referencam zunanjih XML dokumentov dodamo fragment v XPointer sintaksi, ki specificira želeni del XML dokumenta.

4.2. Šifriranje in podpisovanje XML dokumentov

Uporaba XML-ja na veliko področjih zahteva varnost, še posebno v smislu zaupnosti, integritete in avtentikacije avtorja XML dokumentov. V veliko primerih, kot so poslovne transakcije, ki lahko vključujejo več različnih uporabnikov, je zaželena možnost uporabe različnih stopenj integritete in zaupnosti posameznih delov XML dokumentov glede na to, kateremu prejemniku je vsebina posameznega dela namenjena. Kot primer lahko vzamemo kartoteko zaposlenega, katere splošni podatki, kot so telefonska števila ali naslov, so dostopni vsem znotraj podjetja, medtem ko so podatki o njegovi plači zaupne narave. S šifriranjem podatkov z različnimi ključi lahko zagotovimo različno stopnjo zaupnosti podatkov za različne razrede uporabnikov.

Varni XML (*Secure XML*) dobimo z uporabo razširitev XML podpisa in XML šifriranja, ki skupaj definirata sintakso in pravila za podpisovanje in šifriranje posameznih delov XML dokumentov. Tako lahko apliciramo različno stopnjo varnosti različnim delom XML dokumenta. Za vsako od razširitev obstajata priporočilo W3C konzorcija [15, 16], ki specificira sintakso in pravila za obdelavo. Podpisani in šifrirani deli so predstavljeni v XML sintaksi, kar omogoča uporabo orodij za delo z XML dokumenti in ostalih prednosti XML-ja.

Seveda za zagotavljanje varnosti XML dokumentov lahko uporabljamo tudi druge mehanizme, ki ne temeljijo na XML jeziku (npr. S/MIME – *Secure Multipurpose Internet Mail Extensions*). Vendar to zahteva vgradnjo dodatnih mehanizmov v sistem, s čimer lahko povečamo možnost za nekompatibilnost med različnimi sistemi.

Pri prenosih "točka–točka" lahko izmenjavo XML dokumentov zavarujemo z uporabo protokolov, kot so TLS (*Transport Layer Security*), SSL ali IPsec (*Internet Protocol Security*). S tem zavarujemo XML dokumente le med prenosom, ko pa enkrat prispejo do prejemnika, izgubimo tajnost podatkov, njihovo integriteto in identiteto pošiljatelja.

Z uporabo varnega XML-ja premaknemo varovanje podatkov iz nižjih plasti v aplikacijsko plast, ker zagotavljamo varnost na nivoju dokumenta. Za prenos pa lahko uporabimo tudi protokole, ki sami po sebi ne zagotavljajo varnosti. Podatki ostanejo zaščiteni tudi, če sistemi beležijo (*logging*) ali posredujejo sprejeta sporočila. Pri posredovanju tudi ni potrebe po ponovnem šifriranju, dešifriranju in podpisovanju podatkov v XML dokumentih. XML podpisovanje in šifriranje ni samo namenjeno varovanju vsebine XML dokumentov, ampak omogoča tudi varovanje datotek, ki niso XML dokumenti (npr. slike).

4.2.1. *Kanonična oblika dokumenta*

Pretvorba XML dokumenta v kanonično obliko pomeni pretvorbo dokumenta v standardno obliko. Pri tem se iz dokumenta izloči nepomembne površinske poglede predstavitve podatkov in omeji zgradbo dokumenta na eno samo možno.

Kanonična oblika je pomembna pri preverjanju digitalnega podpisa, kjer moramo zagotoviti, da bosta lahko oba pošiljatelj in prejemnik na enak način izračunala vrednost digitalnega podpisa dokumenta in ju primerjala. Primerjanje XML-jev je lahko zelo težaven proces, saj lahko različne ureditve teksta še vedno predstavljajo isti pomen podatkov. Drugačne ureditve teksta predstavljajo različne sekvence bitov, ki jih podpišemo. Preverjanje digitalnega podpisa je zato lahko neveljavno kljub temu, da so podatki pravilni. Med prenosom teksta med aplikacijami lahko naletimo na spremenjene znake za konec vrstice, dodane ali odstranjene presledke, zamenjavo tabulatorjev za presledke in obratno, zamenjavo kodne tabele za predstavitev znakov idr. Zato potrebujemo postopke za pretvorbo v kanonično

obliko, ki omogočajo, da lahko oba pošiljatelj in prejemnik pridobita enako obliko XML dokumenta. Pretvorba mora izločiti samo nepotrebne informacije in ne sme omogočati posega v strukturo dokumenta, ki bi lahko spremenil pomen podatkov in se ne bi odrazil v izračunu vrednosti podpisa.

Primera dveh ureditev elementa z enakim pomenom:

```
<pozdrav>Lep pozdrav</pozdrav>        <pozdrav>
                                        Lep pozdrav
                                      </pozdrav>
```

Poznamo dva postopka pretvorbe XML dokumenta v kanonično obliko. Prvi pretvori celoten dokument (*Canonical XML*), drugi pa le del dokumenta (*Exclusive XML Canonicalization*). Pri tem oba izpišeta izhod v UTF-8 predstavitvi znakov. Dokument najprej pretvorita v podatkovni model, ki je razširjen model XPath predstavitve dokumenta. XPath model predstavi XML dokument, kot hierarhično urejeno drevo vozlišč, ki prestavljajo posamezne dele dokumenta. Iz modela se nato izločijo nepotrebna vozlišča, preostalo drevo pa se pretvori v kanonično obliko XML dokumenta.

Kanonična oblika ne omogoča samo robustnosti integritete XML dokumentov, ampak je njena uporaba tudi priporočljiva pri izmenjavi dokumentov med različnimi okolji. Bolj podrobno je sam postopek pretvorbe (*XML Canonicalization*) opisan v [4].

4.2.2. *Podpisovanje*

Prisotnost XML podpisa v dokumentu nakazuje element *Signature*. Glede na to, kje se nahajajo podpisani podatki glede na element *Signature*, ločimo tri oblike [4]:

- podpisani podatki so podelement elementa podpisa (*enveloping signature*);
- podpisani podatki obdajajo element podpisa (*enveloped signature*);
- podatki se nahajajo ločeno od elementa podpisa, bodisi v istem dokumentu bodisi so v drugi datoteki (*detached signature*).

V enem dokumentu je lahko naenkrat prisotnih več podpisov različnih tipov.

Poenostavljena zgradba elementa *Signature* (podrobnosti elementov obdanih z "( )" so zaradi poenostavitve prikaza izpuščene, znaki +,*,? nakazujejo obveznost pojavitev, kot jo definira DTD):

```
<Signature ID?>
   <SignedInfo>
     <CanonicalizationMethod/>
     <SignatureMethod/>
     (<Reference URI? >
       (<Transforms>)?
       <DigestMethod>
       <DigestValue>
     </Reference>)+
   </SignedInfo>
   <SignatureValue>
  (<KeyInfo>)?
  (<Object ID?>)*
</Signature>
```

Element *SignedInfo* nudi informacije o podpisanih podatkih in algoritmih, ki so se pri tem uporabili. Pod element *CanonicalizationMethod* podaja algoritem, s katerim je bil element *SignedInfo* pretvorjen v kanonično obliko. *SignatureMethod* podaja algoritem, s katerim izračunamo podpis elementa *SignedInfo* po pretvorbi v kanonično obliko, rezultat pa je

vrednost elementa *SignatureValue*. Element *Reference* s parametrom *URI* podaja, kateri podatki so podpisani, in se mora pojaviti vsaj enkrat. Z opcijskim elementom *Transforms* lahko definiramo transformacije, s katerim pred tem obdelamo podatke. Element *DigestMethod* je obvezen in podaja algoritem, s katerim je bil izračunan prstni odtis podatkov. Izračunan prstni odtis podatkov je vsebina elementa *DigestValue*. Element *Object* se lahko niti ne pojavi ali pa se večkrat in omogoča vključitev podpisanih podatkov v podpis (*enveloping signature*).

Kompleksno zgradbo ima tudi element *KeyInfo*, ki nosi informacije o uporabljenem ključu. Element srečamo tudi pri XML šifriranju. Pojavitev ni obvezna, vsebuje pa lahko različne elemente, ki vsak na svoj način opisujejo ključ, ki je bil uporabljen pri digitalnem podpisu ali šifriranju. Nekateri izmed podelementov so naslednji:

- KeyName – vsebuje ime ključa, ki aplikaciji poenostavi iskanje po zbirki ključev;
- KeyValue – vsebuje vrednost uporabljenega ključa (npr. javni ključ);
- RetrievalMethod – opisuje postopke za pridobitev ključa;
- X509Data – lahko vsebuje različne podatke o digitalnem certifikatu, katerega ključi so bili uporabljeni. Vsebuje lahko tudi digitalni certifikat, CRL in seznam certifikatov, ki predstavljajo certifikacijsko pot certifikata;
- EncryptedKey – vsebuje zašifriran ključ.

Postopek generiranja podpisa:

1. Generiranje referenc:
    1. Pridobi podatke za podpis.
    2. Nad podatki opravi transformacije (npr. pretvorba XML-ja v kanonično obliko).
    3. Izračunaj vrednost prstnega odtisa transformiranih podatkov.
    4. Ustvari element *Reference* tako, da definiraš URI lokacijo podatkov, uporabljene transformacije, algoritme in rezultat.
2. Generianje podpisa:
    1. Ustvari element *SignedInfo*, v katerega vključiš informacije o algoritmih za pretvorbo v kanonično obliko in izračun podpisa ter generirane reference.
    2. Element *SignedInfo* z navedenim algoritmom pretvori v kanonično obliko.
    3. Izračunaj vrednost podpisa kanonične oblike elementa *SignedInfo* z navedenim algoritmom in ključem ter vrednost zapiši kot vsebino *SignedValue* elementa.
    4. Ustvari element *Signature*, v katerega vključiš elementa *SignedInfo* in *SignedValue*. Z uporabo elementa *KeyInfo* lahko dodaš še informacije o ključu uporabljenem pri generiranju podpisa.

Postopek verifikacije podpisa:

1. Verifikacija podpisa:
    1. Pridobi ključ za verifikacijo (vsebina *KeyInfo* elementa).
    2. Pretvori *SignedInfo* v kanonično obliko.
    3. Z uporabo ključa za verifikacijo in algoritma za izračun podpisa primerjaj rezultat prejšnjega koraka z vsebino *SignedValue* elementa.

2. Verifikacija referenc elementa *SignedInfo* v kanonični obliki:
    1. Pridobi podatke, na katere se referenca nanaša (*URI* atribut).
    2. Podatke transformiraj (*Transforms*) in nato izračunaj (*DigestMethod*) vrednost njihovega prstnega odtisa.
    3. Primerjaj prstni odtis z vsebino elementa *DigestValue* obravnavane reference.

Primer podpisa celotnega dokumenta:

```
<Document>
 <Field1> ... </Field1>
  <Signature>
   <SignedInfo>
    <CanonicalizationMethod
        Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
    <SignatureMethod
        Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha1"/>
    <Reference URI="">
      <Transforms>
        <Transform Algorithm=
                "http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
      </Transforms>
      <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
      <DigestValue> ... </DigestValue>
     </Reference>
    </SignedInfo>
    <SignatureValue> ... </SignatureValue>
    <KeyInfo>
        <X509Data><X509Certificate> ... </X509Certificate></X509Data>
     </KeyInfo>
 </Signature>
</Document>
```

4.2.3. *Šifriranje*

Šifrirane podatke v XML dokumentu predstavimo z elementom *EncrypedData*, ki vsebuje informacije o uporabljenih algoritmih in ključih ter šifrirane podatke. Struktura elementa *EncryptedKey*, ki vsebuje zašifriran ključ, je ekvivalentna strukturi elementa *EncryptedData*. Glede na to, kje se nahajajo šifrirani podatki, ločimo dve obliki [4]:

- šifrirani podatki so vsebina elementa *EncrypedData* (*enveloping encryption*);
- šifrirani podatki niso del elementa *EncrypedData*, ki vsebuje le referenco do lokacije, kjer se šifrirani podatki nahajajo (*detached encryption*).

Ko šifriramo podatke, ki so del istega dokumenta, smo omejeni na šifriranje celotnega elementa ali na šifriranje le njegove vsebine. V obeh primerih nadomestimo del, ki želimo, da ostane tajen, z elementom *EncryptedData*, ki predstavlja zašifrirane podatke. Pri šifriranju zunanjih datotek, ki niso del istega dokumenta, pa lahko element *EncryptedData* vključimo v katerikoli del dokumenta.

Poenostavljena zgradba elementa *EncryptedData* (podrobnosti elementov obdanih z "( )" so zaradi poenostavitve prikaza izpuščene, znaki +,*,? nakazujejo obveznost pojavitev, kot jo definira DTD):

```
<EncryptedData Id? Type? MimeType? Encoding? >
    <EncryptionMethod/>?
    (<KeyInfo>)?
    <CipherData>
      <CipherValue>?
      <CipherReference URI? >?
    </CipherData>
    <EncryptionProperties>?
 </EncryptedData>
```

Element *EncryptedData* vsebuje atributa *Id* in *Type*. Prvi služi zgolj za naslavljanje elementa znotraj dokumenta. Atribut *Type* pove, ali je bil šifriran element, vsebina elementa, ali kakšen drug tip podatkov, ki ga lahko dodatno opišemo z atributoma *MimeType* in

Encoding. *EncryptionMethod* nosi informacije o kriptografskem algoritmu in parametrih, ki so bili uporabljeni pri šifriranju podatkov. *KeyInfo*, tako kot v XML podpisu, nosi informacije o ključu, s katerimi prejemniku olajša delo pri iskanju ustreznega ključa za dešifriranje. Element *CipherData* je edini obvezni element in vsebuje šifrirane podatke v podelementu *CipherValue* ali pa referenco lokacije šifriranih podatkov v podelementu *CipherReference*. *CipherReference* lahko vsebuje tudi informacije o transformacijah, ki so potrebne, da iz naslova pridobimo šifrirane podatke (npr. XPath izraz, če se šifrirani podatki nahajajo v elementu drugega XML dokumenta). V elementu *EncryptionProperties* lahko navedemo dodatne informacije, kot je npr. časovni žig šifriranja.

Postopek šifriranja:

1. Predobdelava podatkov (npr. pretvorba v kanonično obliko).
2. Izberi algoritem za šifriranje.
3. Ustvari informacije o uporabljenem ključu. Če bo tudi ključ zašifriran, najprej rekurzivno izvedi postopek šifriranja za ključ.
4. Pridobi podatke, ki jih je potrebno zaščititi in jih zašifriraj.
5. Ustvari *CipherData* element.
6. Ustvari *EncryptedData* ali *EncryptedKey*, če šifriraš ključ.
7. Če so bili podatki element ali vsebina elementa, jih nadomesti z *EncryptedData* elementom. Drugače ustvari referenco do ustvarjenega elementa *EncryptedData* in ga umesti na primerno mesto v želen dokument.

Postopek dešifriranja:

1. Iz elementa *EncryptedData* ali *EncryptedKey* pridobi informacije o uporabljenem algoritmu (*EncryptedMethod*) in ključu (*KeyInfo*).
2. Pridobi šifrirane podatke iz podelementa *CipherValue* ali iz naslova, ki ga opisuje podelement *CipherReferenc*.
3. Z uporabo algoritma in ključa dešifriraj šifrirane podatke.
4. Po dešifriranju:
    a. Če je rezultat ključ, ga shrani za kasneje.
    b. Če je rezultat element ali vsebina elementa, z njim nadomesti element *EncryptedData*.
    c. Če je rezultat drug tip podatkov, ga obdelaj kot zahteva podatkovni tip.

Primer šifriranja elementa:

```
<PaymentInfo xmlns="http://example.org/paymentv2">
    <Name>John Smith</Name>
    <CreditCard Limit="5.000" Currency="USD">
      <Number>4019 2445 0277 5567</Number>
      <Issuer>Example Bank</Issuer>
      <Expiration>04/02</Expiration>
    </CreditCard>
</PaymentInfo>
```

↓XML šifriranje elementa CreditCard↓

```
<PaymentInfo xmlns="http://example.org/paymentv2">
    <Name>John Smith</Name>
    <EncryptedData Type="http://www.w3.org/2001/04/xmlenc#Element"
     xmlns="http://www.w3.org/2001/04/xmlenc#">
      <CipherData><CipherValue> ... </CipherValue></CipherData>
    </EncryptedData>
</PaymentInfo>
```

4.2.4. *Kombiniranje podpisovanja in šifriranja*

Kadar želimo določen sklop podatkov hkrati šifrirati in podpisati, se vedno pojavi vprašanje, ali prej šifrirati ali prej podpisati. Uporaba različnih oblik XML podpisa in šifriranja še dodatno poveča število možnosti. V nadaljevanju bodo predstavljene tri smiselne kombinacije in njihove prednosti in slabosti.

Šifriranje podpisanih podatkov in podpisa

Prednosti:

- Skrije podatke o podpisniku in metodi podpisa, ki jih vsebuje element *Signature*.
- Ni neuspešnega preverjanja podpisa, ker element *Signature* ni viden.
- Ker se podpis nanaša na podatke v prvotni obliki, je še vedno preverljiv po dešifriranju. Prav tako se lahko podpis ponovno uporabi pri šifriranju.

Slabosti:

- Napake in spremembe pri prenosu lahko odkrijemo šele po dešifriranju.
- Podpis lahko prejemnik zlorabi tako, da podatke in podpis ponovno zašifrira in pošlje drugemu uporabniku, ki nato misli, da je bilo sporočilo poslano direktno njemu. Sporočilo, kot je "Ljubim te!" lahko spravi pošiljatelja v veliko zadrego. To ni samo problem sporočil v digitalni obliki, ampak nanj naletimo tudi v realnem svetu. Ena možnih rešitev je, da poleg sporočila podpišemo še podatke o prejemniku in času podpisa.

Šifriranje podpisanih podatkov brez podpisa

Prednost:

- Ker se podpis nanaša na podatke v prvotni obliki, je še vedno preverljiv po dešifriranju. Prav tako se lahko podpis ponovno uporabi pri šifriranju.

Slabosti:

- Razkrije podatke o podpisniku in metodi podpisa, ki jih vsebuje element *Signature*.
- Napake in spremembe pri prenosu lahko odkrijemo šele po dešifriranju.

Podpis šifriranih podatkov

Prednost:

- Napake in spremembe pri prenosu lahko odkrijemo takoj z verifikacijo podpisa.

Slabosti:

- Podpis postane neuporaben, ko podatke dešifriramo. Če hočemo uporabiti podpis, si moramo zapomniti šifrirane podatke in ključ za dešifriranje. Dokazovanje podpisa tretji osebi pa zahteva razkrije ključa, kar ogrozi preostale podatke, ki so zaščiteni s tem ključem.
- Razkrije podatke o podpisniku in metodi podpisa, ki jih vsebuje element *Signature*.

5. Standard sintakse sporočil

OpenTravel je odprt standard, ki definira sintakso sporočil za komunikacijo med informacijskimi sistemi v potovalni industriji [12, 13]. OTA (*OpenTravel Alliance*) je skupnost podjetij iz področij letalskih prevozov, rent-a-car, hotelov, pomorskih križarjenj, železniških prevozov, potovalnih agencij in ponudbe informacijsko tehnoloških rešitev. OTA skupnost je razvila OpenTravel standard z namenom, da ponudi splošen, široko uporaben in preizkušen protokol ter s tem zmanjša stroške, ki nastanejo pri razvoju in vzdrževanju postopkov izmenjave informacij med ponudniki in distributerji v potovalni industriji.

Z uporabo XML-ja OpenTravel definira besedišče podatkovnih struktur v obliki elementov, atributov in ponovno uporabljivih entitet. Z uporabo XML shem so specificirane strukture sporočil, podatkovne strukture, podatkovni tipi in šifranti. Strukture sporočil ponujajo tudi razširljivost posameznih delov sporočil za specifične potrebe. Trenutna specifikacija standarda je 2008A in vsebuje sporočila za izmenjavo informacij v naslednjih domenah [5]:

- Generična sporočila (*Generic Messages*). Sporočila, namenjena splošni uporabi in niso specifična za določeno domeno. Na primer sporočilo za preklic rezervacije.
- Letalski prevozi (*Air Messages*).
- Najem avtomobila (*Car Messages*).
- Pomorsko križarjenje (*Cruise Messages*).
- Prireditve in ostale aktivnosti (*Destination Activities*). Sporočila, namenjena rezervaciji kart za prireditve in ostale lokalne aktivnosti.
- Golf igrišča (*Golf Tee Times*). Sporočila za iskanje, poizvedovanje in rezervacijo prostih terminov na golf igriščih.
- Hotel (*Hotel Messages*).
- Zavarovanja (*Insurance Messages*). Sporočila, namenjena sklepanju zavarovanj potnikov in njihovih potovanj.
- Potovalni oz. počitniški paketi (*PackageTours/HolidayBookings*).
- Izleti (*Tour Messages*).
- Železniški prevozi (*Rail Messages*).
- Evidenca zvestih strank (*Loyalty Messages*). Pojavljajo se podjetja, ki za podjetja v potovalni industriji vodijo programe in evidenco zvestih strank. Množica sporočil omogoča potovalni industriji komunikacijo s takimi podjetji.
- Kartoteke strank (Profile Messages). Ta sklop za potrebe potovalne industrije definira vsebino kartotek strank in sporočila za njihovo izmenjavo.

Standard za hotelsko področje definira obliko sporočil, ki omogočajo sporazumevanje med sistemi za vodenje hotelov in sistemi za distribucijo. Sporočila omogočajo izmenjavo opisnih podatkov hotelov, informacij o razpoložljivosti hotelov, pravil in omejitev za sklepanje rezervacij, informacij o sestavi cenikov, informacij o konferenčnih dogodkih, informacij, potrebnih za sklepanje in spreminjanje rezervacij idr. Tako lahko sistem za vodenje hotela GDS-ju v popolnosti posreduje logiko za sklepanje rezervacije. Obratno lahko tudi GDS posreduje sistemu za vodenje hotela poljubno podrobne podatke, na podlagi katerih se opravi rezervacija. Za podporo danim sporočilom standard definira tudi številne šifrante, ki dodatno olajšajo sporazumevanje. Specifikacija vsebuje šifrante za starostne razrede gostov, vrste obrokov, tipe ležišč, tipe plačilnih sredstev, načine obračuna gostov, tipe kapacitet, vrste sobnih dodatkov idr. Uporaba OpenTravel sintakse odstrani problem iskanja primerne in razširljive oblike sporočil. Za izmenjavo določenih podatkov je tako potrebno najti le ustrezno sporočilo, ki je namenjeno prenosu podatkov te vrste.

6. Prenos sporočil

Sporočila na poti od pošiljatelja do prejemnika lahko obiščejo več vmesnih postaj oz. sistemov, katerih naloga je dodatna obdelava in usmerjanje sporočil. Zato morajo sporočila poleg osnovnih podatkov, ki so predmet prenosa, vsebovati še dodatne podatke, ki so potrebni za uspešen prenos. Tako tvorimo neke vrste pisemsko ovojnico, katere vsebino tvorijo podatki, ki so predmet prenosa.

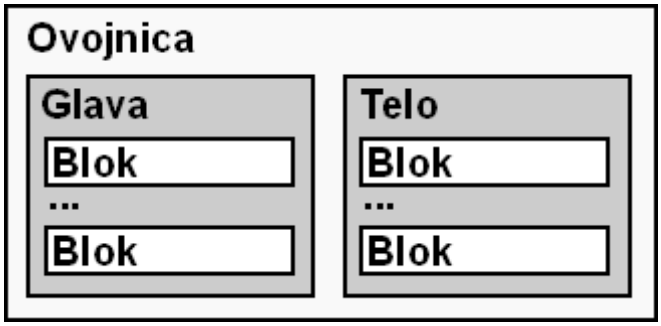
SOAP je enostaven protokol za izmenjavo strukturiranih informacij v decentraliziranih in distribuiranih okoljih [14]. SOAP z uporabo XML tehnologij definira razširljivo ogrodje za izmenjavo sporočil preko enostavnejših protokolov kot so TCP (*Transmission Control Protocol*), SMTP (*Simple Mail Transport Protocol*), HTTP idr. W3C priporočilo trenutno definira različica 1.2, s katero SOAP ni več omejen le na objektno orientirano uporabo, kot jo navaja izpeljava kratice imena.

Specifikacija SOAP vsebuje [4]:

- Definicijo splošne zgradbe sporočil.
- Splošna navodila za obdelavo SOAP sporočil na poti od izvora do ponora preko nobenega ali več vmesnih vozlišč.
- Metode za vračanje poročil o napakah.
- Specifikacijo za predstavitev raznovrstnih podatkovnih tipov v sporočilih.
- Specifikacijo za transport sporočil preko protokolov nižjih plasti.
- Uporabo SOAP-a za klice oddaljenih procedur.

Zaradi enostavnosti in razširljivosti SOAP izključuje posebnosti, kot so grupiranje sporočil, prenos objektov po referenci in distribuirano čiščenjem pomnilnika (*distributed garbage collection*), ki je potrebno za prenos objektov po referenci. Prav tako SOAP vsili nekaj omejitev v uporabo XML-ja v sporočilih:

- Dokumenti ne smejo vsebovati ukazov za obdelavo.
- Vsi elementi in atributi morajo pripadati primernemu imenskemu prostoru.
- DTD-ji so prepovedani. Aplikacije morajo zato poznati DTD-je oz. sheme.



Slika 4: Struktura SOAP sporočila

6.1. Struktura SOAP sporočila

SOAP sporočilo predstavlja ovojnica (element *Envelope*). Sporočilo je sestavljeno iz neobvezne glave (element *Header*) in obveznega telesa sporočila (element *Body*). Glava in telo lahko vsebujeta več blokov, s katerimi razširimo SOAP v poljuben protokol. Tako lahko na primer glava vsebuje blok za avtentikacijo uporabnika in integriteto sporočila, kot je element XML podpisa, ki je bil predstavljen v prejšnjem poglavju. Imenski prostor

xmlns:env="http://www.w3.org/2001/12/soap-envelope" definira elemente *Envelope*, *Header*, *Body* in pripadajoče atribute. Vsi trije elementi lahko vsebujejo več atributov ali pa nobenega. Te atribute definira *env* ali katerikoli drug definiran imenski prostor.

Primer SOAP sporočila:

```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <abc:control xmlns:abc="http://example.org/protocol">
      <abc:from name="user485"/>
      <abc:to name="user123"/>
    </abc:control>
  </env:Header>
  <env:Body>
    <abc:data xmlns:abc="http://example.org/protocol">
      ...
    </abc:data>
  </env:Body>
</env:Envelope>
```

Način predstavitve podatkovnih tipov definira atribut *encodingStyle*, ki se lahko pojavi kot atribut bloka glave, bloka telesa (če ne vsebuje elementa *Fault*, ki vsebuje opis napake), bloka podelementa *Detail* (podelement *Fault* elementa) in kot atribut elementov ovojnice. Lastnost, ki jo definira, se podeduje v vse podelemente oz. do redefinicije. Če atribut ni naveden na trenutnem ali višjem nivoju elementa, se smatra, da je vrednost atributa enaka "*http://www.w3.org/2003/05/soap-envelope/encoding/none"*, ki izključuje definicijo predstavitve podatkovnih tipov.

Vsi elementi, ki predstavljajo blok glave, morajo imeti definiran imenski prostor. Ta element ima lahko lastne atribute ali atribute, ki so potrebni za SOAP obdelavo:

- *encodingStyle*.
- *role* – pove katerim vozliščem je blok namenjen. Uporabimo lahko kateri koli URI naslov, specifikacija pa navaja naslednje tri vloge:
  - "http://www.w3.org/2003/05/soap-envelope/role/next" – namenjeno vsem vozliščem vključno s končnim;
  - "http://www.w3.org/2003/05/soap-envelope/role/none" – ni namenjeno vozliščem;
  - "http://www.w3.org/2003/05/soap-envelope/role/ultimateReceiver" – namenjeno le končnemu vozlišču.
- *mustUnderstand* – vozlišče, kateremu je namenjen blok s "*true*" vrednostjo atributa *mustUnderstand*, ki mora nujno obdelati vsebino bloka. Če vsebine bloka ne zna obdelati, mora vrniti napako.
- *relay* – z vrednostjo *"true"* nakazuje, da mora biti blok namenjen vozlišču po obdelavi vključen v sporočilo, ki bo posredovano naprej.

Vsi elementi, ki predstavljajo blok telesa, morajo imeti definiran imenski prostor. SOAP specifikacija definira tudi element *Fault*, ki služi kot blok telesa za poročanje o napakah. SOAP definira sporočilo, ki nosi poročilo o napaki kot sporočilo, katerega telo vsebuje samo en *Fault* element.

Fault element je definiran v imenskem prostoru *env* in mora nujno vsebovati elementa *Code* in *Reason*. Poleg tega pa lahko še vsebuje elemente *Node*, *Role* in *Detail*. Element *Code* opiše tip napake s podelementoma *Value* in *Subcode*. *Value* lahko vsebuje eno izmed naslednjih vrednosti:

- *VersionMismatch* – korenski element ali njegov imenski prostor ni prepoznan.
- *MustUnderstand* – vozlišče ne zna obdelati bloka glave, ki mu je namenjen, in ima parameter *mustUnderstand* nastavljen na vrednost "*true*".
- *DataEncodingUnknown* – predstavitev podatkov, uporabljena v glavi ali telesu, ni podprta.
- *Sender* – sporočilo ni bilo pravilno oblikovano oz. ni vsebovalo pričakovane vsebine.
- *Receiver* – razlog za napako ni v samem sporočilu, ampak v postopku obdelave. Na primer sistem ni uspel pridobiti določenih informacij od drugih sistemov in zato ne more pravilno obdelati sporočila.

Pri poročilu o napaki tipa *VersionMismatch* SOAP predvideva, da naj bi sporočilo v glavi vsebovalo element *Upgrade*, ki s podelementi *SupportedEnvelope* sporoči pošiljatelju, katere verzije SOAP podpira. Prav tako naj bi ob napaki tipa *MustUnderstand* glava sporočila vsebovala enega ali več elementov *NotUnderstood*, ki vsebujejo informacije o blokih glave z atributom *mustUnderstand* z vrednostjo "*true*", ki jih vozlišče ni znalo obdelati.

Subcode element ni obvezen in lahko prav tako vsebuje element *Value*, ki tokrat lahko zavzame vrednost, ki je specifična za posamezno aplikacijo. Element *Reason* vsebuje obvestilo o napaki v človeku razumljivi obliki, ki je vsebina enega ali več podelementov *Text*. Element *Node* vsebuje URI, ki identificira vozlišče, pri katerem je prišlo do napake. Vlogo tega pa predstavimo z elementom *Role,* ki vsebuje URI, ki predstavlja vlogo. Element *Detail* je namenjen prenosu informacij o napaki v XML strukturi, ki je specifična za posamezno aplikacijo.

6.2. Prenos SOAP sporočila

Za prenos SOAP sporočila od pošiljatelja do prejemnika je potrebna uporaba enega izmed transportnih protokolov, kot so TCP, SMTP, HTTP idr. Katerega se uporabi za transport sporočil, je odvisno od okolja, v katerem se bo dana implementacija uporabila. Najpogosteje se danes za transport SOAP sporočil uporablja HTTP protokol, katerega prednost je, da ne povzroča težav pri prenosih preko požarnih zidov. To hkrati ponuja tudi možnost uporabe HTTPS protokola, ki poleg tajnosti sporočil ponuja tako enosmerno kot tudi vzajemno avtentikacijo. Uporaba takšne oblike transporta za prenos SOAP sporočil je ključnega pomena pri zagotavljanju varnosti klicev procedur realiziranih v obliki spletnih storitev (*web service*).

7. Postavitev komunikacijskega protokola

Zato da lahko hotelir in gost skleneta posel preko portala v realnem času, moramo med portalom in hotelom vzpostaviti direkten način komuniciranja. Pravila, potrebna za varno in pravilno interakcijo med procesi, ki opravljajo omenjeno storitev, definiramo s komunikacijskim protokolom.

WebRes protokol omogoča uporabniku poizvedovanje po hotelskih kapacitetah in rezervacijo le-teh v realnem času. Ime WebRes izhaja iz okrajšave angleške besedne zveze "*Web Reservations*", ki tudi poimenuje storitev spletnih rezervacij, ki jo nudi spletni rezervacijski sistem. Po tem protokolu se preko komunikacijskega strežnika sporazumevajo spletni portal, pomožni odjemalec in vmesnik hotelskega rezervacijskega sistema (v nadaljevanju vmesnik hotela). Rdeče povezave na sliki 1 prikazujejo povezave, po katerih deluje WebRes protokol.

WebRes protokol je realiziran na aplikacijski plasti. Za prenos sporočil protokola se uporablja TCP protokol, ki spada v četrto plast TCP/IP arhitekturnega modela za računalniško komunikacijo. TCP je namenjen vzpostavljanju povezave med aplikacijama (npr. brskalnik in spletni strežnik). Aplikacijam TCP nudi zanesljiv in sekvenčen prenos toka bajtov (zlogov) med odjemalcem in strežnikom. TCP spada med povezane protokole, kar pomeni, da gre vsaka povezava skozi faze vzpostavljanja povezave, prenosa podatkov in prekinitve povezave. Za vzpostavitev povezave s strežnikom mora odjemalec poznati strežnikov IP (Internet Protocol) naslov in vrata, na katerih strežnik sprejema povezave. TCP ne prenaša sporočil, ampak pakete bajtov. Zato morajo aplikacije same poskrbeti, da sprejmejo sporočilo v celoti.

WebRes protokol implementira več uveljavljenih standardov. Protokol uporablja XML in z njim povezane rešitve na naslednje načine:

- Za izmenjavo sporočil se uporablja SOAP preko TCP povezave.
- Vsebina sporočil je predstavljena z OpenTravel sintakso.
- Za preverjanje pravilnosti zgradbe se uporabljajo XML sheme.
- Varnost v smislu integritete in tajnosti podatkov se zagotavlja z uporabo XML podpisa in XML šifriranja.
- Avtentikacija uporabnikov in ne-zanikanje udeleženosti v transakcijah se zagotavlja z uporabo XML podpisa v povezavi z infrastrukturo javnih ključev.

Uporaba standardov pri definiranju protokola poenostavi njegovo končno implementacijo. Standardi namreč v večini primerov nudijo že preizkušene in premišljeno zasnovane algoritme. Seveda pa se moramo zavedati, da so standardi običajno namenjeni precej širši uporabi, kot jo potrebuje naš sistem in zato lahko predstavljajo manj učinkovito implementacijo. Sporočila bodo zaradi uporabe XML-ja nekoliko daljša. Vendar se ne izmenjujejo preko interneta nič bolj pogosto kot spletne strani, ki jih spletni strežnik portala posreduje na zahtevo uporabnikom. Vzrok za izmenjavo sporočil so namreč ravno uporabniki, ki od portala zahtevajo informacije o ponudbi hotelov.

7.1. Osnovna zgradba sporočil

WebRes protokol za izmenjavo sporočil uporablja SOAP (različica 1.2). Pri tem glava SOAP ovojnice vsebuje podatke, ki so potrebni za uspešen prenos sporočila, telo pa vsebuje vsebino sporočila. Prav tako nam SOAP definira tudi obliko poročanja o napakah pri komuniciranju.

7.1.1. *Glava sporočila*

Strukturo glave sporočila predstavimo s posebnim elementom *WebResHeader*, ki ga vključimo v SOAP ovojnico kot blok glave. Čeprav SOAP definira glavo kot neobvezen element, sta glava in blok glave *WebResHeader* v WebRes protokolu zahtevana. Vsebina elementa *WebResHeader* vključuje v glavo podatke, ki so ključnega pomena za prenos in zagotavljanje integritete sporočil. Da zagotovimo posredovanje in upoštevanje elementa *WebResHeader* v vseh vozliščih, mu v skladu s SOAP standardom nastavimo naslednje atribute:

- *role="http://www.w3.org/2003/05/soap-envelope/role/next"*;
- *mustUnderstand="true"*;
- *relay="true"*.

Primer umestitve glave sporočila:

```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
   <env:Header>
      <WebResHeader xmlns="http://www.hotel.si/webres/2008/08/header"
             env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
             env:mustUnderstand="true"
             env:relay="true">
        <Sender>WEBRES81346_HFRI</Sender>
        <Receiver>HFRI</Receiver>
        <Subject>Book_AvailRQ</Subject>
        <TimeStamp>2008-08-25T15:24:13.013</TimeStamp>
        <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
           ...
        </Signature>
      </WebResHeader>
   </env:Header>
   <env:Body>
      ...
   </env:Body>
</env:Envelope>
```

Glava sporočila vsebuje uporabniško ime pošiljatelja, uporabniško ime prejemnika, tip vsebine sporočila, časovni žig pošiljanja in digitalni podpis pošiljatelja. Strukturo elementa *WebResHeader* definira XML shema na siki 5, posamezni elementi pa so predstavljeni v nadaljevanju.

Uporabniško ime pošiljatelja vsebuje element *Sender*. Z njim se pošiljatelj prijavi v komunikacijski strežnik in predstavlja enolično identifikacijo pošiljatelja pri komuniciranju. Prejemniku nudi informacijo o tem, kdo mu pošilja sporočilo in komu mora poslati odgovor.

Uporabniško ime prejemnika vsebuje element *Receiver*. Z njim se prejemnik prijavi v komunikacijski strežnik in predstavlja enolično identifikacijo prejemnika pri komuniciranju. Pošiljatelj s tem sporoča komunikacijskemu strežniku, komu naj posreduje sporočilo.

```
<xs:schema  xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://www.hotel.si/webres/2008/08/header"
    xmlns:tns="http://www.hotel.si/webres/2008/08/header" elementFormDefault="qualified">
    <xs:element name="WebResHeader" type="tns:headerType"/>
    <xs:complexType name="headerType">
        <xs:sequence>
            <xs:element name="Sender" type="xs:string"/>
            <xs:element name="Receiver" type="xs:string"/>
            <xs:element name="Subject" type="tns:subjectType"/>
            <xs:element name="TimeStamp" type="xs:dateTime"/>
            <!-- vključitev XML podpisa -->
            <xs:any namespace="http://www.w3.org/2000/09/xmldsig#" processContents="lax"/>
        </xs:sequence>
        <!-- vključitev atributov  za SOAP Header blok -->
        <xs:anyAttribute namespace="http://www.w3.org/2003/05/soap-envelope"
                         processContents="lax"/>
    </xs:complexType>
    <xs:simpleType name="subjectType">
      <xs:restriction base="xs:string">
        <xs:enumeration value="Auth_Nonce"/>
        <xs:enumeration value="Auth_Client"/>
        <xs:enumeration value="Auth_Server"/>
        <!-- rezervacija sob -->
        <xs:enumeration value="Book_AvailRQ"/>
        <xs:enumeration value="Book_AvailRS"/>
        <xs:enumeration value="Book_OfferRQ"/>
        <xs:enumeration value="Book_OfferRS"/>
        <xs:enumeration value="Book_PriceRQ"/>
        <xs:enumeration value="Book_PriceRS"/>
        <xs:enumeration value="Book_GuaranteeRQ"/>
        <xs:enumeration value="Book_GuaranteeRS"/>
        <xs:enumeration value="Book_ReservationRQ"/>
        <xs:enumeration value="Book_ReservationRS"/>
        <!-- posodabljanje statične vsebine -->
        <xs:enumeration value="Update_StaticDataRQ"/>
        <xs:enumeration value="Update_StaticDataRS"/>
        <!-- preverjanje prisotnosti -->
        <xs:enumeration value="Ping_RQ"/>
        <xs:enumeration value="Ping_RS"/>
        <!-- napaka -->
        <xs:enumeration value="Fault"/>
      </xs:restriction>
    </xs:simpleType>
</xs:schema>
```

Slika 5: XML shema elementa *WebResHeader*

Tip vsebine sporočila predstavlja vsebina elementa *Subject*, ki nakazuje enega od 18 možnih tipov sporočil, ki jih trenutno WebRes protokola implementira. Prejemniku nudi informacijo o tem, kakšna vsebina se nahaja v telesu sporočila. Protokol predvideva pet sklopov sporočil. Prvi sklop sporočil (predpona *Auth*) predstavljajo sporočila za avtentikacijo odjemalca v komunikacijskem strežniku. Drugi sklop (predpona *Book*) predstavljajo sporočila namenjena povpraševanju po kapacitetah hotelov in rezervaciji le-teh. Tretji sklop (predpona *Update*) sporočil omogoča posodabljanje statične vsebine o hotelu. Četrti sklop (*Fault*) predstavlja tip sporočila, ki nakazuje, da je prišlo do napake pri prenosu prejšnjega sporočila. Zadnji sklop sporočil (predpona *Ping*) predstavlja par sporočil za preverjanje prisotnosti s

komunikacijskim strežnikom povezanih odjemalcev. Tako lahko na podlagi navedenega tipa sporočila aplikacija:

- izbere ustrezno shemo, s katero bo preverila pravilnost oblike vsebine;
- izbere pravilne postopke za obdelavo vsebine;
- zavrne sporočilo, če ni pričakovanega tipa.

Element *TimeStamp* vsebuje časovni žig pošiljanja sporočila.

Digitalni podpis služi preverjanju integritete sporočila in hkrati tudi dokazuje pošiljateljevo avtorstvo sporočila. Element *Signature* je rezultat uporabe XML podpisa. Podpis se mora nanašati na celotno sporočilo, s čimer dobimo obliko XML podpisa, kjer podpisani podatki obdajajo element podpisa (*enveloped signature*). Element *Signature* je obvezen pri vseh tipih sporočila in služi tudi za izmenjavo digitalnih certifikatov, ki se prenesejo kot vsebina elementa *KeyInfo*.

7.1.2. *Telo sporočila*

V telo SOAP ovojnice vključimo blok, ki predstavlja vsebino sporočila. Vsebino predstavljajo podatki, katerih izmenjava je vzrok za pošiljanje samega sporočila. Vsebina mora vedno ustrezati tipu, navedenemu v glavi sporočila. V nasprotnem primeru prejemnik ne bo mogel obdelati vsebine sporočila in bo odgovoril z napako. Za predstavitev vsebine sporočil, ki se nanašajo na ponudbo in rezervacijo hotelskih kapacitet, WebRes protokol uporablja OpenTravel sintakso. Pri tem iz nabora sporočil za hotelsko področje uporablja pare sporočil za povpraševaje po razpoložljivostih, sklepanje rezervacij in posodabljanje podatkov o hotelu. Poročila o napakah, ki so rezultat neuspešne izmenjave sporočil, se prenašajo v skladu s SOAP specifikacijo z uporabo elementa *Fault*. Ker WebRes protokol podpira vrsto mehanizmov za preverjanje izmenjave sporočil, vsebuje tudi lastne kode napak. Za bolj podrobno pojasnjevanje napak je definiran element *FaultCode*, ki ga vključimo v poročilo o napaki kot podelement elementa *Detail*.

Element *FaultCode* definira naslednje kode napak:

- *NotWellFormed* – sprejeto sporočilo ni dobro oblikovan XML dokument,
- *InValid* – sprejeto sporočilo ne ustreza sintaksi, ki jo opisujejo XML sheme,
- *Signature* – neuspeh pri preverjanju digitalnega podpisa sporočila,
- *Decryption* – neuspeh pri dešifriranju sporočila,
- *Certificate* – preverjanje digitalnega certifikata je bilo neuspešno,
- *MissingData* – sporočilo ne vsebuje vseh potrebnih podatkov,
- *WrongSubject* – nepričakovan tip vsebine,
- *NoReceiver* – prejemnik trenutno ni prisoten,
- *AuthFailed* – prijava na komunikacijski strežnik je bila neuspešna,
- *Other* – ostalo.

Od naštetih kod napak se prvih sedem nanaša na SOAP status *Sender*, osma in deveta na status *Receiver* ter zadnja na splošno poročanje o napakah. Dodatne informacije o napaki pa lahko priložimo kot vsebino elementa *Reason*.

Primer pojasnila o napaki:

```
<env:Fault>
  <env:Code>
    <env:Value>env:Sender</env:Value>
  </env:Code>
  <env:Reason>
    <env:Text xml:lang="en">HotelCode Missing</env:Text>
  </env:Reason>
  <env:Detail>
    <FaultCode xmlns="http://www.hotel.si/webres/2008/08/faultcode">
      MissingData
    </FaultCode>
  </env:Detail>
</env:Fault>
```

Napake, ki niso posledica prenosa, kot je na primer negativen odgovor vmesnika hotela zaradi zasedenosti hotela, se prenašajo v obliki, ki jo predvideva par sporočil za izmenjavo določene informacije.

Posebno strukturo vsebine sporočila definira protokol tudi za izmenjavo podatkov pri prijavi odjemalca na komunikacijski strežnik. Ta struktura vsebine je podrobneje pojasnjena pri opisu postopka avtentikacije v razdelku 7.2.1.

```
<xs:schema  xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://www.hotel.si/webres/2008/08/faultcode"
    xmlns:tns="http://www.hotel.si/webres/2008/08/faultcode" elementFormDefault="qualified">
    <xs:element name="FaultCode" type="tns:webresFaultCode"/>
    <xs:simpleType name="webresFaultCode">
        <xs:restriction base="xs:QName">
            <xs:enumeration value="tns:NotWellFormed"/>
            <xs:enumeration value="tns:InValid"/>
            <xs:enumeration value="tns:Signature"/>
            <xs:enumeration value="tns:Decryption"/>
            <xs:enumeration value="tns:Certificate"/>
            <xs:enumeration value="tns:MissingData"/>
            <xs:enumeration value="tns:WrongSubject"/>
            <xs:enumeration value="tns:NoReceiver"/>
            <xs:enumeration value="tns:AuthFailed"/>
            <xs:enumeration value="tns:Other"/>
        </xs:restriction>
    </xs:simpleType>
</xs:schema>
```

Slika 6: XML shema elementa *FaultCode*

Za prenos binarnih podatkov, kot so slike hotelskih sob in potrdilna pisma, se uporablja pretvorba binarnih podatkov v Base64 znakovno predstavitev. Base64 za predstavitev uporablja nabor 64 znakov, ki so si sorodni med posameznimi kodnimi tabelami. S tem zmanjšamo možnost spremembe podatkov pri prehodu skozi različne sisteme. Táko predstavitev binarnih podatkov v XML dokumentu uporabljata tudi XML podpis in XML šifriranje.

Zaupnost vsebine sporočil zagotavlja uporaba XML šifriranja. Ker oba standarda XML šifriranje in XML podpis v svoji strukturi opišeta uporabljene algoritme in ključe, ni potrebe po dogovarjanju o le-teh. Zato WebRes protokol ponuja veliko fleksibilnost pri zaščiti podatkov. Tako lahko zaščitimo le določene dele sporočila. Lahko izbiramo tudi med različno

močno zaščito podatkov. Za manj pomembne podatke lahko uporabimo bolj šibke algoritme in s tem pospešimo določene operacije. Zaščiteno vsebino sporočila predstavlja element XML šifriranja, ki je vstavljen v SOAP ovojnico kot blok telesa. Glave sporočila se ne šifrira, ker morajo biti podatki glave na voljo zaradi usmerjanja sporočila. Integriteto celotnega sporočila pa zagotavlja digitalni podpis v glavi, ki se nanaša na celotno SOAP ovojnico.

7.1.3. *Rokovanje s sporočili*

Vsa sporočila preidejo skozi fazi pošiljanja in sprejemanja. Večina sporočil pa bo prešla tudi skozi fazo posredovanja na komunikacijskem strežniku. Pri tem se sporočila obdelujejo v skladu s SOAP modelom. WebRes protokol razširja SOAP v svoje namene, zato obdelava sporočil v posameznih fazah zahteva nekaj dodatnega preverjanja.

Pošiljanje

V vlogi pošiljatelja nastopajo vse komponente sistema, vpletene v komunikacijo z WebRes protokolom. Pošiljatelj mora poskrbeti, da je sporočilo pravilno zgrajeno in odposlano. Fazo pošiljanja sporočila predstavljajo naslednji koraki:

1. **Priprava vsebine sporočila**. Pošiljatelj pravilno oblikuje enega izmed 18 različnih tipov vsebine.
2. **Šifriranje vsebine sporočila**. Pošiljatelj pripravljeno vsebino sporočila zašifrira s pomočjo XML šifriranja in javnega ključa, ki pripada digitalnemu certifikatu prejemnika. Pri tem naj se za šifriranje dokumenta uporabi simetrični blokovni algoritem v CBC načinu z naključno generiranim ključem. Ta ključ nato z uporabo javnega ključa prejemnika zašifriramo z RSA algoritmom in ustvarimo pripadajoči *EncryptedKey* element. *EncryptedKey* nato z elementom *KeyInfo* vključimo v element *EncryptedData*, ki predstavlja zašifrirano vsebino. Tako združimo prednosti simetrične in asimetrične kriptografije. Vsi simetrični algoritmi za blokovno šifriranje, ki jih zahteva standard za XML šifriranje, zadoščajo varovanju zaupnih podatkov. Na splošno se je potrebno izogibati le uporabi algoritmov, ki jih standard navaja kot opcijske, ker obstaja možnost, da bo prejemnik implementiral knjižnico za XML šifriranje, ki bo podpirala le zahtevane algoritme.
3. **Priprava glave sporočila**. Pošiljatelj zgradi element *WebResHeader* brez elementa *Signature*. V njem navede svoje uporabniško ime kot ime pošiljatelja in prejemnikovo ime kot ime prejemnika. Prav tako navede tudi trenutni časovni žig in tip vsebine, ki ga je v prejšnjem koraku zašifriral.
4. **Izgradnja SOAP ovojnice**. Pošiljatelj v glavo SOAP ovojnice vstavi rezultat 3. koraka in v telo SOAP ovojnice rezultat 2. koraka.
5. **Podpis sporočila**. Z uporabo XML podpisa pošiljatelj podpiše celotno sporočilo z uporabo lastnega digitalnega certifikata. Element *Signature* nato vstavi v glavo na primerno mesto.
6. **Pošiljanje sporočila.** Pošiljatelj pošlje sporočilo.

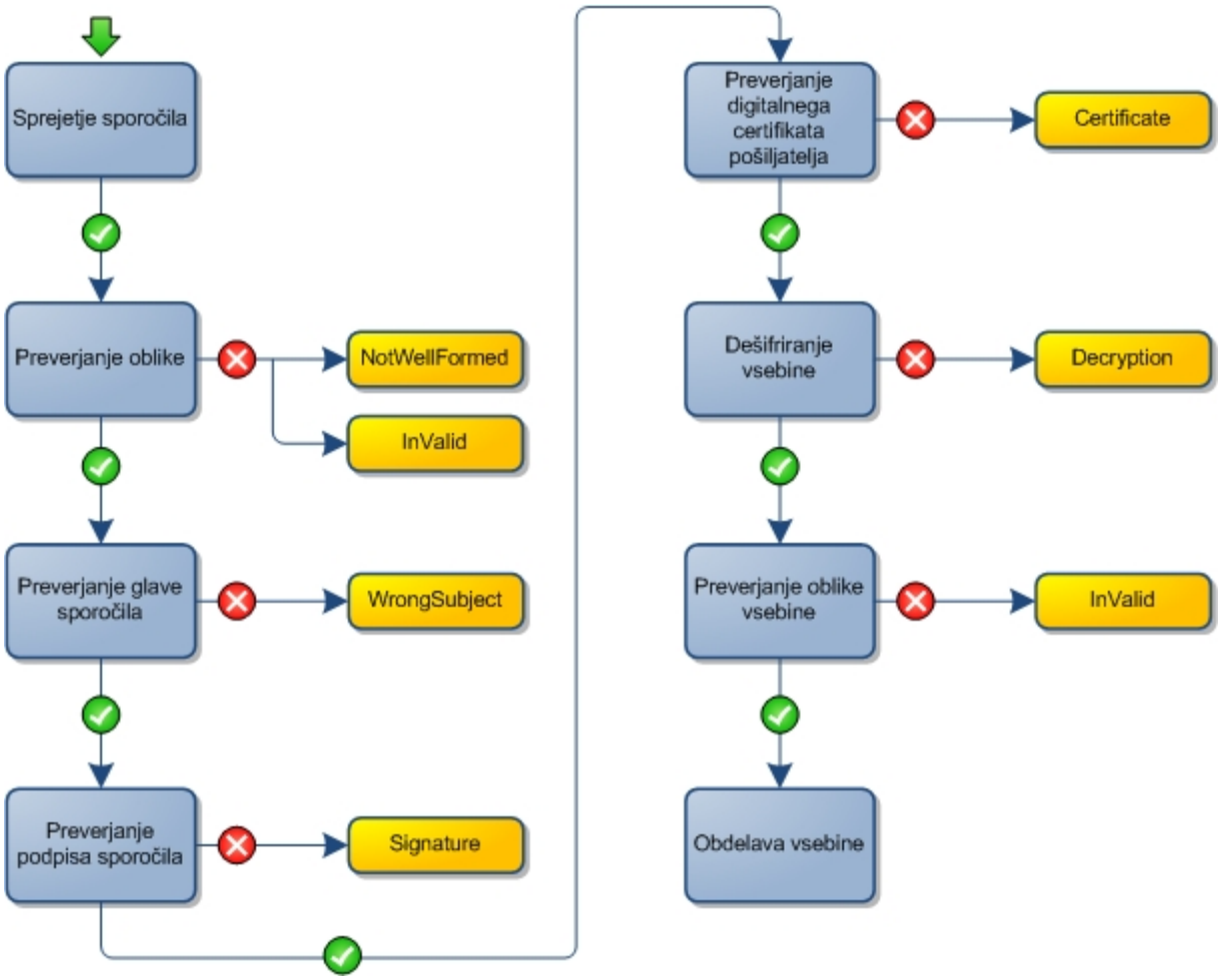
Sprejemanje

V vlogi prejemnika prav tako nastopajo vse komponente sistema, vpletene v komunikacijo z WebRes protokolom. Prejemnik mora preveriti, da sporočilo med prenosom ni bilo poškodovano ali spremenjeno. Fazo sprejemanja sporočila predstavljajo naslednji koraki:

1. **Sprejetje sporočila**. Prejemnik sprejme sporočilo.
2. **Preverjanje oblike**. Sprejeto sporočilo najprej obdela XML razčlenjevalnik, ki nato opravi še preverjanje pravilnosti oblike s pomočjo XML shem. V primeru da oblika sporočila ne ustreza pravilom XML oblikovanja, prejemnik

pošiljatelju odgovori s kodo napake *NotWellFormed*. V primeru, ko spodleti preverjanje pravilnosti oblike, pa s kodo *InValid*.

3. **Preverjanje glave sporočila**. Prejemnik preveri, ali je tip vsebine ustrezen. Če tip ni ustrezen, pošiljatelju odgovori s kodo napake *WrongSubject*.
4. **Preverjanje podpisa sporočila**. Prejemnik preveri XML podpis sporočila in v primeru neuspeha odgovori s kodo napake *Signature*.
5. **Preverjanje digitalnega certifikata pošiljatelja**. Prejemnik preveri, ali uporabniško ime ustreza digitalnemu certifikatu, ki je bil uporabljen pri podpisu sporočila. Nato še izvede postopek za preverjanje certifikata. Če je preverjanje neuspešno, pošiljatelju odgovori s kodo napake *Certificate*.
6. **Dešifriranje vsebine**. Prejemnik dešifrira vsebino sporočila in v primeru neuspeha odgovori s kodo napake *Decryption*.
7. **Preverjanje oblike vsebine**. Prejemnik z uporabo sheme, ki ustreza v glavi navedenemu tipu vsebine, preveri obliko vsebine. V primeru neuspeha odgovori s kodo napake *InValid*.
8. **Obdelava vsebine**. Prejemnik obdela vsebino v skladu s tipom vsebine.

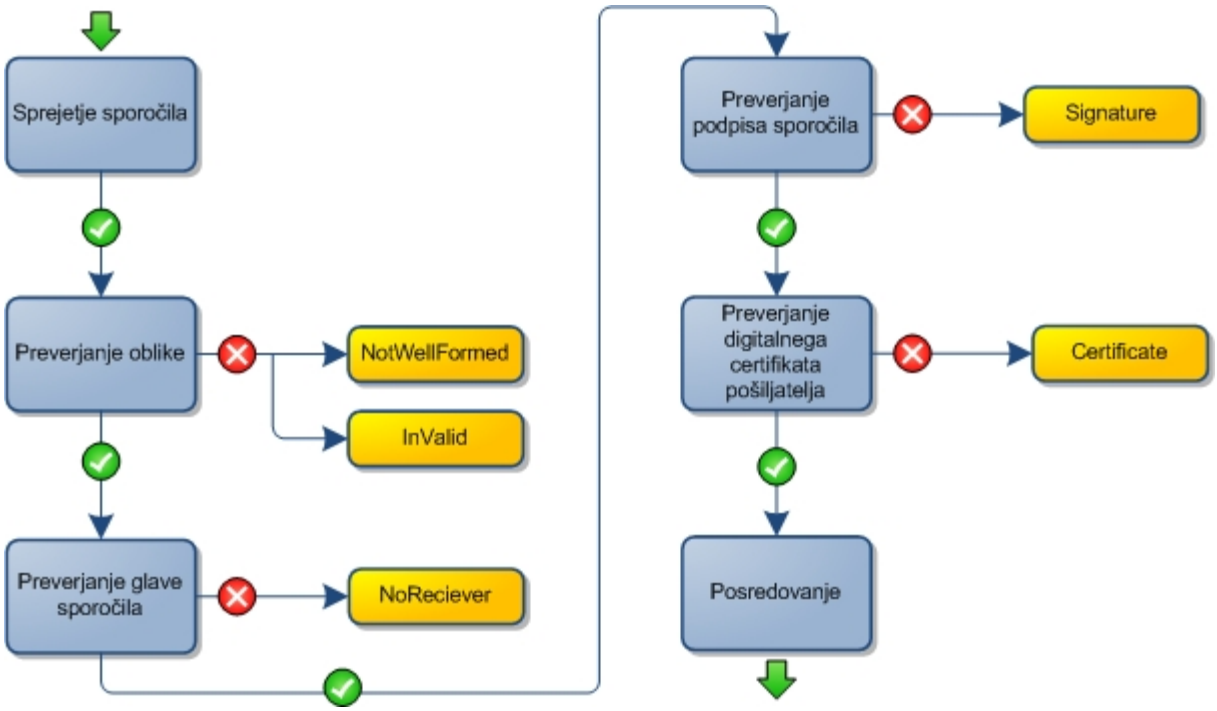


Slika 7: Sprejemanje sporočila

Posredovanje

Posredovanje sporočil opravlja komunikacijski strežnik. Ko sprejme sporočilo od določenega odjemalca, sporočilo posreduje naprej odjemalcu, ki je v glavi sporočila naveden kot prejemnik. Fazo posredovanja sporočila predstavljajo naslednji koraki:

1. **Sprejetje sporočila**. Strežnik sprejme sporočilo.
2. **Preverjanje oblike**. Sprejeto sporočilo najprej obdela XML razčlenjevalnik, ki nato opravi še preverjanje pravilnosti oblike s pomočjo XML shem. V primeru da oblika sporočila ne ustreza pravilom XML oblikovanja, strežnik pošiljatelju odgovori s kodo napake *NotWellFormed*. V primeru ko spodleti preverjanje pravilnosti oblike pa s kodo *InValid*.
3. **Preverjanje glave sporočila**. Strežnik preveri, ali je prejemnik sporočila prisoten. Če le-ta ni prisoten, strežnik odgovori s kodo napake *NoReceiver*.
4. **Preverjanje podpisa sporočila**. Strežnik preveri XML podpis sporočila in v primeru neuspeha odgovori s kodo napake *Signature*.
5. **Preverjanje digitalnega certifikata pošiljatelja**. Strežnik preveri, ali uporabniško ime ustreza digitalnemu certifikatu, ki je bil uporabljen pri podpisu sporočila. Nato še izvede postopek za preverjanje certifikata. Če je preverjanje neuspešno, pošiljatelju odgovori s kodo napake *Certificate*.
6. **Posredovanje.** Strežnik pošlje sporočilo v glavi navedenemu prejemniku.



Slika 8: Posredovanje sporočila

7.2. Postopki izmenjave sporočil

WebRes protokol predvideva izmenjavo informacij po modelu zahteva-odgovor. Zato je za vsako izmenjavo informacij predviden par sporočil, v katerem prvi predstavlja zahtevo (pripona RQ - *request*) in drugi odgovor na zahtevo (pripona RS - *response*). Izjema so sporočila, ki so namenjena vzpostavljanju povezave in poročanju o napakah. Postopki izmenjave sporočil v WebRes protokolu se izogibajo vodenju stanja, ki bi koordiniral ali

drugače povezoval sporočila. Razlog za to je, da je osnovni namen protokola omogočiti spletnemu portalu povpraševanje hotelskih rezervacijskih sistemov v realnem času. Ker je navadno s spletnim portalom hkrati v interakciji več uporabnikov, bi hranjenje stanja za vsakega od njih lahko pomenilo veliko obremenitev. Prav tako lahko uporabniki zapustijo spletni portal v kateremkoli trenutku, ne da bi celoten sistem o tem dobil obvestilo.

7.2.1. *Avtentikacija*

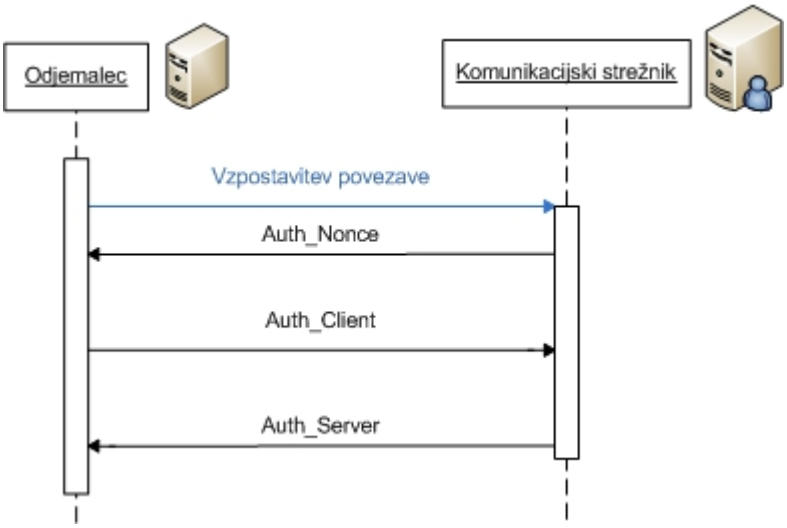
Postopek avtentikacije se izvrši med odjemalcem in komunikacijskim strežnikom. Namen postopka je omogočiti vzajemno avtentikacijo med komunikacijskim strežnikom in vmesnikom hotela. Tako se lahko komunikacijski strežnik prepriča, da sprejema povezavo od pravega vmesnika. Vmesnik pa se prepriča, da se priklaplja na želeni sistem. Uspešen postopek avtentikacje je predpogoj za izvajanje vseh ostalih postopkov izmenjave sporočil.

Pri tem postopku se izmenjajo tri sporočila v naslednjem zaporedju:

1. Ko odjemalec vzpostavi povezavo s strežnikom, mu strežnik pošlje sporočilo z vsebino tipa *Auth_Nonce*, ki vsebuje uporabniško ime strežnika in število avtentikacije (*nonce*), ki je za vsak postopek avtentikacije drugačno.
2. Odjemalec po sprejetju prejšnjega sporočila odgovori s sporočilom z vsebino tipa *Auth_Client*. S tem sporočilom odjemalec dokaže svojo identiteto strežniku. Sporočilo vsebuje uporabniško ime klienta, število avtentikacije (iz sprejetega sporočila) in podpis vsebine. Postopek za generiranje podpisa vsebine bo opisan v nadaljevanju.
3. Ko strežnik sprejme prejšnje sporočilo od odjemalca, najprej preveri, če je število avtentikacije enako poslanemu in če podpis vsebine ustreza uporabniškemu imenu odjemalca. Postopek preverjanja podpisa vsebine bo opisan v nadaljevanju. Če je preverjanje uspešno, strežnik doda odjemalca na seznam prijavljenih in odjemalcu pošlje še zadnje sporočilo z vsebino tipa *Auth_Server*. S tem sporočilom strežnik dokaže svojo identiteto odjemalcu. Sporočilo vsebuje uporabniško ime strežnika, za ena povečano vrednost števila avtentikacije in podpis vsebine.
4. Ko odjemalec sprejme prejšnje sporočilo od strežnika, najprej preveri, če je število avtentikacije povečano za ena in če je podpis vsebine pravilen. Če je preverjanje uspešno, je odjemalec uspešno povezan in lahko prične opravljati svojo nalogo pri nudenju storitve spletnih rezervacij.

```
<xs:schema  xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://www.hotel.si/webres/2008/08/authentication"
    xmlns:tns="http://www.hotel.si/webres/2008/08/authentication" elementFormDefault="qualified">
    <xs:element name="Authentication" type="tns:authenticationType"/>
    <xs:complexType name="authenticationType">
        <xs:sequence>
            <xs:element name="Nonce" type="tns:integer"/>
            <xs:element name="UserName" type="tns:string"/>
            <!-- podpis elementa Authentication nadomesti element Password -->
            <xs:choice minOccurs="0">
                <xs:element name="Password" type="tns:string"/>
                <xs:any namespace="http://www.w3.org/2000/09/xmldsig#" processContents="lax"/>
            </xs:choice>
        </xs:sequence>
    </xs:complexType>
</xs:schema>
```

Slika 9: XML shema elementa *Authentication*



Slika 10: Diagram zaporedja avtentikacije odjemalca

Vsebino sporočil avtentikacije predstavlja element *Authentication*, katerega shema je na sliki 9. Njegovo vsebino predstavljajo elementi *Nonce*, ki vsebuje število avtentikacije, *UserName*, ki vsebuje uporabniško ime ter elementa *Password* in *Signature*, ki sta si med seboj izključujoča. Postopek generiranja podpisa vsebine za avtentikacijo je naslednji:

1. V element *Nonce* zapiši število avtentikacije.
2. V element *UserName* zapiši svoje uporabniško ime.
3. V element *Password* zapiši geslo. V obeh primeri, ko sporočilo pošilja ali odjemalec ali strežnik, se zapiše geslo odjemalca.
4. Podpiši tako sestavljen element *Authentication* z XML podpisom.
5. V elementu *Authentication* nadomesti element *Password* z elementom *Signature*, ki je rezultat podpisa iz prejšnjega koraka.

Preverjanje podpisa vsebine:

1. Sprejeti vsebini dodaj med elementa *UserName* in *Signature* element *Password*, katerega vsebina je odjemalčevo geslo.
2. Preveri XML podpis v elementu *Signature*.

Na takšen način se geslo nikoli ne prenaša v čisti obliki, ampak je vključeno le preko XML podpisa. Gesla tako ni mogoče izluščiti, podpis pa lahko preverimo le, če geslo poznamo. Geslo in uporabniško ime za vmesnik hotel pridobi ob priključitvi na portal. Geslo za avtentikacijo v resnici ni potrebno, ker je zadosten dokaz že digitalni podpis. Z njim aplikacija dokaže lastništvo privatnega ključa digitalnega certifikata, s katerim se predstavlja. Gesla vključujejo le dodaten faktor zaščite.

7.2.2. *Rezervacija*

Postopek rezervacije omogoča spletnemu portalu nudenje storitev poizvedovanja po ponudbi in rezervaciji hotelskih namestitev v realnem času. Vsi ostali postopki so zgolj podpora temu postopku. Postopek sestavlja več korakov, v katerih si portal in hotel izmenjata informacije potrebne za izvedbo rezervacije. Vsak korak predstavljata par sporočil za zahtevo in odgovor, ki si jih spletni portal in vmesnik hotela izmenjata preko komunikacijskega strežnika.

Za uspešno izvršitev rezervacije mora spletni portal opraviti zaporedje naslednjih korakov:

0. Povpraševanje po razpoložljivosti hotela (opcijsko).
1. Povpraševanje po ponudbi hotela.
2. Izračun cene izbrane ponudbe.
3. Določitev garancije.
4. Potrditev rezervacije.

Za definiranje oblike sporočil v vseh korakih je bil uporabljen OpenTravel standard. Za obliko vsebine sporočil iz prvih treh korakov (0., 1. in 2.) je bil uporabljen par sporočil *OTA_HotelAvailRQ/RS*, ki je namenjen povpraševanju po razpoložljivosti in cenikih hotelskih produktov. V zadnjih dveh korakih (3. in 4.) pa je bil uporabljen par sporočil *OTA_HotelResRQ/RS*, ki je namenjen sklepanju rezervacij in pogojev za izvedbo rezervacije. Sporočila OpenTravel standarda omogočajo različno podrobno izmenjavo informacij, kar se opazi tudi v tem, da lahko le z dvema paroma sporočil pokrijemo več korakov. Bistvo WebRes protokola je, da vso poslovno logiko prepusti hotelskemu rezervacijskemu sistemu. Pri tem pa je portal namenjen le interakciji z uporabnikom. Zato je izbrano oblikovanje vsebine takšno, da zagotavlja izmenjavo minimalnega števila podatkov, ki so potrebni za krmiljenje poteka rezervacije na portalu. Na primer pri prikazu ponudbe zadostujejo zgolj opisi cenikov in ne podrobno razdelani plani cenikov.

Vsako izmed sporočil v teh korakih bo vsebovalo tudi informacije, ki se nanašajo na trenutno sejo uporabnika. Tem podatkom bomo rekli podatki o izvoru zahteve in so naslednji:

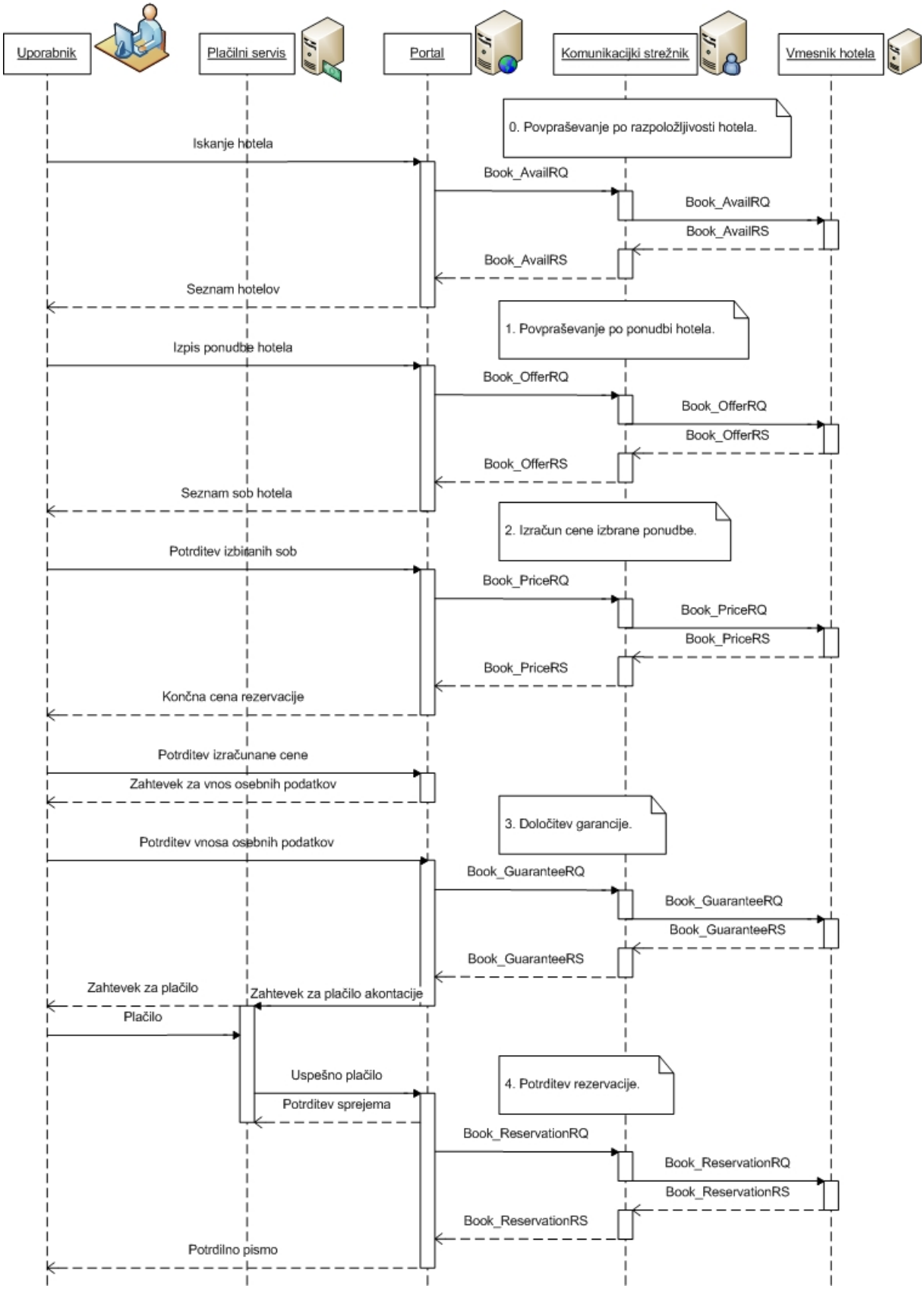
- IP naslov uporabnika;
- koda uporabnikovega jezika na portalu (sl – Slovenski, en – Angleški, de – Nemški in it – italijanski);
- enolični identifikator transakcije (uporabniku se dodeli v 1. koraku, v 0. koraku ni naveden);
- podatki o distribucijski točki (zaenkrat bo vedno navedeno le ime portala, parameter je namenjen morebitni širitvi sistema).

Povpraševanje po razpoložljivosti hotela

Ta korak ni obvezen za izvršitev rezervacije, lahko pa se ponovi večkrat. Portal opravi povpraševanje po razpoložljivosti hotela, ko uporabniku na spletni strani izpiše seznam hotelov, ki ustrezajo iskalnemu kriteriju. Rezultat izmenjave sporočil je pridobitev informacije o tem, ali hotel lahko uporabniku ponudi nastanitev glede na prametre, ki jih je le-ta navedel v iskalnem kriteriju.

Izmenjava sporočil:

1. Portal pošlje vmesniku hotela sporočilo z vsebino tipa *Book_AvailRQ*, v katerem navede parametre, ki jih je uporabnik izbral:
   - datum prihoda;
   - datum odhoda;
   - število odraslih oseb;
   - število otrok;
   - kodo hotela (služi vmesniku hotela za identifikacijo hotela v primeru, da ta predstavlja verigo hotelov).
2. Vmesnik hotela nato odgovori s sporočilom z vsebino tipa *Book_AvailRS*, v katerem pošlje kratek opis informativne cene nastanitve v hotelu. Če hotel nima prostih kapacitet v navedenem obdobju, ki bi ustrezale podanim prametrom, odgovori z elementom *Error* z atributoma *ShortText="NOROOMS"* in *Type="1"*.



Slika 11: Diagram zaporedja postopka rezervacije

Povpraševanje po ponudbi hotela

Ta korak je obvezen, ker lahko le na ta način portal pridobi seznam sob, ki jih hotel ponuja. Poizvedbo ponudbe lahko uporabnik zahteva večkrat, dokler ne najde ustrezne ponudbe. Portal opravi povpraševanje po ponudbi hotela, ko uporabnik od portala zahteva izpis ponudbe hotela za izbran kriterij. Rezultat povpraševanja je seznam sob, med katerimi lahko uporabnik izbira. V tem koraku portal uporabniku dodeli identifikator transakcije, ki določa celoten postopek ene rezervacije in omogoča sledenje le-tega. Identifikator transakcije preteče v naslednjih primerih:

- uporabnik uspešno zaključi rezervacijo;
- uporabnik prične povpraševanje po ponudbi drugega hotela;
- uporabnik predčasno prekine sejo spletnega portala.

Izmenjava sporočil:

1. Portal pošlje vmesniku hotela sporočilo z vsebino tipa *Book_OfferRQ*, v katerem navede parametre, ki jih je uporabnik izbral:
   - datum prihoda;
   - datum odhoda;
   - število odraslih oseb;
   - število otrok;
   - kodo hotela (služi vmesniku hotela za identifikacijo hotela v primeru, da ta predstavlja verigo hotelov).
2. Vmesnik hotela nato odgovori s sporočilom z vsebino tipa *Book_OfferRS*, v katerem pošlje seznam opisov sob, seznam opisov storitev in dodatna pojasnila, ki se nanašajo na ponudbo, cenike in odpovedne pogoje. Če hotel nima prostih kapacitet v navedenem obdobju, ki bi ustrezale podanim prametrom, odgovori z elementom *Error* z atributoma *ShortText="NOROOMS"* in *Type="1"* ter vsebino, ki se prikaže uporabniku.

Opis posamezne sobe vsebuje:

- identifikator sobe;
- kodo tipa sobe,
- kodo cenika;
- kodo hotela;
- naziv tipa sobe;
- opis tipa sobe;
- ime datoteke slike sobe;
- ceno (lahko tudi interval cene);
- opis cene oz. cenika;
- zaporedno številko storitve, ki jo vključuje soba;
- maksimalno število vseh oseb, ki jih sprejme soba,
- maksimalno število oseb posameznega starostnega razreda;
- maksimalno dovoljeno število izbranih sob tega tipa;
- obdobje, na katerega se ponudba nanaša;
- ali soba predstavlja alternativno ponudbo.

Portal za identifikacijo posamezne sobe uporablja identifikator sobe, kodo tipa sobe, kodo cenika in kodo hotela. S temi štirimi prametri se tudi v nadaljevanju sklicuje na vsako posamezno sobo.

To, da soba predstavlja alternativno ponudbo, pomeni, da obdobje ponudbe sobe ne ustreza datumom prihoda in odhoda, ki jih je izbral uporabnik. Na takšen način lahko vmesnik hotela ponudi tudi sobe, ki so na voljo v obdobju, ki za kakšen dan odstopa od izbranega.

Starostne razrede hoteli definirajo različno in na podlagi tega nudijo določene popuste. Vmesnik za vsako sobo definira maksimalno število oseb s tem, da navede *GuestCount* element brez informacije o starostnem razredu. Temu nato sledi eden ali več *GuestCount* elementov, ki definirajo maksimalno število gostov določenega starostnega razreda. Starostni razred navedejo z zgornjo mejo starosti ali pa s kodo starostnega razreda iz OpenTravel šifranta za starostne razrede. Pri tem mora vmesnik hotela poskrbeti za smiselno razčlenitev na starostne razrede. Portal na podlagi teh informacij lahko omeji število gostov za posamezno sobo in s tem zmanjša vnos napačnega števila oseb.

Primer navedbe maksimalnega števila oseb sobe:

```
<GuestCounts>
   <GuestCount Count="3"/> ← število vseh oseb
   <GuestCount AgeQualifyingCode="10" Count="2"/> ← število odraslih oseb
   <GuestCount Age="13" Count="2"/> ← število oseb starih največ 13 let
</GuestCounts>
```

Opis storitve vsebuje:

- zaporedno številko;
- naziv storitve;
- opis storitve.

Dodatna pojasnila v zvezi s ponudbo, ceniki in odpovednimi pogoji vmesnik hotela navede v elementu *Warning* z atributoma *ShortText="DETAILS"* in *Type="1"* ter vsebino, ki se na portalu prikaže pod seznamom ponudbe.

Slika 12: Prikaz ponudbe sob na spletnem portalu

Izračun cene izbrane ponudbe

Ko uporabnik potrdi vnos števila sob in oseb, portal pošlje seznam izbranih sob vmesniku hotela. Vmesnik hotela na podlagi izbranega izračuna končne cene rezervacij sob in jih pošlje portalu. Portal nato ceno izbranih sob prikaže uporabniku. Korak izračuna cene lahko uporabnik zahteva večkrat, ko primerja različne ponudbe.

Izmenjava sporočil:

1. Portal pošlje vmesniku hotela sporočilo z vsebino tipa *Book_PriceRQ*, v katerem navede seznam izbranih sob. Pri tem za vsako sobo navede:
   - identifikator sobe;
   - kodo tipa sobe;
   - kodo cenika;

- kodo hotela;
- obdobje rezervacije sobe;
- količino izbranih sob tega tipa;
- seznam števila oseb po starostnih razredih.

2. Vmesnik hotela nato odgovori s sporočilom z vsebino tipa *Book_PriceRS*, v katerem pošlje seznam izbranih sob. Na seznamu vsako sobo opiše s podatki, ki so mu bili poslani v zahtevi in končno ceno. Če vmesnik ugotovi nepravilnost, odgovori z elementom *Error* z atributom *Type="1"* in vsebino, ki se prikaže uporabniku. Atribut *ShortText* pri tem pojasnjuje napako z eno izmed naslednjih vrednosti:
   - *NOROOMS* – izbrane sobe niso več na voljo;
   - *ROOMCOUNT* – napaka je povezana s številom sob;
   - *GUESTCOUNT* – napaka je povezana s številom oseb;
   - *OTHER* – nobeno od prejšnjih vrednosti ne ustreza vrsti napake.

V korakih v nadaljevanju mora vmesnik vedno preveriti, če so izbrane sobe še vedno na voljo. Lahko se namreč zgodi, da je kakšen drug uporabnik med tem opravil rezervacijo in s tem zasedel del izbranih sob. Hotel vmesnika sobe rezervira šele na koncu rezervacije, saj uporabnik lahko večkrat ponovi izbiro sob. V tem koraku in v korakih v nadaljevanju lahko vmesnik hotela z uporabo katerega od naštetih pojasnil napake zahteva povrnitev uporabnika na 1. korak postopka rezervacije.

Določitev garancije

Ko uporabnik na portalu potrdi izračunano ceno iz prejšnjega koraka, portal od njega zahteva vnos osebnih podatkov. Po vnosu osebnih podatkov portal vmesniku hotela pošlje osebne podatke in seznam izbranih sob. Vmesnik hotela na podlagi sprejetih podatkov lahko določi enega izmed naslednjih načinov garancije:

- plačilo akontacije – uporabnik plača vnaprej del cene rezervacije;
- podatke kreditne kartice – uporabnik vnese podatke kreditne kartice, ki jo lahko nato hotel bremeni ob morebitni nepravočasni odpovedi rezervacije;
- garancija ni potrebna.

V večini primerov zadostujejo podatki kreditne kartice. Plačilo akontacije pride v poštev v primerih rezervacije sob za daljše obdobje in za visoko ceno. Izključitev garancije pa lahko zahteva vmesnik v primeru, ko na podlagi osebnih podatkov uporabnika ugotovi, da gre za rednega gosta. V primeru, ko vmesnik hotela zahteva akontacijo, mora tudi že rezervirati izbrane sobe in portalu sporočiti referenco te rezervacije. Če vmesnik hotela nebi rezerviral izbranih sob, lahko po izvedbi plačila akontacije ne bi bile več na voljo.

Izmenjava sporočil:

1. Portal pošlje vmesniku hotela sporočilo z vsebino tipa *Book_GuaranteeRQ*, v katerem navede seznam sob in osebne podatke uporabnika. Seznam sob vsebuje enake podatke kot v zahtevi za določitev cene. Osebni podatki gosta pa so naslednji:
   - ime;
   - priimek,
   - telefon;
   - faks (neobvezen podatek);
   - elektronska pošta;

- ulica;
- poštna številka;
- mesto;
- federalna enota (neobvezen podatek);
- država;
- uporabniško ime (v primeru, ko je uporabnik registriran uporabnik portala).

2. Vmesnik hotela nato odgovori s sporočilom z vsebino tipa *Book_GuaranteeRS*, v katerem navede način garancije. Če je zahtevano plačilo akontacije, navede tudi višino zneska in referenčno številko rezervacije, s katero zadrži izbrane sobe. Če vmesnik hotela ugotovi, da izbrane sobe niso več na voljo, odgovori z elementom *Error* z atributoma *Type="1"* in *ShortText="NOROOMS",* ter vsebino, ki se prikaže uporabniku. V primeru, da je ugotovljena nepravilnost osebnih podatkov, vmesnik hotela odgovori z napako *ShortText="PROFILE"* in opisom, ki se prikaže uporabniku.

Vmesnik hotela v sporočilu navede način garancije z elementom *GuaranteePayment*.

Podatki kreditne kartice

```
<GuaranteePayment
   GuaranteeType="CC/DC/Voucher">
   <AcceptedPayments>
      <AcceptedPayment>
        <PaymentCard CardType="1"/>
      </AcceptedPayment>
   </AcceptedPayments>
</GuaranteePayment>
```

Plačilo akontacije

```
<GuaranteePayment
   GuaranteeType="Deposit">
   <AmountPercent
       Amount="100"
       CurrencyCode="EUR"/>
</GuaranteePayment>
```

Garancija ni potrebna

```
<GuaranteePayment GuaranteeType="None"/>
```

Potrditev rezervacije

Ko portal opravi postopke, ki jih zahteva izbran način garancije, pošlje vmesniku hotela še zadnje sporočilo, s katerim zaključi rezervacijo. Ta korak se v postopku rezervacije izvede le enkrat. Vmesnik hotela v sporočilu, ki predstavlja uspešno zaključeno rezervacijo, pošlje zahvalo hotela in sliko potrdilnega pisma. Zahvalo hotela portal pošlje uporabniku po elektronski pošti, ki ji pripne tudi sliko potrdilnega pisma. Potrdilno pismo se prikaže tudi uporabniku na portalu ob zaključku rezervacije. V primeru neuspelega plačila akontacije pošlje portal vmesniku hotela preklic rezervacije, s katero so bile zadržane sobe v koraku določanja garancije.

Izmenjava sporočil:

1. Portal pošlje vmesniku hotela sporočilo z vsebino tipa *Book_ReservationRQ*, katerega vsebina ima lahko štiri različne oblike glede na to, kakšen način garancije je bil zahtevan:
   a. Če garancija ni bila zahtevana, sporočilo vsebuje le seznam izbranih sob in osebne podatke uporabnika. Korenski element vsebine vsebuje atribut *ResStatus="Commit"*, ki nakazuje potrditev rezervacije.
   b. Če so bili zahtevani podatki kreditne kartice, sporočilo vsebuje seznam izbranih sob, osebne podatke uporabnika in podatke o kreditni kartici (ime lastnika, številko kartice, tip kartice in datum veljavnosti). Korenski element vsebine vsebuje atribut *ResStatus="Commit"*, ki nakazuje potrditev rezervacije.

c. Če je bilo zahtevano plačilo akontacije in je bilo plačilo uspešno izvedeno, sporočilo vsebuje seznam izbranih sob, osebne podatke uporabnika, informacije o plačilu akontacije in referenčno številko rezervacije, s katero je vmesnik pri določitvi garancije zadržal izbrane sobe. Korenski element vsebine vsebuje atribut *ResStatus="Commit"*, ki nakazuje potrditev rezervacije.
d. Če je bilo zahtevano plačilo akontacije in plačilo ni bilo izvedeno, sporočilo vsebuje le referenčno številko rezervacije, s katero je vmesnik pri določitvi garancije zadržal izbrane sobe. Korenski element vsebine vsebuje atribut *ResStatus="Cancel"*, ki nakazuje preklic rezervacije.

2. Vmesnik hotela odgovori s sporočilom z vsebino tipa *Book_ReservationRQ*, katerega vsebina ima lahko tri različne oblike:
   a. Če je zahteva vsebovala potrditev rezervacije in je bila rezervacija uspešno zabeležena, sporočilo vsebuje zahvalo hotela in sliko potrdilnega pisma. Korenski element vsebine vsebuje atribut *ResResponseType="Committed"*, ki nakazuje uspešno zabeleženje rezervacije.
   b. Če je zahteva vsebovala preklic rezervacije in je bila rezervacija uspešno preklicana, sporočilo vsebuje le informacijo o uspehu operacije. Korenski element vsebine vsebuje atribut *ResResponseType="Cancelled"*, ki nakazuje preklicano rezervacijo.
   c. Če vmesnik hotela ugotovi, da izbrane sobe niso več na voljo, odgovori z elementom *Error* z atributoma *Type="1"* in *ShortText="NOROOMS"* ter vsebino, ki se prikaže uporabniku. Korenski element vsebine vsebuje atribut *ResResponseType="Unsuccessful"*, ki nakazuje neuspeh zahteve. V primeru plačila akontacije se to ne more zgoditi, saj so sobe že zadržane.

Portal vmesniku pridobitev zahtevane garancije sporoči tako, da znotraj elementa *ResGlobalInfo* navede eno izmed naslednjih dveh vsebin:

Podatki kreditne kartice

```
<Guarantee>
  <GuaranteesAccepted>
    <GuaranteeAccepted>
      <PaymentCard CardCode="VS"
           CardNumber="4111111111111202"
           ExpireDate="0509">
        <CardHolderName>
           Janez Novak
        </CardHolderName>
      </PaymentCard>
    </GuaranteeAccepted>
  </GuaranteesAccepted>
</Guarantee>
```

Plačilo akontacije

```
<GuaranteePayment
     GuaranteeType="Deposit">
     <AmountPercent
       Amount="100"
       CurrencyCode="EUR"/>
</GuaranteePayment>
```

Zadnji korak rezervacije je občutljiv na izpad posamezne komponente sistema. Predvsem je potrebno poskrbeti, da se zaključijo rezervacije s plačano akontacijo. Rezervacije, ki niso zahtevale plačila, ne predstavljajo izgubljenega denarja, zato jih uporabnik lahko ponovno opravi, ko sistem ponovno deluje. Ob izpadu komponent sistema so predvideni naslednji scenariji:

- V primeru, da je bilo izvedeno plačilo akontacije in vmesnik hotela ni več na voljo, portal zabeleži plačilo in referenco rezervacije zadržanih sob. Rezervacijo nato potrdi pomožni odjemalec v trenutku, ko se vmesnik hotela prvič po izpadu prijavi v komunikacijski strežnik. Uporabnik dobi potrdilo rezervacije naknadno po

elektronski pošti. Če plačilo ni bilo izvedeno, pomožni odjemalec le prekliče rezervacijo zadržanih sob.

- V primeru, da uporabnik ni več prisoten, ko portal od plačilnega servisa sprejme potrditev plačila, bo portal potrdil rezervacijo in uporabniku poslal potrditev rezervacije po elektronski pošti.
- V primeru, da pride do izpada portala med čakanjem potrditve plačila, plačilni servis ne bo mogel uspešno sporočiti potrditve plačila. Plačilni servisi v tem primeru stalno poskušajo sporočiti potrditev plačila, dokler ne dobijo potrditve, da ga je ciljni sistem sprejel. Ko bo portal ponovno začel delovati in bo obveščen o plačilu, bo opravil rezervacijo na enak način, kot bi jo v primeru neprisotnosti uporabnika.

7.2.3. *Posodabljanje statične vsebine*

Posodabljanje statične vsebine omogoča vmesniku hotela, da zamenja slikovni material hotelskih sob. Ker se slike sob redkeje spreminjajo kot njihovi opisi, bi bilo prenašanje slik skupaj z vsako ponudbo sob nesmiselno. Pri tem postopku preko komunikacijskega strežnika komunicirata med seboj vmesnik hotela in pomožni odjemalec portala.

Za obliko vsebine sporočil se pri tem postopku uporablja par sporočil OTA_HotelDescriptiveContentNotifRQ/RS iz OpenTravel sintakse. S tem parom sporočil hoteli lahko zahtevajo posodobitev lastnih podatkov v drugih sistemih (npr. sistemu potovalne agencije). WebRes protokol trenutno uporablja par sporočil le za prenos slik sob v Base64 predstavitvi. Datoteka slike se pri tem v znakovni predstavitvi vključi v sporočilo kot element opisa slike. Z uporabo te vrste sporočila omogočimo kasnejšo razširitev postopka na posodabljanje tudi drugih podatkov o hotelu (npr. posodabljanje opisa hotela, ki se trenutno posodablja ročno preko spleta).

Izmenjava sporočil:

1. Vmesnik hotela pošlje pomožnemu odjemalcu sporočilo z vsebino tipa *Update_StaticDataRQ*, ki vsebuje seznam slik. Vsaka slika je predstavljena z imenom datoteke in Base64 predstavitvijo datoteke slike.
2. Pomožni odjemalec odgovori s sporočilom z vsebino tipa *Update_StaticDataRS*, ki potrjuje uspešen prenos slik. Ob morebitni napaki odgovori z elementom *Error* z atributoma *Type="1"* in opisom napake. Primer napake bi bile lahko neustrezne dimenzije slik.

7.2.4. *Preverjanje prisotnosti odjemalcev*

Sporočilo za preverjanje prisotnosti uporablja komunikacijski strežnik. Z njim preverja aktivnost priklopljenih odjemalcev. Če odjemalec ne odgovori v predvidenem času, ga strežnik odstrani iz seznama povezanih odjemalcev. Pri tem se uporablja par generičnih sporočil OTA_PingRQ/RS iz OpenTravel sintakse.

Izmenjava sporočil:

1. Komunikacijski strežnik pošlje odjemalcu sporočilo z vsebino tipa *Ping_RQ*.
2. Odjemalec odgovori s sporočilom z vsebino tipa *Ping_RS*.

7.3. Vpliv komunikacije na komponente sistema

Komunikacijski strežnik, spletni portal, pomožni odjemalec in vmesniki hotelov predstavljajo komponente spletnega rezervacijskega sistema, ki implementirajo WebRes

protokol. Vsaka od naštetih komponent mora pravilno realizirati postopke, ki jih protokol od nje zahteva. Le tako bo zagotovljena varnost in učinkovitost izmenjave podatkov.

7.3.1. *Priprava sporočila*

Komponente morajo sporočilo najprej sestaviti in šele potem ga lahko odpošljejo. Obliko vsakega dela sporočila v WebRes protokolu definira XML shema. Zato je potrebno poskrbeti, da so sporočila oblikovana tako, kot narekujejo XML sheme. Posamezno sporočilo v programskem jeziku predstavlja niz (*string*), v katerem je zapisan XML dokument. XML dokument lahko v objektno usmerjenih programskih jezikih predstavimo tudi kot hierarhično strukturo objektov, ki ustreza hierarhiji elementov. Tak pristop pri programiranju postopkov za izgradnjo sporočil vsiljuje strukturo posameznega XML dokumenta, s čimer se zmanjša možnost izgradnje nepravilnega dokumenta. Za pretvorbo objektov oz. strukture objektov v niz, ki predstavlja XML dokument, se uporablja postopek serializacije in ga implementira večina programskih knjižnic današnjih programskih jezikov (npr.: Java, .NET, Perl, PHP idr.). Deserializacija XML dokumenta pa predstavlja inverzen postopek, ki pretvori XML dokument v objekt oz. strukturo objektov. Izgradnjo struktur objektov pa nam olajšajo orodja, ki omogočajo pretvorbo XML shem v strukture razredov izbranega programskega jezika. Sporočilo je tako na koncu sestavljeno iz objektov, ki so instance razredov, pridobljenih iz XML shem.

XML sheme igrajo pomembno vlogo tudi pri preverjanju sprejetih sporočil. XML sheme uporabljenih standardov in tiste, ki so specifične za WebRes protokol, so dostopne preko spleta. Zato, da bodo aplikacije lahko hitreje dostopale do shem, je priporočljivo, da se le-te hranijo lokalno. Predvsem to velja za OpentTravel sheme, ki presegajo velikost datoteke 100kB.

7.3.2. *Sprejemanje in pošiljanje sporočila*

Sporočila WebRes protokola se prenašajo s pomočjo TCP protokola. TCP prenaša pakete bajtov in se ne zaveda sporočila. Zato mora aplikacija, ki iz TCP povezave zajema bajte oz. znake, poskrbeti, da sprejme sporočilo v celoti. Pri testiranjih zaradi neobremenjenega omrežja pogosto ne opazimo te lastnosti TCP povezave in sporočilo vsakokrat sprejmemo v celoti kot en niz. V realnih pogojih se ob obremenitvah omrežja pogosto zgodi, da sprejmemo sporočilo v več delih. V takem primeru mora aplikacija počakati, da sprejme še preostale dele sporočila in jih združiti v en niz. Problem lahko rešimo na dva načina. Pri prvem načinu mora pošiljatelj sprejemniku vnaprej sporočiti dolžino sporočila z obvestilom. Obvestilo je sporočilo dogovorjene fiksne dolžine, zato sprejemnik lahko ugotovi njegov konec. Konec sporočila, ki sledi obvestilu, pa sprejemnik ugotovi na podlagi dolžine, ki je zapisana v obvestilu. Drug način predvideva uporabo posebne sekvence znakov, ki označuje konec sporočila. Sprejemnik tako ugotovi konec sporočila, ko sprejme sekvenco, ki označuje konec sporočila. Drugi način je bolj enostaven, vendar postavlja omejitve glede vsebine sporočila. Vsebina sedaj ne sme vsebovati sekvence znakov, ki označuje konec sporočila.

Pri WebRes protokolu nismo definirali pošiljanja dolžine sporočila vnaprej. Zato bomo izbrali za označevanje konca sporočila, kar niz "*</env:Envelope>*", ki tudi končuje vsebino korenskega elementa sporočila. Omejitve, ki smo jih postavili na vsebino sporočila, so naslednje:

1. Predpona za imenski prostor SOAP ovojnice mora biti vedno *env*.
2. Noben blok glave ali telesa SOAP ovojnice ne sme vsebovati elementa *Envelope* s predpono *env*. SOAP prepoveduje uporabo imenskega prostora ovojnice znotraj blokov. Vendar lahko *env* predpono ponovno definiramo za drug imenski prostor

(imenske prostore definirajo URI naslovi) in s tem ne kršimo omejitev SOAP sheme.

3. Vsebina elementov znotraj CDATA sekcij ne sme vsebovati niza "*</env:Envelope>*". Zunaj CDATA sekcij se sme pojaviti kot tekstovna vsebina le v primeru, če za XML oznake uporabimo ubežne sekvence, kar pa pomeni, da kot niz ne ustreza več enakemu zaporedju znakov.

Ker *WebRes* protokol ne predvideva vsebine, ki bi vsebovala niz "*</env:Envelope>*", lahko brez problema uporabimo ta niz za prepoznavo konca sporočila. Ker pa imajo sporočila XML obliko, lahko tudi zgornje omejitve odpravimo z enostavno logiko za delno razčlenjevanje XML dokumenta:

1. Pri sprejemanju sporočila lahko razberemo uporabljeno predpono iz prvega elementa tako, da v prvi znački poiščemo deklaracijo imenskega prostora SOAP ovojnice. Nato poiščemo konec sporočila z uporabo najdene predpone.
2. Ker morajo značke nastopati v parih, moramo pri sprejemanju spremljati le pravilnost gnezdenja elementov z imenom *env:Envelope*.
3. Tekst med začetkom in koncem CDATA sekcij ignoriramo.

Izmenjavo sporočil med portalom in vmesniki hotela opravlja komunikacijski strežnik. Portal lahko v enem trenutku predstavlja več niti, ki obdelujejo zahteve različnih uporabnikov. To pomeni, da lahko v kateremkoli trenutku komunikacijski strežnik od portala sprejme več sporočil, ki jih mora posredovati enemu ali več vmesnikom. V aplikaciji komunikacijskega strežnika teče za vsakega prijavljenega odjemalca ena nit. Pri posredovanju sporočila nit pošiljatelja sprejme sporočilo in ga pošlje naprej po povezavi, ki pripada prejemniku. Pri tem mora aplikacija komunikacijskega strežnika poskrbeti za sinhronizacijo med nitmi odjemalcev z uporabo sinhronizacijskih konstruktov (npr. semaforji). S stališča implementacije WebRes protokola je potrebno poskrbeti za naslednje dva problema:

1. Hkrati se ne sme pošiljati več kot eno sporočilo enemu odjemalcu. Če bi hkrati po eni povezavi pošiljalo več niti, bi se sporočila pomešala. Zato mora nit, ki posreduje sporočilo, najprej zavzeti povezavo odjemalca in jo zadržati za čas pošiljanja sporočila.
2. Če odjemalec ni sposoben obdelovati več sporočil hkrati, mora nit, ki posreduje sporočilo, najprej zavzeti povezavo in jo zadržati za toliko časa, dokler odjemalec ne pošlje odgovora.

Prvi problem je splošen, drugi pa se navezuje na način implementacije posameznega odjemalca. Zaradi drugega problema mora komunikacijski strežnik voditi evidenco o tem, katerim odjemalcem se lahko posreduje več sporočil zaporedoma. Obdelava več sporočil hkrati je zaželena lastnost vmesnika hotela. Tako lahko več uporabnikov preko portala hkrati povprašuje po ponudbi enega hotela, ne da bi njihove zahteve v ozadju čakale.

7.3.3. *Varnost podatkov*

Digitalni certifikati predstavljajo primarni vir ključev za javno kriptografijo pri zaščiti komunikacije z uporabo WebRes protokola. Prav tako se digitalni certifikati uporabljajo za preverjanje identitete posamezne komponente. Zato je potrebno vzpostaviti vez med digitalnim certifikatom in uporabniškim imenom tako odjemalca kot tudi strežnika. Komunikacijski strežnik mora zato voditi evidenco o tem, kateri digitalni certifikat uporablja kateri odjemalec. Le tako lahko omeji odjemalce, da se avtenticirajo vedno z istim digitalnim certifikatom. Odjemalci lahko vzdržujejo le del evidence, ki se nanaša le na odjemalce s katerimi komunicirajo. Ker bodo portal, pomožni odjemalec in komunikacijski strežnik

uporabljali isti digitalni certifikat, morajo vmesniki hotelov imeti le informacijo o tem, kateri digitalni certifikat uporabljajo. Na drugi strani pa mora imeti komunikacijski strežnik v podatkovni bazi zabeleženo, kateri digitalni certifikat uporablja posamezen vmesnik hotela. Portal in pomožni odjemalec pa lahko do teh informacij dostopata, ker uporabljata isto podatkovno bazo kot komunikacijski strežnik.

Za identifikacijo digitalnega certifikata se uporablja nabor parametrov, ki je enak naboru parametrov, ki ga za identifikacijo certifikata uporablja OCSP protokol. Ta za identifikacijo certifikata uporablja [9]:

- Ime CA, ki je podpisala in izdala digitalni certifikat.
- Javni ključ CA, ki je podpisala in izdala digitalni certifikat.
- Serijsko številko digitalnega certifikata.

OCSP predvideva pošiljanje prvih dveh parametrov v zgoščeni obliki, zato se kot četrti prameter pošlje še algoritem za izračun zgoščevalne funkcije. Poleg imena CA se uporablja še njen javni ključ, ker ni mogoče postaviti globalne omejitve, ki bi preprečevala dvema CA uporabo istega imena. Isti javni ključ pa bosta dve CA uporabljali le v primeru, da se o tem dogovorita, v nasprotnem primeru se smatra to kot ogrozitev privatnega ključa.

Za izmenjavo digitalnih certifikatov poskrbi XML podpis, pri katerem lahko podpisnik v element *KeyInfo* vključi lastni digitalni certifikat. Izmenjavo digitalnih certifikatov opravita komunikacijski strežnik in vmesnik hotela pri postopku avtentikacije.

WebRes protokol varuje podatke le med prenosom. Za varno upravljanje s podatki pa mora biti poskrbljeno tudi v ostalih ozirih sistema. Pri izvajanju postopka rezervacije se izmenja med komponentami sistema veliko podatkov zaupne narave. Ti podatki se v določenih fazah tudi shranijo. Zato je potrebno poskrbeti, da do njih lahko dostopajo le pooblaščene osebe. Posebno pozornost je potrebno posvetiti beleženju sporočil (*logging*), ki ga izvajajo posamezne komponente. Sporočila se beležijo največkrat z razlogom, da se napake v delovanju lažje odkrijejo. Sporočila lahko vsebujejo zaupne podatke, zato je najbolje, da se le-ta hranijo v šifrirani obliki. XML šifriranje v sporočilo vključi vse podatke, ki so potrebni za dešifriranje, zato ob kasnejši analizi sporočil dešifriranje ne bo predstavljalo ovire.

Problem predstavlja tudi hranjenje privatnih ključev digitalnih certifikatov. Velja pravilo, da mora biti zaščita privatnega ključa vsaj toliko močna, kot je zaščita sporočil, ki jih ščiti ta privatni ključ. Zato privatnih ključev nikoli ne smemo hraniti v čisti obliki. Najenostavnejša zaščita je simetrično šifriranje privatnega ključa z uporabo gesla, vendar moramo pri tem uporabiti dovolj močna gesla. Varno hranjenje privatnega ključa, zaščitenega z geslom, ponujajo tudi nekateri operacijski sistemi (Microsoft Windows podpira varno hranjenje ključev od različice 2000 dalje). Za hrambo privatnih ključev obstajajo tudi posebne naprave, ki preprečujejo, da bi iz njih pridobili ključ, ko ga enkrat vanje shranimo. Te naprave omogočajo nato le uporabo ključa tako, da same opravljajo osnovne kriptografske operacije namesto računalnika s katerim so povezane. Primer takih naprav so kriptografske pametne kartice (c*ryptographic smart cards*).

8. Sklepne ugotovitve

Elektronsko poslovanje omogoča sklepanje poslov preko interneta brez posrednika. Posameznim podjetjem se v večini primerov ne izplača vzpostaviti lastne rešitve za spletno distribucijo. Zato dostikrat posežejo po uporabi storitev drugih, ki za njihovo ponudbo ponujajo obliko spletne prodaje. Kar pa jim v večini primerov onemogoči direktno interakcijo s kupcem, ki bi jo imeli z lastnim sistemom.

S portalom hotel.si želimo hotelom ponuditi storitev, ki jim omogoča prisotnost na skupni spletni distribucijski točki z možnostjo direktne interakcije z bodočim gostom. S tem hotel dobi dostop do infrastrukture za sklepanje poslov z internetnim uporabnikom brez vmesnega posrednika. Portal preko WebRes protokola ponuja hotelskemu sistemu popoln nadzor nad procesi rezervacij in prikazom ponudbe, ki jih preko spleta uporabljajo gosti. Rezervacija v realnem času je prednost, ki je namenjena predvsem gostom, ki se s tem izognejo neprijetnostim zakasnjene potrditve.

Za definiranje WebRes protokola so bili uporabljeni XML standard in iz njega izpeljani standardi. Pri tem je uporaba standardov z isto osnovo omogočila bolj kompaktno realizacijo protokola. Namreč, vsi postopki, ki jih definirajo standardi, so namenjeni obdelavi podatkov predstavljenih v XML obliki. To pa je omogočilo enostavno združevanje le-teh pri definiranju postopkov za obdelavo sporočil protokola.

Pri zasnovi protokola je bilo potrebno rešiti številne probleme, ki se pojavijo pri prenašanju podatkov. Zunanjo predstavitev sporočil rešuje uporaba SOAP standarda. Poleg oblike poda SOAP tudi postopke za obdelavo in prenašanje sporočil. Vsebino sporočil predstavljajo sporočila OpenTravel standarda, ki definirajo strukturo podatkov, ki so potrebni za izmenjavo informacij. Sintakso za specifične dele sporočil protokola pa definirajo lastne XML sheme. Zaščito podatkov v sporočilih protokol rešuje z uporabo kriptografskih postopkov, ki jih vključi s pomočjo XML šifriranja in XML podpisovanja. Oba standarda odstranita potrebo po dogovarjanju o uporabljenih ključih in algoritmih, ker informacijo o teh na sistematičen način priložita rezultatu. Na enak način nudita tudi izmenjavo digitalnih certifikatov, ki se uporabljajo pri podpisovanju in šifriranju sporočil. Pri tem digitalni certifikati v povezavi z infrastrukturo javnih ključev omogočajo preverjanje identitete pošiljatelja sporočila.

Protokol je bil razvit z namenom, da omogoči postopek rezervacije hotelskih sob. To je hkrati zahtevalo tudi identifikacijo podatkov, katerih izmenjava je potrebna za uspešno izvedbo tega postopka. Iskanje ustrezne oblike sporočil OpenTravel standarda se je izkazalo za dokaj zamudno delo. Sporočila so namreč namenjena precej široki uporabi, zato so obsežne tudi XML sheme, ki definirajo njihovo sintakso. Pregledovanje obsežnih shem kljub orodjem za grafični prikaz shem ni enostavno. Uporaba standarda sintakse se bo izkazala za priročno v prihodnosti, ko bo morebitna potrebna širitev protokola zahtevala tako dodajanje novih parametrov v obstoječa sporočila kot tudi dodajanje novih sporočil.

Protokol kot tak ni strogo vezan na sporazumevanje med portalom in vmesnikom hotela. Brez problema bi na komunikacijski strežnik priklopili še kak drug sistem, ki bi lahko sklepal rezervacije z uporabo protokola. Tako bi na primer lahko dodali posebnega odjemalca, ki bi omogočal rezervacijo in pregled ponudbe potovalnim agencijam. Z razširitvijo sporočil bi bil protokol lahko primeren tudi za direktno prodajo drugih hotelskih produktov in storitev. To pa bi lahko bila tudi naslednja razširitev protokola, s katero bi lahko hotel ponujal dodatne storitve, ki niso neposredno vezane samo na hotelsko namestitev.

9. Priloge

9.1. Primer sporočil postopka rezervacije

9.1.1. *Povpraševanje po razpoložljivosti hotela*

Zahteva

```
<OTA_HotelAvailRQ xmlns="http://www.opentravel.org/OTA/2003/05" Version="2.003">
   <POS>
      <Source TerminalID="127.0.0.1" ISOCountry="si">
         <BookingChannel Type="7">
            <CompanyName>hotel.si</CompanyName>
         </BookingChannel>
      </Source>
   </POS>
   <AvailRequestSegments>
      <AvailRequestSegment>
         <StayDateRange End="2008-12-05" Start="2008-11-30"/>
         <RoomStayCandidates>
            <RoomStayCandidate>
               <GuestCounts>
                  <GuestCount AgeQualifyingCode="10" Count="2"/>
                  <GuestCount AgeQualifyingCode="8" Count="1"/>
               </GuestCounts>
            </RoomStayCandidate>
         </RoomStayCandidates>
         <HotelSearchCriteria>
            <Criterion>
               <HotelRef HotelCode="HFRI"/>
            </Criterion>
         </HotelSearchCriteria>
      </AvailRequestSegment>
   </AvailRequestSegments>
</OTA_HotelAvailRQ>
```

IP uporabnika.

Koda države se nanaša na izbran jezik.

Podatki o distribucijski točki (7 - internet).

Datuma prihod in odhoda.

Število oseb (10 – odrasli, 8 – otroci).

Koda hotela.

Odgovor

```
<OTA_HotelAvailRS xmlns="http://www.opentravel.org/OTA/2003/05" Version="5.000">
   <POS>
      <Source ISOCountry="si"/>
   </POS>
   <Success/>
   <RoomStays>
      <RoomStay>
         <RatePlans>
            <RatePlan>
               <RatePlanDescription>
                  <Text>
                     40 - 70 EUR za sobo na dan
                  </Text>
               </RatePlanDescription>
            </RatePlan>
         </RatePlans>
      </RoomStay>
   </RoomStays>
</OTA_HotelAvailRS>
```

Koda države se nanaša na izbran jezik.

Informativna cena za namestitev v hotelu.

9.1.2. *Povpraševanje po ponudbi hotela*

Zahteva

```
<OTA_HotelAvailRQ xmlns="http://www.opentravel.org/OTA/2003/05" Version="2.003">
   <POS>
      <Source ERSP_UserID="465465" TerminalID="127.0.0.1" ISOCountry="si">
         <BookingChannel Type="7">
            <CompanyName>hotel.si</CompanyName>
         </BookingChannel>
      </Source>
   </POS>
   <AvailRequestSegments>
      <AvailRequestSegment>
         <StayDateRange End="2008-12-05" Start="2008-11-30"/>
         <RoomStayCandidates>
            <RoomStayCandidate>
               <GuestCounts>
                  <GuestCount AgeQualifyingCode="10" Count="2"/>
                  <GuestCount AgeQualifyingCode="8" Count="1"/>
               </GuestCounts>
            </RoomStayCandidate>
         </RoomStayCandidates>
         <HotelSearchCriteria>
            <Criterion>
               <HotelRef HotelCode="HFRI"/>
            </Criterion>
         </HotelSearchCriteria>
      </AvailRequestSegment>
   </AvailRequestSegments>
</OTA_HotelAvailRQ>
```

Enolični identifikator transakcije.

Podatki o izvoru.

Datuma prihod in odhoda.

Število oseb (10 – odrasli, 8 – otroci).

Koda hotela.

Odgovor

```
<OTA_HotelAvailRS xmlns="http://www.opentravel.org/OTA/2003/05" Version="5.000">
   <POS>
      <Source ISOCountry="si"/>
   </POS>
   <Success/>
   <RoomStays>
      <RoomStay IsAlternate="false">
         <RoomTypes>
            <RoomType
                  RoomID="DP123"
                  RoomType="Dvoposteljna soba"
                  RoomTypeCode="DP"
                  Quantity="2">
               <RoomDescription>
                  <Text>
                     Francosko ležišče, tuš, satelitska TV, balkon, pogled na morje.
                  </Text>
                  <Image>
                     DP.jpg
                  </Image>
               </RoomDescription>
            </RoomType>
         </RoomTypes>
         <RatePlans>
            <RatePlan RatePlanCode="NZ1">
               <RatePlanDescription>
                  <ListItem ListItem="1">
                     190 - 200 EUR
                  </ListItem>
                  <ListItem ListItem="2">
                     V ceni je vključena nočitev in zajtrk.
                     Cena je za celotno obdobje.
                  </ListItem>
               </RatePlanDescription>
```

Koda države se nanaša na izbran jezik.

Ali je alternativna ponudba.

Element se ponovi za vsako sobo.

Element predstavlja podatke o tipu sobe.

Opis tipa sobe.

Ime datoteke slike sobe.

Element predstavlja podatke o ceniku sobe.

Cena.

Opis cenika.

```
            </RatePlan>
         </RatePlans>
         <GuestCounts>
            <GuestCount Count="3"/>
            <GuestCount AgeQualifyingCode="10" Count="2"/>
            <GuestCount Age="13" Count="2"/>
         </GuestCounts>
         <TimeSpan End="2008-12-05" Start="2008-11-30"/>
         <BasicPropertyInfo HotelCode="HFRI"/>
         <ServiceRPHs>
            <ServiceRPH RPH="1"/>
         </ServiceRPHs>
      </RoomStay>
   </RoomStays>
   <Services>
      <Service ServiceRPH="1">
         <ServiceDetails>
            <Comments>
               <Comment>
                  <ListItem ListItem="1">
                     Nočitev z zajtrkom
                  </ListItem>
                  <ListItem ListItem="2">
                     Nočitev z zajtrkom.
                  </ListItem>
               </Comment>
            </Comments>
         </ServiceDetails>
      </Service>
   </Services>
</OTA_HotelAvailRS>
```

Element definira maksimalno število oseb v sobi (10 – odrasli).

Obdobje, na katerega se ponudba nanaša.

Koda hotela.

Številka storitve, ki jo soba vključuje.

Element se ponovi za vsako storitev. Element predstavlja podatke o storitvi.

Naziv storitve.

Opis storitve.

9.1.3. *Izračun cene izbrane ponudbe*

Zahteva

```
<OTA_HotelAvailRQ xmlns="http://www.opentravel.org/OTA/2003/05" Version="2.003">
   <POS>
      <Source ERSP_UserID="465465" TerminalID="127.0.0.1" ISOCountry="si">
         <BookingChannel Type="7">
            <CompanyName>hotel.si</CompanyName>
         </BookingChannel>
      </Source>
   </POS>
   <AvailRequestSegments>
      <AvailRequestSegment>
         <StayDateRange Start="2008-11-30" End="2008-12-05"/>
         <RoomStayCandidates>
            <RoomStayCandidate RoomID="DP123" RoomTypeCode="DP" Quantity="1">
               <GuestCounts>
                  <GuestCount AgeQualifyingCode="10" Count="2"/>
                  <GuestCount Age="13" Count="1"/>
               </GuestCounts>
            </RoomStayCandidate>
         </RoomStayCandidates>
         <RatePlanCandidates>
            <RatePlanCandidate RatePlanCode="NZ1">
               <HotelRefs HotelCode="HFRI"/>
            </RatePlanCandidate>
         </RatePlanCandidates>
      </AvailRequestSegment>
   </AvailRequestSegments>
</OTA_HotelAvailRQ>
```

Podatki o izvoru.

Element se ponovi za vsak izbran tip sobe.

Obdobje, na katerega se ponudba nanaša.

Seznam števila oseb.

Identifikator sobe, koda tipa sobe in količina izbranih sob navedenega tipa.

Koda cenika.

Koda hotela.

Odgovor

```
<OTA_HotelAvailRS xmlns="http://www.opentravel.org/OTA/2003/05" Version="5.000">
   <POS>
      <Source ISOCountry="si"/>
   </POS>
   <Success/>
   <RoomStays>
      <RoomStay>
         <RoomTypes>
            <RoomType RoomID="DP123" RoomTypeCode="DP" Quantity="1"/>
         </RoomTypes>
         <RatePlans>
            <RatePlan RatePlanCode="NZ1"/>
         </RatePlans>
         <GuestCounts>
            <GuestCount AgeQualifyingCode="10" Count="2"/>
            <GuestCount Age="13" Count="1"/>
         </GuestCounts>
         <TimeSpan End="2008-12-05" Start="2008-11-30"/>
         <Total AmountAfterTax="195.00" CurrencyCode="EUR"/>
         <BasicPropertyInfo HotelCode="HFRI"/>
      </RoomStay>
   </RoomStays>
</OTA_HotelAvailRS>
```

Koda države se nanaša na izbran jezik.

Element se ponovi za vsak izbran tip sobe.

Identifikator sobe, koda tipa sobe in količina izbranih sob navedenega tipa.

Koda cenika.

Seznam števila oseb.

Obdobje, na katerega se cena nanaša.

Koda hotela.

Končna cena.

9.1.4. *Določitev garancije*

Zahteva

```
<OTA_HotelResRQ xmlns="http://www.opentravel.org/OTA/2003/05" Version="4.000">
   <POS>
      <Source ERSP_UserID="465465" TerminalID="127.0.0.1" ISOCountry="si">
         <BookingChannel Type="7">
            <CompanyName>hotel.si</CompanyName>
         </BookingChannel>
      </Source>
   </POS>
   <HotelReservations>
      <HotelReservation>
         <RoomStays>
            <RoomStay>
               <RoomTypes>
                  <RoomType RoomID="DP123" RoomTypeCode="DP" Quantity="1"/>
               </RoomTypes>
               <RatePlans>
                  <RatePlan RatePlanCode="NZ1"/>
               </RatePlans>
               <GuestCounts IsPerRoom="true">
                  <GuestCount AgeQualifyingCode="10" Count="2"/>
                  <GuestCount Age="13" Count="1"/>
               </GuestCounts>
               <TimeSpan End="2008-12-05" Start="2008-11-30"/>
               <BasicPropertyInfo HotelCode="HFRI"/>
            </RoomStay>
         </RoomStays>
         <ResGuests>
            <ResGuest>
               <Profiles>
                  <ProfileInfo>
                     <Profile ProfileType="1">
                        <Customer>
                           <PersonName>
                              <NamePrefix>g.</NamePrefix>
                              <GivenName>Janez</GivenName>
                              <Surname>Novak</Surname>
                           </PersonName>
```

Podatki o izvoru.

Element se ponovi za vsak izbran tip sobe.

Identifikator sobe, koda tipa sobe in količina izbranih sob navedenega tipa.

Koda cenika.

Seznam števila oseb.

Obdobie.

Koda hotela.

Uporabnikovi osebni podatki. Tip 1 – stranka.

```
                        <Telephone PhoneNumber="+386 133 12 21"
                                   PhoneTechType="1"/>
                        <Telephone PhoneNumber="+386 133 12 22"
                                   PhoneTechType="3"/>
                        <Email>janez.novak@posta.si</Email>
                        <Address>
                           <AddressLine>Tržaška 25</AddressLine>
                           <CityName>Ljubljana</CityName>
                           <PostalCode>1000</PostalCode>
                           <StateProv/>
                           <CountryName>SI</CountryName>
                        </Address>
                     </Customer>
                     <UserID Type="1" ID="JNovak"/>
                  </Profile>
               </ProfileInfo>
            </Profiles>
         </ResGuest>
      </ResGuests>
   </HotelReservation>
</HotelReservations>
</OTA_HotelResRQ>
```

1 – telefon
3 – faks

Uporabniško ime uporabnika (v primeru, ko je uporabnik registriran uporabnik portala). Tip 1 – stranka.

Odgovor

```
<OTA_HotelResRS xmlns="http://www.opentravel.org/OTA/2003/05" Version="5.000">
   <POS>
      <Source ISOCountry="si"/>
   </POS>
   <Success/>
   <HotelReservations>
      <HotelReservation>
         <ResGlobalInfo>
            <DepositPayments>
               <GuaranteePayment GuaranteeType="Deposit">
                  <AmountPercent Amount="100.00" CurrencyCode="EUR"/>
               </GuaranteePayment>
            </DepositPayments>
            <HotelReservationIDs>
               <HotelReservationID ResID_Value="86048384"/>
            </HotelReservationIDs>
         </ResGlobalInfo>
      </HotelReservation>
   </HotelReservations>
</OTA_HotelResRS>
```

Koda države se nanaša na izbran jezik.

Zahtevan način garancije.

Višina akontacije.

Referenčna številka rezervacije, s katero so bile zadržane izbrane sobe.

9.1.5. *Potrditev rezervacije*

Zahteva

```
<OTA_HotelResRQ xmlns="http://www.opentravel.org/OTA/2003/05" Version="4.000"
   ResStatus="Commit">
   <POS>
      <Source ERSP_UserID="465465" TerminalID="127.0.0.1" ISOCountry="si">
         <BookingChannel Type="7">
            <CompanyName>hotel.si</CompanyName>
         </BookingChannel>
      </Source>
   </POS>
   <HotelReservations>
      <HotelReservation>
         <RoomStays>
            <RoomStay>
               <RoomTypes>
                  <RoomType RoomID="DP123" RoomTypeCode="DP" Quantity="1"/>
               </RoomTypes>
               <RatePlans>
                  <RatePlan RatePlanCode="NZ1"/>
```

Potrditev rezervacije.

Podatki o izvoru.

Element se ponovi za vsak izbran tip sobe.

Identifikator sobe, koda tipa sobe in količina izbranih sob navedenega tipa.

Koda cenika.

```
            </RatePlans>
            <GuestCounts IsPerRoom="true">
               <GuestCount AgeQualifyingCode="10" Count="2"/>
               <GuestCount Age="13" Count="1"/>
            </GuestCounts>
            <TimeSpan End="2008-12-05" Start="2008-11-30"/>
            <BasicPropertyInfo HotelCode="HFRI"/>
         </RoomStay>
      </RoomStays>
      <ResGuests>
         <ResGuest>
            <Profiles>
               <ProfileInfo>
                  <Profile ProfileType="1">
                     <Customer>
                        <PersonName>
                           <NamePrefix>g.</NamePrefix>
                           <GivenName>Janez</GivenName>
                           <Surname>Novak</Surname>
                        </PersonName>
                        <Telephone PhoneNumber="+386 133 12 21"
                                   PhoneTechType="1"/>
                        <Telephone PhoneNumber="+386 133 12 22"
                                   PhoneTechType="3"/>
                        <Email>janez.novak@posta.si</Email>
                        <Address>
                           <AddressLine>Tržaška 25</AddressLine>
                           <CityName>Ljubljana</CityName>
                           <PostalCode>1000</PostalCode>
                           <StateProv/>
                           <CountryName>SI</CountryName>
                        </Address>
                     </Customer>
                     <UserID Type="1" ID="JNovak"/>
                  </Profile>
               </ProfileInfo>
            </Profiles>
         </ResGuest>
      </ResGuests>
      <ResGlobalInfo>
         <DepositPayments>
            <GuaranteePayment GuaranteeType="Deposit">
               <AmountPercent Amount="100.00" CurrencyCode="EUR"/>
            </GuaranteePayment>
         </DepositPayments>
         <HotelReservationIDs>
            <HotelReservationID ResID_Value="86048384"/>
         </HotelReservationIDs>
      </ResGlobalInfo>
   </HotelReservation>
</HotelReservations>
</OTA_HotelResRQ>
```

Seznam števila oseb.

Obdobje.

Koda hotela.

Uporabnikovi osebni podatki.
Tip 1 – stranka.

1 – telefon
3 – faks

Uporabniško ime uporabnika (v primeru, ko je uporabnik registriran uporabnik portala).
Tip 1 – stranka.

Podatki o zahtevani garanciji.

Referenčna številka rezervacije, s katero so bile zadržane izbrane sobe.

Odgovor

```
<OTA_HotelResRS xmlns="http://www.opentravel.org/OTA/2003/05" Version="5.000"
   ResResponseType="Committed">
   <POS>
      <Source ISOCountry="si"/>
   </POS>
   <Success/>
   <HotelReservations>
      <HotelReservation>
         <ResGlobalInfo>
            <HotelReservationIDs>
               <HotelReservationID ResID_Value="86048384"/>
            </HotelReservationIDs>
         </ResGlobalInfo>
         <WrittenConfInst>
            <SupplementalData Name="Confiramation">
               <Text>
                  Zahvaljujemo se Vam za rezervacijo v hotelu FRI.
                  Podrobnosti si oglejte v priponki.

                  Lep pozdrav,
                  Vaš hotel
               </Text>
               <Image>
                  ...
               </Image>
            </SupplementalData>
         </WrittenConfInst>
      </HotelReservation>
   </HotelReservations>
</OTA_HotelResRS>
```

Rezervacija je potrjena.

Koda države se nanaša na izbran jezik.

Referenčna številka rezervaciie.

Tekst za elektronsko pošto.

Slika potrdilnega pisma v Base64 predstavitvi.

9.2. Primer sporočil postopka posodabljanja statične vsebine

Zahteva

```
<OTA_HotelDescriptiveContentNotifRQ xmlns="http://www.opentravel.org/OTA/2003/05"
   Version="6.003">
   <HotelDescriptiveContents>
      <HotelDescriptiveContent HotelCode="HFRI">
         <FacilityInfo>
            <GuestRooms>
               <GuestRoom>
                  <MultimediaDescriptions>
                     <MultimediaDescription>
                        <ImageItems>
                           <ImageItem>
                              <ImageFormat FileName="EP.jpg"/>
                              <Description>
                                 ...
                              </Description>
                           </ImageItem>
                           <ImageItem>
                              <ImageFormat FileName="DP.jpg"/>
                              <Description>
                                 ...
                              </Description>
                           </ImageItem>
                        </ImageItems>
                     </MultimediaDescription>
                  </MultimediaDescriptions>
               </GuestRoom>
            </GuestRooms>
         </FacilityInfo>
      </HotelDescriptiveContent>
   </HotelDescriptiveContents>
</OTA_HotelDescriptiveContentNotifRQ>
```

Koda hotela.

Ime datoteke.

Datoteka slike v Base64 predstavitvi.

Odgovor

```
<OTA_HotelDescriptiveContentNotifRS xmlns="http://www.opentravel.org/OTA/2003/05"
   Version="1.008">
   <Success/>
</OTA_HotelDescriptiveContentNotifRS>
```

9.3. Primer sporočil postopka preverjanja prisotnosti odjemalcev

Zahteva

```
<OTA_PingRQ xmlns="http://www.opentravel.org/OTA/2003/05" Version="1.005">
   <EchoData>Are you There</EchoData>
</OTA_PingRQ>
```

Odgovor

```
<OTA_PingRS xmlns="http://www.opentravel.org/OTA/2003/05" Version="1.007">
   <Success/>
   <EchoData>Are you There</EchoData>
</OTA_PingRS>
```

Uporabljeni viri

Literatura:

[1] S. Choudhury, K.Bhatnagar, E. Haque, *Public key infrastructure implementation and design*, New York: M&T Books, 2002.
[2] S. Das, *Global Distribution Systems in Present Times*, http://www.hotel-online.com/News/PR2002_4th/Oct02_GDS.html (21.06.2008).
[3] M. Dworkin, *Recomendation for Block Cipher Modes of Operation*, Washington: National Institute of Standards and Technology, 2001.
[4] D. E. Eastlake, Kitty Niles, *Secure XML: The New Syntax for Signatures and Encryption*, Boston [etc.]: Addison Wesley, 2002.
[5] OpenTravel Alliance, *Message Users Guide 2008A*, različica 1.0, Junij 2008.
[6] M. Stamp, *Information security principles and practice*, Hoboken (NJ) : Wiley-Interscience, 2006.

Internetni viri:

[7] Computer Reservation System, http://en.wikipedia.org/wiki/Computer_reservations_system, (20.06.2008).
[8] Extensible Markup Language (XML) 1.0 (Fourth Edition), http://www.w3.org/TR/2006/REC-xml-20060816/, (31.7.2008).
[9] Online Certificate Status Protocol, http://www.ietf.org/rfc/rfc2560.txt, (20.7.2008).
[10] OpenTravel (uradna stran), http://www.opentravel.org/, (01.08.2008).
[11] OpenTravel Alliance (OTA), http://xml.coverpages.org/openTravel.html, (01.08.2008).
[12] Payment Service Provider, http://en.wikipedia.org/wiki/E-Payment_Providers, (30.6.2008).
[13] Payment Service Providers on the Net, http://www.papron.net, (27.06.2008).
[14] SOAP Version 1.2 Part 1: Messaging Framework (Second Edition), http://www.w3.org/TR/soap12-part1/, (18.08.2008).
[15] XML Encryption Syntax and Processing, http://www.w3.org/TR/xmlenc-core/, (05.08.2008).
[16] XML Signature Syntax and Processing (Second Edition), http://www.w3.org/TR/xmldsig-core/, (05.08.2008).

Izjava o avtorstvu

Izjavljam, da sem diplomsko delo izdelal samostojno pod vodstvom mentorja prof. dr. Nikolaja Zimica. Izkazano pomoč drugih sodelavcev sem v celoti navedel v zahvali.

Blaž Hafnar

Ljubljana, 9.9.2008